INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Trusted Civitas:
Client Trust in CIVITAS Electronic Voting Protocol

João Miguel Barros da Silva Mendes

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Júri

Presidente: Prof. Doutor Nuno João Neves Mamede
Orientador: Prof. Doutor Pedro Miguel dos Santos Alves Madeira Adão
Vogal: Prof. Doutor Carlos Nuno da Cruz Ribeiro

Junho de 2011

Acknowledgements

I would like to thank everyone who helped me accomplish this work, especially my supervising professor Pedro Adão, for embracing the idea of this work from the bottom line to the top, for guiding me throughout its evolution, clarifying my doubts (some of them philosophical, others very annoying), having all the time in the world for our meetings, for his patience, reviewing every draft produced, and motivating me towards the conclusion of this thesis.

I would also like to mention and be graceful to my girlfriend (at the begin of the thesis) and wife (near the end of it) for all the support, patience (lots of it, believe me), helping me maintain my mental sanity and supporting me through this time of my life.

I like to give a word of appreciation to my father, mother and sister, for their endless efforts in my education and their commitment in giving me all the time and resources to meet the ending of this thesis.

Abstract

Abstract. The electronic vote promises the possibility of a convenient, efficient and safe way to capture and count votes in an election. It is necessary to make the elections accessible to all, using new forms of voting that facilitate life for voters. Issues in the logistics involved in paper elections, their cost and the resources involved are very large, in addition to the time needed for all phases of the electoral process are the reason why the electronic vote is so important, especially when implemented remotely. The Civitas protocol tries to reach all properties of the electronic voting, including resistance to coercion, which is the strongest of all privacy properties. Despite everything, the Civitas is not yet ready to be used in the real world, because it still has unsolved issues, in particular regarding the trust in voting client. This thesis describes a potential solution to some of the issues raised by the Civitas protocol. The proposed solution is based in the use of smart cards and in the CodeVoting system. The latter includes the exchange of codes between the central electoral server and the voter, in order to confirm the reception of his vote. The major difference to the original CodeVoting system consists in a simplified procedure that does not require the existence of code cards.

Keywords: Electronic Voting, Civitas, CodeVoting, Privacy, Security, Smart Card, Voter, Coercer, Coercion-Resistance, Trust in Client Voting

Resumo A votação electrónica promete a possibilidade de uma conveniente, eficaz e segura forma de capturar e contar votos numa eleição. Não se pode esquecer que é necessário tornar as eleições acessíveis a todos, usando novas formas de votação que facilitem a vida aos eleitores. Problemas com a logística envolvida nas eleições em papel, cujo custo e recursos envolvidos são bastante grandes, além do tempo necessário para todas as fases do processo eleitoral são a razão por quê a Votação Electrónica é tão importante, especialmente quando implementada remotamente. O protocolo Civitas tenta alcançar todas as propriedades dea votação electrónica, nomeadamente a resistência à coacção, que é a propriedade mais forte das propriedades de privacidade. Apesar de tudo, o Civitas não está ainda preparado para ser usado no mundo real, pois tem ainda questões em aberto por solucionar, nomeadamente na confiança no cliente de voto. Esta tese descreve uma potencial solução para algumas das questões levantadas pelo protocolo Civitas. A solução proposta baseia-se no uso de smart cards e no sistema CodeVoting. Este último inclui a troca de códigos entre o servidor eleitoral e o eleitor, para confirmar a recepção do seu voto. A principal diferença para o sistema CodeVoting original consiste num processo simplificado que não requer a existência de code cards.

Palavras Chave: Votação Electrónica, Civitas, CodeVoting, Privacidade, Segurança, Smart Card, Eleitor, Coactor, Resistência à Coacção, Confiança no Cliente de Voto

Table of Contents

1 Introduction 11
1.1 Traditional Paper Voting 11
1.2 Types of Electronic Voting 13
1.3 Organization of the Thesis 15
2 Electronic Voting 16
2.1 Seven Principles for Secure E-Voting 16
2.2 Security Properties 16
2.3 Verifiability 17
2.4 Accountability 18
2.5 Privacy 18
2.6 Electronic Voting in Practice 19
3 Cryptographic Schemes used in Electronic Voting 29
3.1 Homomorphic Encryption 29
3.2 Blind Signature 29
3.3 Mix Network 31
4 End-to-End Voter Verifiable Systems 32
4.1 Prêt à Voter 32
4.2 Helios: Web-based Open-Audit Voting 32
4.3 Civitas 35
4.4 Which one to choose? 37
5 Civitas, Towards a Secure Voting System? 38
5.1 Design 38
5.2 Verifying an election 41
5.3 Security 41
5.4 How Civitas works 42
6 Where Do We Stand? 45
6.1 Unsolved Issues 46
6.2 Old Assumptions 48
6.3 Goals 50
7 Trusting in Voting Clients 51
7.1 Unreliable Vote Client 51
7.2 CodeVoting 54
8 Tools for the Solution 59
8.1 New and Changed Entities 59
8.2 Registration 60
8.3 Trust in Voting Client 62
8.4 Remote Voting 63
8.5 Ballot Box and CodeCardReplier 64
8.6 Tabulation 65
8.7 Coercion-Resistance 66
8.8 Alternative Solutions 68

8.9 New Trust Assumptions 69
8.10 Voting Steps 71
9 Future Work 73
9.1 Usability 73
9.2 Anonymity Network 73
9.3 Removing hardware dependency 73
9.4 High Availability 73
10 Conclusion 74
References 76
A Smart Cards 80
A.1 Types of Smart Cards 80
A.2 The Microprocessor Smart Card 80
A.3 The Micromodule 81
A.4 Smart Card Operating System 82
B Citizen's Card 84
B.1 Chip Physical Characteristics 85
C Traffic Analysis 86
C.1 Countermeasures 86
C.2 Onion Routing 86
C.3 Tor, Anonymity Network 88
D Threshold cryptosystem 90

List of Figures

1 Paper ballot box. 11
2 Electronic voting machine used in the 2005 Brazilian referendum. 20
3 Simulator of *Urna Electrônica*. 22
4 The envelope method. 25
5 Estonia e-vote design. 26
6 Homomorphic Encryption. 29
7 Blind signature using RSA. 30
8 Basic decryption mix net. 31
9 Helios' voting interface. 34
10 Civitas architecture. 36
11 Civitas architecture. 38
12 CodeVoting components. 54
13 Example of a ballot (left) and a CodeCard (right). 56
14 New architecture based in Civitas. 59
15 Portuguese citizen's card. 61
16 Smart Card Reader with keypad and screen. 63
17 Election's web page with candidates. 71
18 Smart card pinout. 81
19 Portuguese Citizen's card. 84
20 Onion routing. 87
21 How Tor works. 88

1 Introduction

The current state of secure electronic voting is far from perfect. Major commercial electronic voting systems fail to offer strong security guarantees, a fact well known by the community. And to some extent, the research community has been pessimistic about the feasibility of building a secure voting system.

Electronic voting protocols are formal protocols that specify the messages sent between the voters and administrators. Such protocols have been studied for several decades. They offer the possibility of abstract analysis of the voting system against formally-stated properties.

1.1 Traditional Paper Voting

A ballot is a device (physical or electronic) used to record voters' choices. Each voter uses one ballot, and ballots are not shared, therefore each ballot is unique. In the simplest elections, a ballot may be a simple scrap of paper in which each voter writes or selects the name of a candidate, but in real world usage, for example, governmental elections, it uses pre-printed ballots to protect the secrecy of the voters. The voter casts his ballot in a box at a polling station. This box is called *ballot box*.

Fig. 1. Paper ballot box.

A ballot box is a temporarily sealed container, usually cuboid though sometimes a tamper resistant bag, with a narrow slot in the top, large enough to

accept a ballot paper in an election but it must prevents anyone from accessing the votes cast until the close of the voting period (e.g. Figure 1).

Humans have a profound affinity for that which they can see and touch. This results in a deep reverence for the printed word, whether it is true or false, and explains the comfort people derive from paper receipts. There are very few paper documents that have preclusive legal effect, meaning that the writing on the face of the document is not subject to challenge.

There are basically four types of paper records:

1. *Bearer instruments* – Examples: currency, bearer bonds, checks, movie tickets. Here the instrument itself entitles the bearer to rights with no further inquiry into his bona fides. Title to the document passes with possession. These instruments are extremely convenient for transactions because they can convey rights and title instantaneously without resort to offline records and databases. They are also a frequent subject of theft;

2. *Receipts* – Instead of being a instrument used to effectuate a transaction, a receipt is merely evidence of the transaction. As such, a receipt takes its place among all of the other forms of evidence, including spoken words, videotapes, witness testimony, business records, computer databases, among other things. The contents of a receipt may be challenged or rebutted and the effect it has will be determined by the trier of fact;

3. *Business records* – These are notes kept by a business as part of its operations. Records kept in the ordinary course of business are admissible as evidence, but they are only evidence and may be challenged. They differ from receipts in that they are created by one party to a transaction and but are not normally reviewed for correctness by the other party. A dispute between a bank and its customer over a questioned ATM transaction usually turns on the question of which records are more credible, the customer's paper receipt or the bank's computerized business records;

4. *Ballots* – A ballot is an expression by a person indicating how she wishes to cast her vote. A ballot is a unique document defined by election law and is itself only evidence of how a voter wanted to vote. A ballot may be challenged on many grounds, including an allegation that the voter was not entitled to vote, the ballot was mismarked, the voter voted in the wrong precinct, the voter cast votes for candidates she was not entitled to vote for, the ballot was mangled, defaced or was otherwise unreadable. In many, but not all, states when the content of a ballot is disputed, a court is required to determine the intent of the voter in marking the ballot and is not bound by that the ballot actually says.

There are numerous other forms of paper records, such as documents of title, licenses, wills, diplomas, written offers, etc., that are not relevant to the discussion here. The question is what desirable properties, if any, do paper records have that would cause us to prefer them over electronic ones for voting.

Issues There are various problems inherently in paper-based systems:

- The logistics involved in elections in paper, cost and resources involved are very large;
- time needed for all phases of the electoral process is very huge (e.g. printing votes, tally votes).

Voting system attackers operate with limited information during the voting period. For example, winners are not known during the voting day and there is little value in manipulating a contest that will be won anyway. Similarly, victory margin cannot be predicted and it is not useful to manipulate votes and still lose the target contest.

On the other hand, it is well-known that paper ballots are vulnerable to malice after the voting period ends, when the outcome is clear, the margin is known, and malicious attacks can be precisely honed to accomplish the desired intent. The questions that must be asked now are:

- What are the inherent security properties in paper used?
- How does electronic voting solve or reduce the problems with paper ballots?

1.2 Types of Electronic Voting

Three approaches to the problem of electronic voting have been proposed so far [53]:

- *Poll-site voting* – commonly seen as DRE's (direct recording electronic) – special voting machines with dedicated software are installed in voting places at polling stations. Voters can cast votes by interacting with such a machine, and in some cases he or she can receive a receipt for verification. The terminal and the environment can be controlled. In some cases, some steps of the protocol may be performed by an election official, for instance the voter can be personally authorized. Several well designed solutions have been proposed so far, including [4,16,51,58];
- *Kiosk voting* – voting takes place through publicly available terminals (e.g. sophisticated ATM or dedicated state-owned machines). In this scenario only the terminal can be controlled;
- *Voting via Internet* – performed by a client-server application, run by voter's PC, mobile phone, PDA, smart card, and on the server side, by trusted authority or authorities. Neither the terminal not the environment can be controlled [54].

In the last few years several experiences have been conducted in order to facilitate the voting process in elections. Such facilitations were introduced by new ways of expressing votes besides the traditional paper-based. Examples of new voting interfaces and systems are touch screens, SMS (short message service) from cellular phones and distributed voting systems using the Internet [39].

It's this last one that has great potential to succeed and prevail. The remote voting approach is the most convenient and cost-effective. It also reflects the needs of the modern society. Nonetheless, it is considerably more challenging as more attentions have to take into account, such as various cyber-attacks (possibly launched from a hostile country), and less control of the voter [57]. Vulnerabilities of voter's PC may open the door to many serious abuses, e.g., automated vote selling or malicious changing of votes. Several countermeasures have been proposed to minimize the trust put in voter's PC. Apart from the expensive ones (trusted hardware) and the idealistic ones (clean operating system), code sheets and test ballots seem to be promising [39,54]. Code sheets impose a complete asymmetry in the computational sense – no computations are done on the voter's side. This is achieved by providing voters with ballots that contain unique codes representing candidates (different set of codes for each ballot). Each candidate code has a verification code assigned to it. The PC is used to pass the entered code on to the election authority, which returns the relevant verification code. The response is displayed by the PC, integrity of remotely casted vote. Honest election authorities may prevent cyber-attacks, but a dishonest one can try to influence elections results or breach voter's privacy. There are attempts to solve the untrusted platform problem by utilizing trusted hardware [47].

Internet voting systems are appealing for several reasons:

- people are getting more used to work with computers to do all sort of things, namely sensitive operations such as shopping and home banking;
- they allow people to vote far from where they usually live, helping to reduce abstention rates;
- they may support arbitrary voting ballots and check their correct fulfillment during the voting process.

Although this sounds promising, Internet voting systems face several problems that prevent their widespread use today. The problems can be broadly divided in three main classes.

The first class of problems includes security and fault tolerance issues inherited from the current Internet architecture. Vital services, such as DNS name resolution, can be tampered in order to mislead users into spoofing servers. IP routing mechanisms and protocols, managed by many different organizations, should deal with partial communication outages, however communication problems may arise.

The second class of problems includes issues that are specific to voting protocols. These problems derive from the assumptions of the protocols about the execution environment, namely:

- Client machines used by voters must be trusted, in order to act as *trusted agents*, which is hard to ensure in personal or multi-user computers with general-purpose commercial operating systems;

– Servers controlling the voting process cannot fail, become unreachable or pervert the voting protocol. The protocol perversion can be done either by not reacting properly to client requests or by trying to influence the election by acting as a voter;
– The voting protocol is not disturbed by communication problems or machine failures.

The third class of problems includes those difficulties that may be created by specific attacks against a voting protocol or a running election. Such attacks may try to get some useful outcome, by subverting the voting protocol, or simply ruin an election using DoS[1] attacks against the participating machines or applications. Another kind of attack is the coercion of voters, which can happen if they can vote anywhere without supervision of electoral committees.

1.3 Organization of the Thesis

This paper is organized as follows:

– Section 2 offer a brief introduction into electronic voting, its security properties, Privacy and examples of electronic voting throughout the world;
– Section 3 define some crypographic schemes used in electronic voting;
– Section 4 describe and shows some voting protocols and their properties;
– Section 5 show the Civitas' voting protocol in detail;
– Section 6 and 7 describe various issues of Civitas and some solutions for them;
– Section 8 and 9 validate an architecture proposed to solve trust in client voting in Civitas protocol and provide possible interesting directions for future research using this work;
– Section 10 display the final conclusions about this work;
– In the end there are some appendixes with some related work.

[1] *DoS* stands for Denial of Service.

2 Electronic Voting

Electronic voting is a term encompassing several different types of voting, embracing both electronic means of casting a vote and electronic means of counting votes. Electronic voting is also called e-voting and in this section this matter will be approached, through several aspects of it.

2.1 Seven Principles for Secure E-Voting

Frank Gerlach [64] has enumerated the seven principles for secure E-Voting:

1. *Proven Security*: All protocols and techniques must be proved and verified to be secure;
2. *Trustworthy Design Responsibility*: The e-voting responsibility should be assigned to a highly reliable and trustworthy organization;
3. *Published Source Code*: Application source code should be made public;
4. *Vote Verification*: All voters should be able to verify their votes;
5. *Voter Accessibility*: A full list of voters must be available to all citizens;
6. *Ensure Anonymization*: No one should be able to see or identify whatever a voter casts into the network;
7. *Expert Oversight*: The election system must be handled by experts. For example, experts must be employed to handle DoS attacks.

Mathematically secure methods for e-voting are conceivable, and voters' confidence can be increased by allowing them to verify their votes in the tally. As with any cryptographic method, the system must still rely on a chain of mutual trust, which will always be necessary.

2.2 Security Properties

There are several properties that an Electronic Voting System should have, in order to be secure and therefore usable in real-world [24,43]:

- *Eligibility*: only legitimate voters can vote;
- *Fairness*: no early results can be obtained as they could influence the remaining voters;
- *Individual Verifiability*: a voter can verify that her vote was really counted (at some point at the elections);
- *Universal Verifiability*: the published outcome of the tally is really the sum of all the votes;

- *Vote-Privacy*: the fact that a particular voter voted in a particular way is not revealed to anyone;
- *Receipt-Freeness*: a voter does not gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way;
- *Coercion-Resistance*: a voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

Another three non-core properties that are important in remote electronic voting [39] are:

- *Collusion Resistance*: No electoral entity or group of entities can work in a conspiracy to introduce votes; no electoral entity or group of entities can work in a conspiracy to prevent others from voting;
- *Availability*: The system works properly as long as the poll stands; any voter can participate from the beginning until the end of the poll;
- *Resume Ability*: The system allows any voter who had interrupted his voting process to resume or restart it as long as the poll stands.

This last three properties are important for remote voting and important properties for availability.

2.3 Verifiability

In the electronic voting literature, verifiability addresses the security requirement of the integrity of the election result. First of all, this means that it is possible for the voter to audit that his vote has been properly created (in general encrypted), stored, and tallied (*individual verifiability*). Further, this means that everyone can audit the fact that only votes from eligible voters are stored in a ballot box, and that all stored votes are properly tallied (*universal verifiability*). Systems providing both forms are called End-to-End[2] verifiable.

Traditionally, votes were cast on paper and counted by hand. Voters were confident that the marks they made on ballots reflected their intended vote. Voting machines that used levers and punch card systems also provided voters with a high degree of confidence that their vote is cast as intended. Because they are paperless, DRE systems raise the question: how can one know that when a voter chooses a particular candidate on the screen, a vote for that candidate is recorded?

The most pressing verifiability problem with the use of computerized voting is that the systems are provided by private companies, and the government usually has no oversight into the production of the systems beyond choosing whether or not to use them. It is easy to imagine a scenario whereby a malicious, or simply a careless, programmer sets up a situation in which votes for Candidate A appear

[2] See section 4 for more details about these systems.

to go to Candidate A as far as the user's display, but actually are tabulated for Candidate B.

More critically, suppose that the same situation occurred, but only with a small percentage of the votes cast. The use of a DRE system in this case would be catastrophic, because there would be no way to review voting records to conduct a recount. With paper ballots, voters can visually inspect the official record, but with computer-based voting this is next to impossible. A simple solution to this problem is to provide the user with a printed record of the votes electronically recorded. Before leaving the polling place, the voter would be required to certify the contents of the paper record and place it into a ballot box. The printed records could then be manually counted in the event of a challenge, and this procedure would foil any attempt at falsifying votes internally to the voting system.

2.4 Accountability

Cryptographic tasks and protocols, such as contract signing, voting, auction, identity-based encryption, and certain forms of secure multi-party computation, involve the use of trusted or partial trusted parties, such as notaries and authorities. It is crucial that such parties can be held accountable in case they misbehave as this is a strong incentive for such parties to follow the protocol. Unfortunately, there is no general and convincing definition of accountability that would allow us to assess the level of accountability a protocol provides.

Ralf Küsters et al. [44] definition reveals that accountability is closely related to verifiability. He proves that verifiability can be interpreted as a weak form of accountability and that verifiability are of independent interest. Verifiability can be interpreted as a restricted form of accountability. The relationship offers a deeper understanding of the two notions and allows to derive statements for verifiability from statements for accountability, as illustrated by our case studies. Accountability is the property protocol designers should aim for, not just verifiability, which on its own is often too weak a property in practice: if a protocol participant (rightly) complains that something went wrong, then it should be possible to (rightly) hold specific protocol participants accountable for their misbehaviour, and by this, resolve the dispute.

2.5 Privacy

Privacy is very important in electronic voting.

The last three security properties presented before are broadly privacy-type properties since they guarantee that the link between the voter and her vote is not revealed by the protocol.

- *Vote-Privacy*: the fact that a particular voter voted in a particular way is not revealed to anyone;

- *Receipt-Freeness*: a voter does not gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way;
- *Coercion-Resistance*: a voter, even if cooperating with a coercer, cannot prove to him that she voted in a certain way.

The weakest of the three, called *vote-privacy*, roughly states that the fact that if a voter voted in a particular way, it is not revealed to anyone. When stated in this simple way, however, the property is in general false, because if all the voters vote unanimously then everyone will get to know how everyone else voted. No party receives information which would allow them to distinguish one situation from another one in which two voters swap their votes.

Receipt-freeness Receipt-freeness says that the voter does not obtain any artefact (a *receipt*) which can be used later to prove to another party how she voted. Such a receipt may be intentional or unintentional on the part of the designer of the system. Unintentional receipts might include nonces or keys which the voter is given during the protocol. Receipt-freeness is a stronger property than vote-privacy. Intuitively, vote-privacy says that an attacker cannot discern how a voter votes from any information that the voter necessarily reveals during the course of the election. Receipt-freeness says the same thing even if the voter voluntarily reveals additional information.

Coercion-resistance Coercion-resistance is the third and strongest of the three privacy properties. Again, it says that the link between a voter and her vote cannot be established by an attacker, this time even if the voter cooperates with the attacker during the election process. Such cooperation can include giving to the attacker any data which she gets during the voting process, and using data which the attacker provides in return. When analysing coercion-resistance, it is assumed that the voter and the attacker can communicate and exchange data at any time during the election process. Coercion-resistance is intuitively stronger than receipt-freeness, since the attacker has more capabilities. Obviously, the voter can simply tell an attacker how she voted, but unless she provides convincing evidence the attacker has no reason to believe her. Receipt-freeness and coercion-resistance assert that she cannot provide convincing evidence. Coercion-resistance cannot possibly hold if the coercer can physically vote on behalf of the voter. Some mechanism is necessary for isolating the voter from the coercer at the moment she casts her vote.

2.6 Electronic Voting in Practice

This section will discuss about two examples of electronic voting used in real-world:

- The Brazilian with direct recording electronic;
- The Estonian, with Internet Voting.

Direct recording electronic (DRE) Voting machines are the total combination of mechanical, electromechanical, or electronic equipment (including software, firmware, and documentation required to program control, and support equipment), that is used to define ballots, to cast and count votes, to report or display election results and to maintain and produce any audit trail information. The first voting machines were mechanical but it is increasingly more common to use electronic voting machines.

Fig. 2. Electronic voting machine used in the 2005 Brazilian referendum.

A DRE voting system records votes by means of an electronic display provided with mechanical or electro-optical components that can be activated by the voter, that processes voter selections by means of a computer program, and that records that processed voting data in memory components. It produces a tabulation of the voting data that is stored in a removable memory component and may also provide printed renditions of the data. The system may further provide a means for transmitting the processed vote data to a central location in individual or accumulated forms for consolidating and reporting results from precincts at a central location. DRE systems additionally can produce a paper ballot printout that can be verified by the voter before they cast their ballot.

Electronic Voting in Brazil

Brazil was the first country in the world to have fully electronic elections. Electronic voting was introduced in Brazil in 1996 when the first tests were

carried out in the state of Santa Catarina. The chief goal of the Brazilian voting machine is its extreme simplicity, attempting to be as straightforward as a public phone booth.

The Brazilian voting machine accumulates the three steps (elector identification, secret voting and results in each machine) in only one process, as then it can eliminate the public documents which were considered as a source of fraud.

History In Brazil the vote is mandatory and there exists only one agency, the TSE (*Tribunal Superior Eletoral* – Electoral Superior Court), that exerts the three republican powers of Regulating, Administering and Judging of the Electoral process. This unusual accumulation of powers, results in the anti-democratic centralization of the decisions in the hands of a few and provokes the lack of transparency of the process. Even the electoral laws were written and approved without the civil society being able to, in fact, express any opinion or participate [13].

In 1982, at the peak of the Military Regime of Exception, the first attempt at computerization of the elections happened in what became known as the Proconsult Case. The experience was disastrous, with an attempt of fraud by military agents. But the *esprit d'corps* of the *Justiça Eleitoral* prevailed, and stifled the inquiry. Until today it denies that it happened, banishing this case from its official history. The computerization of the totaling of votes continued to develop in the following elections.

In 1985, the lobby of the TSE in the National Congress obtained the fast approval of Law 7444/85 that commanded the unification of national Voter Registration, with the use of the computers, and gave to the TSE powers to be able to prescribe the re-registration process. The TSE decided, alone, to eliminate the photo of the voter on the Voter ID Cards, creating an enormous security gap, making possible a simple fraud where any person can vote using someone else's card. This error of Electoral Justice remained for 20 years and was only started to be corrected in November of 2005.

In 1995, with renewed lobbying by the TSE of the National Congress a law was passed, written six months earlier by an internal work group of the TSE, resulting in Law 9100/95, which allowed the use of electronic voting machines, and gave to the TSE the power to regulate their use. The TSE opted to use direct recording electronic (DRE) machines without a paper ballot confirmed by the voter. It also opted for the identification of the voter at the voting machine itself, creating a new security gap for the inviolability of the vote. This machine came to be called the *Urna Eletrônica*, the electronic ballot box.

In 1996, $\frac{1}{3}$ of the electorate, approximately 35 million voters, voted in the new DRE's without a paper ballot verified by the voter. In 1998, the electronic ballot boxes were used by $\frac{2}{3}$ of the voters and in 2000, by 100%.

In 1999, the first project of law appeared in the Federal Senate that compelled voting machines to print the vote for verification by the voter, created the audit requirement of 3% of the ballot boxes which would be selected at random after the election, impeded the identification of the individual voters at the ma-

chine where they voted, and compelled the use of open source software for the electronic ballot boxes.

The Minister-Judges of the TSE came back to exert a strong lobby in the National Congress and obtained, in only two days of 2001, the approval of seven amendments to the proposal that created Law 10480/02 which postponed the application of the VVPB (Voter-Verifiable Paper Ballot) until 2004, ordered the random selection of the ballot boxes to be audited be done before the elections, allowed the identification of the voter at the voting machine, and allowed the TSE to use undisclosed computer programs in the DRE's whose source-code is not presented to anyone for inspection.

The pressure of the TSE in the National Congress against the auditing of the electronic verification continued in 2003 and in less than six months, obtained approval of Law 10708/03 that revoked the VVPB and the auditing of the electronic verification of the votes, even before they came into effect in 2004. In this new law, the identification of the voters remained at the voting machines, and it strengthened the authorization for use of secret software by the TSE.

Fig. 3. Simulator of *Urna Electrônica*.

A biometric voting machine To ensure that Brazilian elections are safer, in the 2008 elections, a new method of identification of voters began to be deployed. Rather than just produce documents that can be falsified, citizens are identified by fingerprints. The digital recording has started and, when completed, it is supposed to be one of the world's more accurate and advanced databases.

In the first round of 2010's elections (see Figure 3.) the results were called before midnight on the day after voting, and the run-off election results broke all records for speed. At 8:04 pm (Brasilia), just over one hour after the polling stations across the country were closed, the president of the Superior Electoral Tribunal (TSE), Ricardo Lewandowski, announced that Dilma Rousseff (PT - Workers Party) had beaten José Serra (PSDB - Brazilian Social Democracy Party) in the Presidential election[3].

According to the TSE, of the 420000 electronic voting machines used in 2010, only 2244 had to be replaced in the first round, which represents 0.56% of the total, and 1609 in the second round (or 0.4%).Overall, in only nine sections of the country, in the the states of Amazonas, Piaui, Sao Paulo (two in each round), Parana, Santa Catarina, Rio Grande do Sul, Rio de Janeiro and Sergipe, did voting have to be done manually.

Major Problems In Brazil, the TSE combines the duties of inspection and administration of the election process and the power to judge all election material resources, even those which are contrary to its administrative acts. Due to human nature, this accumulation of powers, which are not found in mature democratic regimes, naturally lead to authoritarianism and lack of transparency. Because of this, official requests for Security Penetration Tests presented repeated times by some political parties have been systematically blocked by this elections super-organ by the simple fact that it had the power to centralize the decision and obstruct such tests [13].

In this way, it can be affirmed that the security (and insecurity) resources of the Brazilian DRE's are similar, when compared to the *Diebold TSx* model:

1. They do not create a printed vote verified by the voter hence not permitting an audit of the vote-counting, and as consequence they are highly dependent on the trusting of the software;
2. They have a BIOS chip (with boot function) installed in a socket and re-recordable by software, as specified in the bidding specifications;
3. They have a *BIOS extension* accomplished by a *jumper* on the motherboard, where the BIOS-extension was deactivated, which permitted the boot from the external socket from an external flash-card memory;
4. It is possible to execute *batch file* software recorded on a diskette;

[3] The simulator can be seen at: `http://www.tse.jus.br/internet/eleicoes/urna_eletronica/simulador_Votacao_2010/br.htm`

5. The application program which verifies the internal integrity of the system is itself extremely vulnerable to adulterations;
6. The system of seals and closure of the box are simple and permit access to the socket of the internal memory cartridge.

This set of insecure characteristics of the Brazilian DRE's make them just as subject to fraud as their American similar models.

I-Vote: Estonia's Case

The idea of having electronic voting in Estonia originated in early 2001 and quickly gained popularity among heads of the then pro-actively coalition government of the small northeastern European country. The realization of the project came in the October 2005 local elections when Estonia became the first country to have legally binding general elections using the Internet as a mean of casting the vote [48,49].

The main principle of e-voting is that it must be as similar to regular voting as possible, compliant with election legislation and principles and be at least as secure as regular voting. Therefore e-voting must be uniform and secret, only eligible persons must be allowed to (e-)vote, every voter should be able to cast only one vote, and a voter must not be able to prove in favour of whom he/she voted. In addition to this the collection of votes must be secure, reliable and accountable.

According to Estonian election legislation e-voting takes place from 6th to 4th day before Election Day and the following requirements are laid out:

1. On advance polling days, voters may vote electronically on the web page of the National Electoral Committee. A voter shall vote himself or herself;
2. A voter shall identify himself or herself using the certificate on his or her identity card which enables digital identification;
3. After identification of the voter, the consolidated list of candidates in the electoral district of the residence of the voter shall be displayed to the voter on the web page;
4. The voter shall indicate on the web page the candidate in the electoral district of his or her residence for whom he or she wishes to vote and shall confirm the vote by signing it digitally using the certificate on his or her identity card which enables digital signing;
5. A notice that the vote has been taken into account shall be displayed to the voter on the web page;
6. Voter may change his or her electronic vote during the advance voting period from 6th to 4th day before Election Day:
   (a) by voting electronically;
   (b) by voting in polling station.

The following principles are specific to e-voting:

- ID-cards are used for voter identification;
- Possibility of electronic re-vote: e-voter can cast his vote again and the previous vote will be deleted;
- The priority of traditional voting: should the voter go to polling station on advance voting day and cast a vote, his or her electronically casted vote shall be deleted.

The envelope method The main principle of e-voting system lies in double-envelope scheme illustrated in the Figure 4. The scheme is familiar from the postal voting in some countries. A voter seals his choice into inner blank envelope (encrypts it) and puts this envelope into bigger one writing his name/address on it (digitally signs it). These bigger envelopes are collected to central site. Prior vote counting, outer envelopes with personal data (digital signatures) are removed and anonymous white envelopes (encrypted votes) are sent to counting process which outputs summarized voting results.

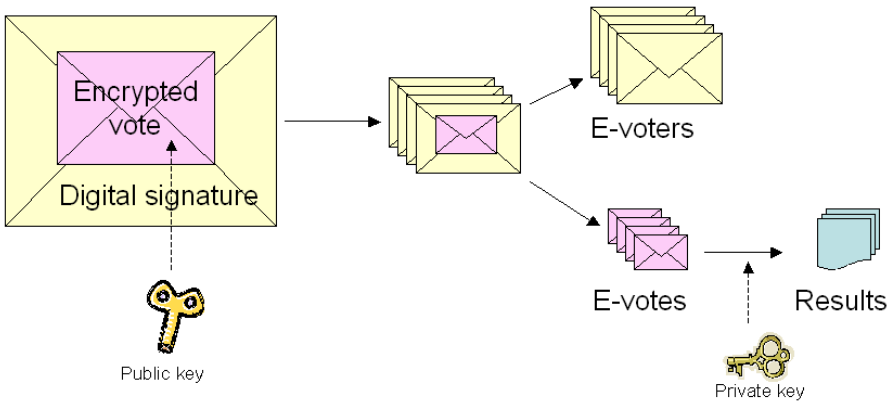

Fig. 4. The envelope method.

The e-voting system architecture consists of several building blocks illustrated in the Figure 5. The voter uses a Voter Application, which is downloaded from Vote Forwarding Server, to do all necessary selection, encryption, signing and sending activities. The received votes are immediately forwarded to Vote Storing Server which is inaccessible from outer world. Counting of votes happens off-line with Vote Counting Application and Hardware Security Module involved. All central components have extensive logging mechanisms in place -

every transaction in the system shall leave a track. The key management requires extra attention here as the security and anonymity of e-votes lie on encryption and decryption of votes.

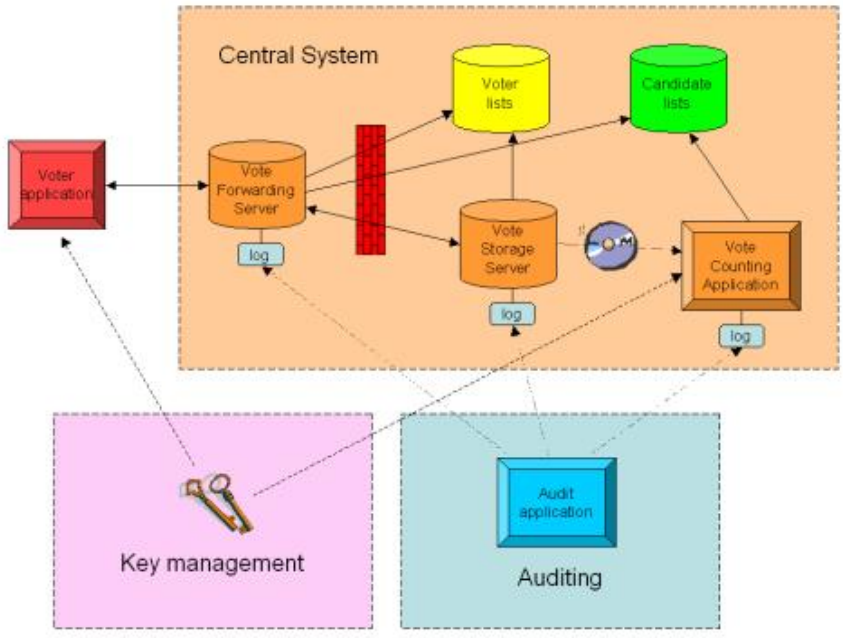

Fig. 5. Estonia e-vote design.

- *Voter* – e-voter with his PC. Creates an encrypted and digitally signed vote and sends it to the Central System;
- *Central System* – System component that is under the responsibility of the National Election Committee. Receives and processes the votes until the composite results of e-voting are output;
- *Key Management* – Generates and manages the key pair(s) of the system. The public key (keys) are integrated into Voter's applications, private key(s) are delivered to Vote Counting Application;
- *Auditing* – solves disputes and complaints, using logged information from the Central System.

The Central System has 3 components:

- Vote Forwarding Server (VFS) – authenticates the voter using his of ID-card, displays the candidates of voter's constituency to the voter and receives the

encrypted and digitally signed e-vote. The e-vote is immediately sent to the Vote Storage Server and the confirmation received from there is then forwarded to the voter. Completes its work after the close of advance polls;

- Vote Storage Server (VSS) – receives e-votes from the VFS and stores them. After the close of advance polls, removes double votes, cancels the votes by ineligible voters, and receives and processes e-vote cancellations. Finally it separates inner envelopes from outer envelopes and readies them for the Vote Counting Application;
- Vote Counting Application (VCA) – offline component to which crypted votes are transmitted with the digital signatures removed. The Vote Counting Server uses the private key of the system, tabulates the votes and outputs the results of e-voting.

User's perspective To vote electronically voter needs an ID-card and a computer, which is connected to the Internet and has an installed card-reader. From the user's perspective, the voting procedure looked like this:

1. The voter inserted the ID-card into card reader and opened the web page for voting (`http://www.valimised.ee/`);
2. The voter authenticated him using the PIN1 of the ID-card;
3. The server checked whether the voter was eligible (using the data from population register);
4. Candidate list of the appropriate electoral district was displayed;
5. The voter made his voting decision. The system encrypted it;
6. The voter confirmed his choice with a digital signature (by entering the PIN2-code);
7. The system confirmed retrieval of the vote.

After the electronic voting and advance polls ended (4th day before Election Day) the list of voters who had voted electronically was comprised and sent to polling stations in order to prevent voters from voting more than once on Election Day. After the polls are closed the members of the National Electoral Committee could collegially open the anonymous e-votes and count them.

Problems implementing e-voting There are many aspects of elections besides technical security problems that may bring e-voting into question.

E-voting brings along many concerns of fraud and privacy associated with remote balloting, including the risk that voters who do not cast their votes in the privacy of a voting booth, may be subject to coercion, or that voters have the opportunity to easily sell their vote. During the last elections in Estonia some vote-buying incidents became public and the problem has been blown up

in mass media. This is partly the reason why the e-voting concept suggests that the re-voting should be allowed. The fact that voter has always a possibility to re-vote, even in the controlled area on elections day, can minimise the number of manipulative attempts.

The legislative basis to conduct e-voting has been created but according to e-voting concept evolved during the last year, the election laws should be amended in some crucial points like allowing to re-vote electronically. Also the priority of traditional voting should be enacted. It is indispensable to convince politicians that the e-voting system can still guarantee that there is only one vote per voter in the ballot box.

There are still many concerns about the confidentiality of electronic voting and fears that a vote can be related to voter. An information campaign could be one of the measures to make the details of e-voting security, including the role of cryptology in it, publicly acquainted. Building public trust is one of the most difficult aspects of introducing the e-voting. The proposed e-voting methods need public acceptance otherwise legitimacy of e-voting can be placed in doubt.

3 Cryptographic Schemes used in Electronic Voting

Cryptographic voting schemes can be divided into three categories [18], based on the technique used to anonymize votes:

- *Homomorphic encryption*
- *Blind Signatures*
- *Mix Networks*

3.1 Homomorphic Encryption

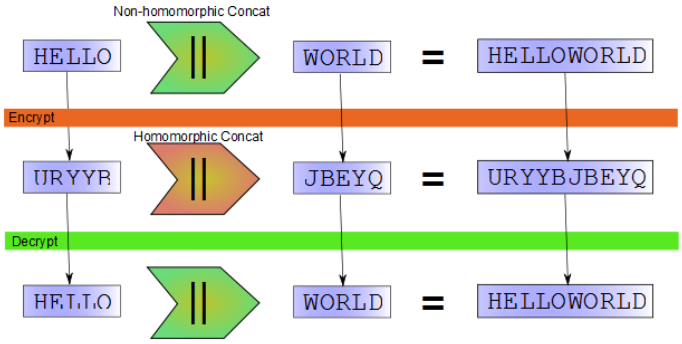

Fig. 6. Homomorphic Encryption.

Homomorphic is an adjective that describes a property of an encryption scheme. That property is the ability to perform computations on the ciphertext without decrypting it first.

Homomorphic encryption is a form of encryption where a specific algebraic operation is performed on the plaintext and another (possibly different) algebraic operation is performed on the ciphertext. Depending on one's viewpoint, this can be seen as either a positive or negative attribute of the cryptosystem. Homomorphic encryption schemes are malleable by design. The homomorphic property of various cryptosystems can be used to create secure voting systems, collision-resistant hash functions, and private information retrieval schemes. Figure 6 shows an example of how it works.

In schemes based on homomorphic encryption, voters submit encrypted votes that are never decrypted. Rather, the submitted ciphertexts are combined (using some operation that commutes with encryption) to produce a single ciphertext containing the election tally, which is then decrypted.

3.2 Blind Signature

In cryptography, a blind signature, as introduced by David Chaum, is a form of digital signature in which the content of a message is disguised (blinded)

before it is signed. The resulting blind signature can be publicly verified against the original, unblinded message in the manner of a regular digital signature. Blind signatures are typically employed in privacy-related protocols where the signer and message author are different parties. Examples include cryptographic election systems and digital cash schemes.

An often-used analogy to the cryptographic blind signature is the physical act of enclosing a message in a special write-through-capable envelope, which is then sealed and signed by a signing agent. Thus, the signer does not view the message content, but a third party can later verify the signature and know that the signature is valid within the limitations of the underlying signature scheme.

Blind signatures can also be used to provide *unlinkability*, which prevents the signer from linking the blinded message it signs to a later unblinded version that it may be called upon to verify. In this case, the signer's response is first *unblinded* prior to verification in such a way that the signature remains valid for the unblinded message. This can be useful in schemes where anonymity is required.

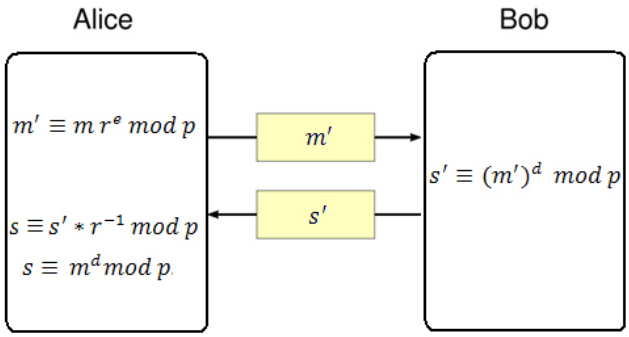

Fig. 7. Blind signature using RSA.

Blind signature schemes can be implemented using a number of common public key signing schemes, for instance RSA[4] and DSA[5]. To perform such a signature, the message is first *blinded*, typically by combining it in some way with a random *blinding factor*. The blinded message is passed to a signer, who then signs it using a standard signing algorithm. The resulting message, along with the blinding factor, can be later verified against the signer's public key. In some blind signature schemes, such as RSA, it is even possible to remove the blinding factor from the signature before it is verified. In these schemes, the final

[4] *RSA* stands for Rivest, Shamir and Adleman who first publicly described it, is an algorithm for public-key cryptography. See Figure 7.

[5] *DSA* stands for Digital Signature Algorithm, which is a United States Federal Government standard for digital signatures.

output (message/signature) of the blind signature scheme is identical to that of the normal signing protocol.

Blind signature schemes split the election authority into an authenticator and tallier. The voter authenticates to the authenticator, presents a blinded vote, and obtains the authenticator's signature on the blinded vote. The voter *unblinds* the signed vote and submits it via an anonymized channel to the tallier.

3.3 Mix Network

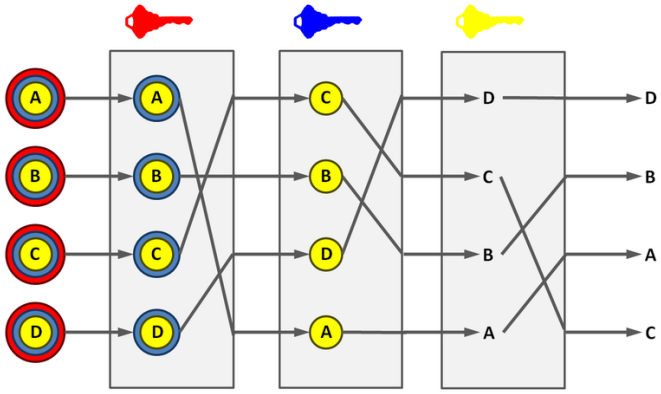

Fig. 8. Basic decryption mix net.

Mix networks, also known as Digital mixes, were invented by David Chaum in the early 1980's [17]. Digital mixes create hard-to-trace communications by using a chain of proxy servers. Each message is encrypted to each proxy using public key cryptography; the resulting encryption is layered like a Russian doll (except that each *doll* is of the same size) with the message as the innermost layer. Each proxy server strips off its own layer of encryption to reveal where to send the message next. If all but one of the proxy servers are compromised by the tracer, untraceability can still be achieved against some weaker adversaries. Figure 8 describes in a simple way, how mix network works[6].

In mix network schemes, voters authenticate and submit encrypted votes. Votes are anonymized using a mix, and anonymized votes are then decrypted. JCJ [40] and Civitas [18] are both based on mix networks.

[6] Simple decryption mix net: Messages are encrypted under a sequence of public keys. Each mix node removes a layer of encryption using its own private key. The node shuffles the message order, and transmits the result to the next node.

4 End-to-End Voter Verifiable Systems

End-to-end auditable or end-to-end voter verifiable (E2E) systems are voting systems with stringent integrity properties and strong tamper-resistance. E2E systems often employ cryptographic methods to create receipts that allow voters to verify that their votes were not modified, without revealing which candidates were voted for. As such, these systems are sometimes referred to as receipt-based systems. Some examples of E2E systems will be presented.

4.1 Prêt à Voter

Prêt à Voter [58] is an E2E voting system devised by Peter Ryan of Newcastle University. It aims to provide guarantees of accuracy of the count and ballot privacy that are independent of software, hardware etc. Assurance of accuracy flows from maximal transparency of the process, consistent with maintaining ballot privacy. In particular, Prêt à Voter enables voters to confirm that their vote is accurately included in the count whilst avoiding dangers of coercion or vote buying.

The key idea behind the Prêt à Voter approach is to encode the vote using a randomized candidate list. The randomisation of the candidate list on each ballot form ensures the secrecy of each vote. Incidentally, it also removes any bias towards the top candidate that can occur with a fixed ordering.

The value printed on the bottom of the receipt is the key to extraction of the vote. Buried cryptographically in this value is the information needed to reconstruct the candidate order and so extract the vote encoded in the receipt. This information is encrypted with secret keys shared across a number of tellers. Thus, only the set of tellers acting together is able to interpret the vote encoded in the receipt.

After the election, voters (or perhaps proxies acting on their behalf) can visit the Web Bulletin Board (WBB) and confirm their receipts appear correctly. Once this is over, the tellers take over and perform anonymising mixes and decryption of the receipts. All the intermediate stages of this process are posted to the WBB and are audited later.

There are various auditing mechanisms to ensure that all the steps, the creation of the ballot forms, the mixing and decryption are all performed correctly but these are carefully designed so as not to impinge on ballot privacy.

4.2 Helios: Web-based Open-Audit Voting

Helios [2], is a web-based open-audit voting system. Using a modern web browser, anyone can set up an election, invite voters to cast a secret ballot, compute a tally, and generate a validity proof for the entire process. Helios is deliberately simpler than most complete cryptographic voting protocols in order to focus on the central property of public auditability: any group can outsource its election to Helios, yet, even if Helios is fully corrupt, the integrity of the election can be verified.

Voting online or by mail is typically insecure in high-stakes elections because of the coercion risk: a voter can be unduly influenced by an attacker looking over her shoulder. Some protocols, as JCJ [40] and Civitas [18], attempt to reduce the risk of coercion by letting voters override their coerced vote at a later (or earlier) time. In these schemes, the privacy problem is shifted from vote casting to voter registration. In other words, no matter what, some really private interaction is required for coercion resistance.

Low-coercion elections With Helios, Ben Adida [2,3] does not attempt to solve the coercion problem. Rather, he suggests that student government, local clubs, on-line groups such as open-source software communities, and others do not suffer from nearly the same coercion risk as high-stakes government elections. Yet these groups still need voter secrecy and trustworthy election results, properties they cannot currently achieve short of an in-person, physically observable and well orchestrated election, which is often not a possibility. Helios was produced for exactly these groups with low-coercion elections.

Helios takes an interesting approach: there is only one trustee, the Helios server itself. Privacy is guaranteed only if you trust Helios. Integrity, of course, does not depend on trusting Helios: the election results can be fully audited even if all administrators – in this case the single Helios sever – is corrupt.

Components Helios [2,3] is divided into 4 major components:

- *election builder*: a web-based tool to create an election;
- *voting booth*: a web-based voting booth where ballots are filled out and encrypted;
- *ballot casting server*: the server where filled-out ballots are submitted;
- *audit server*: the place where all data is posted at the end of an election.

Some details Helios is a web-based open-audit voting system [3]. Votes are encrypted, using the *browser-based ballot encryption program*, with El-Gamal encryption of a plaintext representation of the choices. Once loaded, this ballot encryption single-page web application does not access the network again until the vote is encrypted and ready for casting. Encryption is specifically achieved within the web browser using LiveConnect to access the Java Virtual Machine from *JavaScript*, enabling vote encryption in 3 or 4 seconds on a typical configuration. Ballot casting assurance is achieved using the Benaloh cast-or-audit voting protocol [11] implemented in part by the ballot verifier which ensures that an audited ballot indeed corresponds to the fingerprint generated before the cast-or-audit choice. A Sako-Kilian [60] / Benaloh [10] mixnet provably shuffles the votes, then the election server decrypts the shuffled votes and tallies the results. An election verification program downloads all encrypted votes, shuffled votes, decrypted ballots and proofs, and verifies that the election was run correctly.

Fig. 9. Helios' voting interface.

Homomorphic Tallying Helios [3] shifts from mixnet-based to homomorphic tallying [19], because homomorphic tallying is easier to implement efficiently, and thus easier to verify, especially when a third party writes verification code. Ben Adida use *Exponential El-Gamal*, a variant of El-Gamal [29] where one encrypts g^m rather than m in order to achieve an additive homomorphism, because it is easier to implement than alternative additively homomorphic schemes such as Paillier [55]. (Decryption requires a discrete logarithm computation, though with a relatively small exponent that ensures that the computation is, in fact, quite tractable.)

In addition, El-Gamal lends itself quite easily to distributed decryption with joint key generation, where other additively homomorphic systems like Paillier are significantly more complicated. For simplicity and ease of discrete-logarithm computation, Clarkson et al. used a single ciphertext for each option of each election question, rather than attempt multi-answer encoding with the more involved proofs of correct ballot form [7]. A ballot is then composed of:

- a ciphertext for each available answer to each election question,
- a disjunctive zero-knowledge proof [20] that each such ciphertext encodes either a 0 or a 1, and
- a disjunctive zero-knowledge proof that the homomorphic sum of all ciphertexts for a given question is the encryption of one out of *0, 1, ... , max* for a pre-set maximum (ensuring that between *0* and *max* answers are selected for each question.)

In order to achieve high security with efficient modular exponentiation, all operations were performed in a subgroup of order q of $\mathbb{Z}_p^*$ where p and q are 2048 and 256 bit long primes, respectively.

Distributed Decryption Another significant feature of Helios is distributed decryption to ensure that multiple trustees are required for decryption. This ensures that only the homomorphic tally of all votes is decrypted, never an individual ballot. Two options were available:

- having each trustee generate a typical El-Gamal public key using the same (p, q, g) parameters, and combining the public keys using simple multiplication, or
- having each trustee to generate a typical El-Gamal public key using the same (p, q, g) parameters, then have each trustee generate and publish a Lagrange coefficient to enable threshold decryption [30,56].

Because the second approach is a little bit tricky to implement securely, with an additional step in the interaction between trustees to generate the Lagrange coefficients, Clarkson et al. opted for the slightly less robust, but just as secure, first option. Ben Adida noted that robust key generation, though crucial, is not entirely novel in this space: at least one other voting system, ADDER [41], already implements distributed key generation.

4.3 Civitas

Civitas protocol was recently proposed by Clarkson et al. [18], and is based on previous voting protocol by A. Juels et al. (JCJ) [40]. The Civitas electronic voting scheme promises the possibility of a convenient, efficient, and secure facility for recording and tallying votes during a remote election and has been developed in order to fit the most ambitious security requirements identified so far for remote voting. It is the first implementation of a voting system that allows voters to conveniently vote from a remote client of their choice.

Entities In Civitas, there exist five different groups of entities or agents: a supervisor, a registrar, voters, registration talliers, and tabulation talliers. Four of them are together with the ballot boxes and the bulletin board depicted in Figure 10. The responsibilities of these entities are described below whereas the functionality of the ballot boxes and the bulletin board is described later.

- *Supervisor* The supervisor is the administrator of an election. He is responsible for selecting the participating registration and tabulation talliers, for setting up the ballot design and for starting and stopping the election;
- *Registrar* The registrar solely authorizes voters;

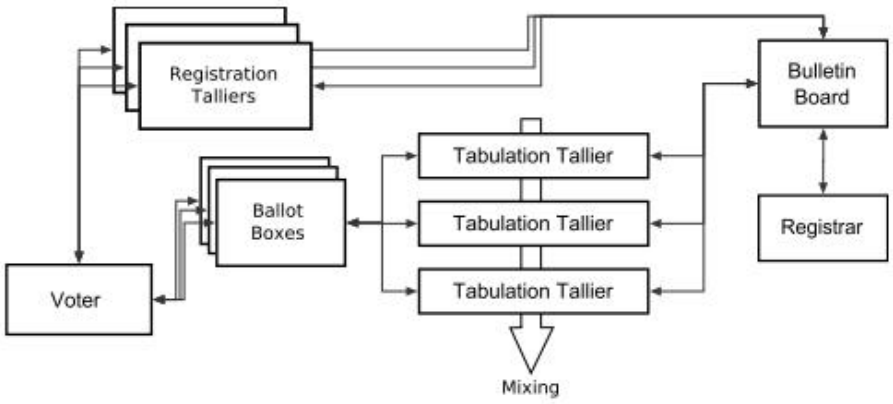

Fig. 10. Civitas architecture.

- *Voters* Voters register for voting, get their credentials and cast a vote;
- *Registration Talliers* The registration talliers together generate the credentials that are used by the voters for voting;
- *Tabulation Talliers* The tabulation talliers are responsible for tallying the votes while preserving the anonymity of voters.

Phases Not all of these entities participate in every of the four protocol phases: setup, registration, voting and tallying.

1. *Setup Phase* During the setup phase, the supervisor first sends the ballot design and the public keys of the registration and tabulation talliers to the bulletin board. By this he starts the election. Afterwards, the registrar posts the electoral roll which includes for every authorized voter an unique identifier, his public registration key, and his public designation key. The designation key is needed later on for the Designated Verifier Reencryption Proof (DVRP). Then the tabulations talliers jointly generate a key for the distributed encryption scheme (decryption needs participation of every single tallier) and post it on the bulletin board.
2. *Registration Phase* After the election is set up, each voter requests his private credentials by sending his public registration key to the tabulations talliers. It needs at least one secure channel between the voter and the registration teller.
3. *Voting Phase* After the voter has successfully constructed his private credential, he casts a vote by sending his encrypted private credential, the encrypted vote, and the zero-knowledge proof to one or more *Ballot Boxes*. Both the *Bulletin Board* and the *Ballot Boxes* have restricted access rights, so that data stored once can only be read but not modified or deleted.

4. *Tallying Phase* After all voters have casted their votes, the tabulation talliers collect the votes from the ballot boxes. A voter needs to send a vote to at least one valid ballot box to get it definitely counted. Together, the tabulation talliers run a mix network [17] to anonymize the votes in which each tabulation tallier performs two permutations. Plaintext equivalence tests (PETs) [35] are used to eliminate duplicates and invalid credentials. These are used to compare ciphertexts. Given c and c', a PET reveals whether $Dec(c) = Dec(c')$, but nothing more about the plaintexts of c and c'.

4.4 Which one to choose?

Prêt à Voter [58] offers a weak form of coercion resistance, if voting is supervised. The construction of ballots depends on non-uniformly distributed seeds, which might enable the adversary to learn information about how voters voted. In remote settings, Prêt à Voter offers no coercion resistance. The adversary, by observing the voter during voting, will learn what vote was cast.

Helios [2] was made for online software communities, local clubs, student government, and other environments where trustworthy, secret ballot elections are required but coercion is not a serious concern. So this system can be taken into account for our solution.

This leaves Civitas, which is the only system that offers stronger coercion resistance than other implemented voting systems. Hence a coercion-resistant voting system is one in which the user can deceive the adversary into thinking that she has behaved as instructed, when the voter has in fact cast a vote according to her own intentions.

5 Civitas, Towards a Secure Voting System?

In this section, Civitas [18] will be explained far more in detail, specially the features which make it that interesting and how it works. Civitas is based on a voting protocol by A. Juels et al. (JCJ) [40] and adapts most of its general functionalities. Many features are extended and one of the main disadvantages of JCJ, the scalability to big elections is solved. In JCJ, the *Plaintext Equivalence Tests* (PETs) are the reason for the lack of scalability. Civitas solves this by clustering the election into small blocks which dramatically decreases the number of PETs for a sufficiently small block size.

5.1 Design

Civitas refines and implements a voting scheme, which is referred to it as JCJ because it was developed by Juels, Catalano, and Jakobsson [40].

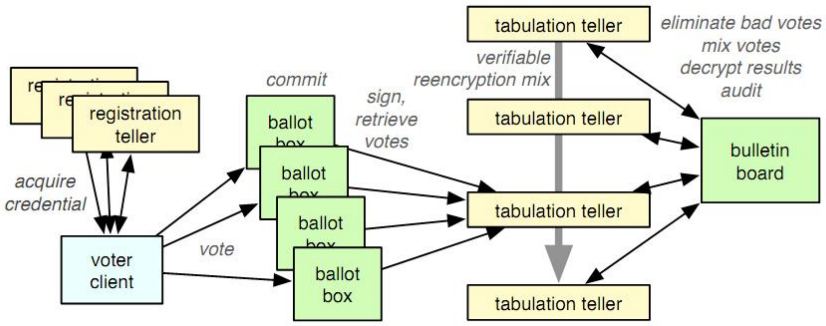

Fig. 11. Civitas architecture.

Agents There are five kinds of agents in the Civitas voting scheme: a *supervisor*, a *registrar*, *voters*, *registration tellers*, and *tabulation tellers*. Some of these are depicted in Figure 11. The agents other than voters are election authorities:

- *The supervisor* administers an election. This includes specifying the ballot design and the tellers, and starting and stopping the election;
- *The registrar* authorizes voters;
- *Registration tellers* generate the credentials that voters use to cast their votes;
- *Tabulation tellers* tally votes.

These agents use an underlying *log* service that implements publicly readable, insert-only storage. Integrity of messages in a log is ensured by digital signatures. Agents may sign messages they insert, ensuring that the log service cannot forge new messages. The log service must sign its responses to reads, ensuring that attempts to present different views of log contents to different readers can be detected. Multiple instances of the log service are used in a single election. One instance, called the *bulletin board*, is used by election authorities to record all the information needed for verifiability of the election. The remaining instances, called *ballot boxes*, are used by voters to cast their votes.

Setup phase First, the supervisor creates the election by posting the ballot design on an empty bulletin board. The supervisor also identifies the tellers by posting their individual public keys. Second, the registrar posts the *electoral roll*, containing identifiers (names or registration numbers) for all authorized voters, along with the voters' public keys. Each voter is assumed to have two keys, a *registration* key and a *designation* key, whose uses are described below. Third, the tabulation tellers collectively generate a public key for a distributed encryption scheme and post it on the bulletin board. Decryption of messages encrypted under this key requires the participation of all tabulation tellers. Finally, the registration tellers generate *credentials*, which are used to authenticate votes anonymously. Each credential is associated with a single voter. Like keys in an asymmetric cryptosystem, credentials are pairs of a public value and a private value. All public credentials are posted on the bulletin board, and each registration teller stores a share of each private credential. Private credentials can be forged or leaked only if all registration tellers collude.

Voting phase Voters register to acquire their private credentials. Each registration teller authenticates a voter using the voter's registration key. The teller and voter then run a protocol, using the voter's designation key, that releases the teller's share of the voter's private credential to the voter. The voter combines all of these shares to construct a private credential. Voting may take place immediately, or a long time after registration. To vote, the voter submits a private credential and a *choice* of a candidate (both encrypted), along with a proof that the vote is well-formed, to some or all of the ballot boxes. (This submission does not require either of the voter's keys.) Replication of the vote across the ballot boxes is used to guarantee availability of the vote for tabulation:

– *Resisting coercion* – The key idea [40] that enables voters to resist coercion, and defeats vote selling, is that voters can substitute fake credentials for their real credentials, then behave however the adversary demands:
  - If the adversary demands the voter *to submit a particular vote*, then the voter *does so with a fake credential*;
  - If the adversary demands the voter *to sell or surrender a credential*, then the voter *supplies a fake credential*;

- If the adversary demands the voter *to abstain*, then the voter *supplies a fake credential to the adversary and votes with a real one.*

To construct a fake credential, the voter locally runs an algorithm to produce fake private credential shares that, to an adversary, are indistinguishable from real shares. The faking algorithm requires the voter's private designation key. The voter combines these shares to produce a fake private credential; the voter's public credential remains unchanged;

– *Revoting* – Voters might submit more than one vote per credential. The supervisor has the flexibility to specify a policy on how to tally such revotes. If revotes are not allowed, then all votes submitted under duplicate credentials are eliminated. If revotes are allowed, then the voter must include a proof in later votes to indicate which earlier votes are being replaced. This proof must demonstrate knowledge of the credential and choice used in both votes, preventing an adversary from revoting on behalf of a voter;

– *Ballot design* – Civitas is compatible with the use of any ballot design for which a proof of well-formedness is possible. Our prototype supports the use of ballots in which voters may choose a single candidate (plurality voting), any subset of candidates (approval voting), or a ranking of the candidates (ranked voting). However, ranked voting introduces covert channels that enable attacks on coercion resistance [18].

Tabulation phase The tabulation tellers collectively tally the election:

1. *Retrieve data* – All tabulation tellers retrieve the votes from each ballot box and the public credentials from the bulletin board;

2. *Verify proofs* – The tellers check each vote to verify the proof of well-formedness. Any vote with an invalid proof is discarded;

3. *Eliminate duplicates* – At most one vote is retained for each credential. Votes with duplicate credentials are eliminated according to the revoting policy;

4. *Anonymize* – Both the list of submitted votes and the list of authorized credentials are anonymized by applying a random permutation, implemented with a *mix network* [17]. In the mix, each tabulation teller in turn applies its own random permutation;

5. *Eliminate unauthorized votes* – The credentials in the anonymized votes are compared against the anonymized authorized credentials. Any votes with invalid credentials are discarded;

6. *Decrypt* – The remaining choices, but not credentials, are decrypted. The final tally is publicly computable.

5.2 Verifying an election

Tabulation is made publicly verifiable by requiring each tabulation teller to post proofs that it is honestly following the protocols. All tabulation tellers verify these proofs as tabulation proceeds. An honest teller refuses to continue when it discovers an invalid proof. Anyone can verify these proofs during and after tabulation, yielding universal verifiability. A voter can also verify that his vote is present in the set retrieved by the tabulation tellers, yielding voter verifiability.

5.3 Security

The Civitas voting system requires certain assumptions about the trustworthiness of agents and system components [18]. Attacks are possible when these trust assumptions are violated:

Assumption 1 *The adversary cannot simulate a voter during registration.*

There must be some period of time during which the adversary cannot simulate the voter. Otherwise the system could never distinguish the adversary from the voter, so the adversary could register and vote on behalf of a voter. Registration is a good time for this assumption because it requires authentication and can be done far in advance of the election. During registration, Civitas authenticates voters with their registration keys. So this assumption restricts the adversary from acquiring a voter's key before the voter has registered. However, voters might attempt to sell their private registration keys, or an adversary might coerce a voter into revealing the voter's key. Both attacks violate the assumption by allowing the adversary to simulate a voter.

Assumption 2 *Each voter trusts at least one registration teller, and the channel from the voter to the voter's trusted registration teller is untappable.*

Constructing a fake credential requires the voter to modify at least one of the shares received during registration. Suppose the adversary can tap all channels to registration tellers and record the encrypted traffic between the voter and the registration tellers. Further suppose that the adversary can corrupt the voter's client so that it records all credential shares received from tellers. Then the adversary can ask the client to reveal the plaintext credential shares corresponding to the encrypted network messages. In this scenario, the voter cannot lie to the adversary about his credential shares, meaning that the voter could now sell his credential and is no longer protected from coercion. So an untappable channel is required for distribution of at least one share. The voter must also trust the teller who issued that share not to reveal it.

Assumption 3 *Voters trust their voting client.*

Voters enter votes directly into their clients. No mechanism ensures that the client will preserve the integrity or the confidentiality of votes. A corrupt voting

client could violate coercion resistance by sending the plaintext of the voter's credential and choice to the adversary. A corrupt client could also violate verifiability by modifying the voter's credential or choice before encrypting it. Clients could be corrupted in many ways. The machine, including the network connection, could be controlled by the adversary. Any level of the software stack, from the operating system to the client application, could contain vulnerabilities or be corrupted by malicious code. The adversary might even be an insider, compromising clients during their development and distribution.

Assumption 4 *The channels on which voters cast their votes are anonymous.*

Without this assumption, the adversary could observe network traffic and learn which voters have voted, trivially violating coercion resistance – although the adversary still could not learn the voter's choice or credential.

Assumption 5 *At least one of each type of authority is honest.*

At least one of the ballot boxes to which a voter submits his vote is correct. A correct ballot box returns all the votes that it accepted to all the tabulation tellers. There exists at least one honest tabulation teller. If all the tellers were corrupted, then the adversary could trivially violate coercion resistance by decrypting credentials and votes.

Assumption 6 *Cryptography works.*

DDH and RSA are standard cryptographic assumptions. The more fundamental assumption for Civitas is DDH, as the JCJ security proof is a reduction from it [9].

5.4 How Civitas works

An election is created by naming a set of registrars and talliers. The protocol is divided into four phases: setup, registration, voting and tallying. Details about the steps of the protocol will be presented now, starting with the setup phase:

1. The registrars (resp. talliers) run a protocol which constructs a public key pair and distributes a share of the secret part amongst the registrars' (resp. talliers'). The public part $\mathrm{pk}(sk_T)$ (resp. $\mathrm{pk}(sk_R)$) of the key is published. The registrars also construct a distributed signing key pair ssk_R, $\mathrm{pk}(ssk_R)$.

   The registration phase then proceeds as follows.

2. The registrars generate and distribute voter credentials: a private part d and a public part $\mathrm{penc}(\mathrm{pk}(sk_R), m", d)$ (the probabilistic encryption of d under the registrars' public key $\mathrm{pk}(sk_R)$). This is done in a distributed manner, so that no individual registrar learns the value of any private credential d.

3. The registrars publish the signed public voter credentials.

4. The registrars announce the candidate list $\tilde{t} = (t_1, \ldots, t_l)$.

   The protocol then enters the voting phase.

5. Each voter selects her vote $s \in \tilde{t}$ and computes two ciphertexts M = penc(pk(sk_T) , m , s) and M' = penc(pk(sk_R),m',d) where m, m' are nonces. M contains her vote and M' her credential. In addition, the voter constructs a non-interactive zero-knowledge proof of knowledge demonstrating the correct construction of her ciphertexts and validity of the candidate ($s \in \tilde{t}$). (The ZKP provides protection against coercion resistance, by preventing forced abstention attacks via a *write in*, and binds the two ciphertexts for eligibility verifiability.) The voter derives her ballot as the triple consisting of her ciphertexts and zero-knowledge proof and posts it to the bulletin board.

   Designated-Verifier Zero-knowledge Proofs

   The zero-knowledge proofs from [6] are non-interactive and therefore transmissible (they can be forwarded to convince other participants of the truth of the statement). However, this is not always desirable. During the registration phase of Civitas, the voter acquires a private credential for voting. For this he contacts each of the registration talliers and asks them for a share of the private credential. In order for a registration tallier to prove the correctness of the share s_i, a designated-verifier zero-knowledge proof is used [34]. This proof should be able to convince only the voter of the correctness of the share s_i, and nobody else. In particular the voter should be able to generate a fake proof of this fact, e.g. using his secret signing key.

6. After some predefined deadline the tallying phase commences.

7. The talliers read the n' ballots posted to the bulletin board by voters (that is, the triples consisting of the two ciphertexts and the zero-knowledge proof) and discards any entries for which the zero-knowledge proof does not hold.

8. The elimination of re-votes is performed on the ballots using pairwise *plaintext equality tests*[7] (PET) [35] on the ciphertexts containing private voter credentials. Re-vote elimination is performed in a verifiable manner with respect to some publicly defined policy, *e.g.*, by the order of ballots on the bulletin board.

9. The talliers perform a verifiable re-encryption mix on the ballots (ballots consist of a vote ciphertext and a public credential ciphertext; the link between both is preserved by the mix.) The mix ensures that a voter cannot trace her vote, allowing the protocol to achieve coercion-resistance.

10. The talliers perform a verifiable re-encryption mix on the list of public credentials published by the registrars. This mix anonymises public voter credentials, breaking any link with the voter for privacy purposes.

[7] A PET is a cryptographic predicate which allows a keyholder to provide a proof that two ciphertexts contain the same plaintext.

11. Ballots based on invalid credentials are weeded using PETs between the mixed ballots and the mixed public credentials. Both have been posted to the bulletin board. (Using PETs the correctness of weeding is verifiable.)
12. Finally, the talliers perform a verifiable decryption and publish the result.

6 Where Do We Stand?

After all previous sections, this comes to a stand where it is needed some reflection about all previously seen. This work started by analysing the paper ballot, a voting method that we are attempting to replace for an electronic voting system. One great reason to try this change is the possibility to vote remotely, with all the privacy and security that any elections need.

Image the possibility of being far away of our standard voting area and be able to vote on our holidays, in a foreign country or simply in our home. That is just one important point. In fact, there is also the real question behind the knowing that our vote is really tallied.

As well seen and detailed, the security properties studied in the last few years by many researchers around the world, it was realised that in electronic voting, there are many concerns about situations that do not happen in the traditional paper ballot voting system. The properties we mentioned before were this:

- *Eligibility* : only legitimate voters can vote;
- *Fairness* : no early results can be obtained which could influence the remaining voters;
- *Individual Verifiability* : a voter can verify that her vote was really counted;
- *Universal Verifiability* : the published outcome really is the sum of all the votes;
- *Vote-Privacy*: the fact that a particular voter voted in a particular way is not revealed to anyone;
- *Receipt-Freeness* : a voter does not gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way;
- *Coercion-Resistance* : a voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

From this seven properties, the latest one, *coercion-resistance*, is the most difficult to achieve because it gives a great liberty to an adversary[8]. It is needed to start from a simple principle: the harder the adversary is, the best the voting system must be. In other words, the most an adversary do without compromising this seven principles of a voting system, the better this system is.

So, another goal to our system is to give as much freedom as possible to our adversary. It is kind of awkward goal, but in the end will be more useful to the system. The voter must also be convinced and sure that this vote is correctly captured. It is needed both individual and universal verifiability.

The analysis of some electronic voting system in use, gave us a clear picture of the status of the acceptance of such systems. One of the first countries to use

[8] Adversary or coercer can be used here. It is designated by adversary anyone that wants to do some warm either to the election, either to the voter's choice itself.

electronic voting was Brazil. And due to the fact that their system has wrongly designed, led to big issues and doubts surrounding electronic voting itself. Too many suspicious of fraud and an awkward voting machine (Diebold's DRE), that had just to many design faults, drove electronic voting to a state of disbelieve in the mind of the voters, that was proven that the use of DRE do not solve the issues we have mention before.

In Brazil's case, everything is wrong. There is only one trust entity that is in charge for everything surrounding the elections. With such situation, it is easy to understand that no real control and verifiability over that entity is made. There is just to many powers concentrated in one entity. Lets not forget the purpose too. Electronic voting must easy all the process, maintaining the same levels of security and privacy a paper ballot has.

Estonia's case is completely different from Brazil's. They manage to go right into the future, and built a system that is the first remote voting system of the world. It is based on the ability of voting by the Internet, using the citizen's identification card as credentials. All sounds and looks great, if it was not the fact that the system does not comply with some of the properties needed for electronic voting.

To be able to be coercion-resistance, designers of the system allowed the voter to revote as many times he or she wants, from the 6^{th} until the 4^{th} day before the election day. This is a great feature, although quite weak in terms of coercion resistance, but one of the problems happens next, then it is to *replace* the electronic vote by the paper ballot. This is a great threat to the vote privacy, because in some point of the voting process, someone will know, at least, if someone has voted or not. This is not desired.

Then we saw different electronic voting protocol, like Prêt à Voter, Helios and Civitas. All of them have their strengths, but only Civitas has able to handle coercion-resistance in a proper way and also comply with the other six properties.

Civitas is the first electronic voting protocol that is coercion-resistant, universally and voter verifiable, and suitable for remote voting.

Civitas is based on a previously-known voting system, but elaborating the scheme into an implemented system led to new technical advances: a secure registration protocol and a scalable vote storage system. Civitas thus contributes to both the theory and practice of electronic voting. But as all, it is not perfect. It still have open issues, which make the opportunity for this work.

The first to do is to begin by analysing its weaknesses and try to solve some of the trust assumptions made by the authors.

6.1 Unsolved Issues

Some open technical problems must be solved before Civitas, or a system like it, could be used to secure national elections. Two such problems are that Civitas assumes a trusted voting client, and that in practice, the best way to satisfy two of the Civitas trust assumptions is in-person registration. Address availability is still not solved. However, the design of Civitas accommodates complementary

techniques for achieving availability. To improve the availability of election authorities, they could be implemented as Byzantine fault-tolerant services [39]. The encryption scheme used by Civitas could be generalized from the current distributed scheme to a threshold scheme[9]. This would enable election results to be computed even if some tabulation tellers become unresponsive or exhibit faulty behavior, such as posting invalid zero-knowledge proofs.

For a threshold scheme requiring k out of n tabulation tellers to participate in decryption, no more than k - *1* tellers may be corrupted, otherwise coercion resistance could be violated. For availability, a new trust assumption must be added: At least k tellers do not fail.

Credentials Management

Management of credentials is an interesting problem for the use of Civitas. Voters might find generating fake credentials, storing and distinguishing real and fake credentials (especially over a long term), and lying convincingly to an adversary to be quite difficult. Recovery of lost credentials is also an open problem.

Other issues

There are open non-technical problems as well, as we can see in this three examples:

1. Some people believe that any use of cryptography in a voting system makes the system too opaque for the general public to accept;
2. Remote electronic voting requires voters to have access to computers, but not all people have such access now;
3. Some real-world attacks, such as attempts to confuse or misinform voters about the dates, significance, and procedures of elections, are not characterized by formal security models. Mitigation of such attacks is important for real-world deployments, but beyond the scope of this paper.

Finally, a report on the security of a real-world remote voting system, SERVE, identifies a number of open problems in electronic voting [36]. These problems include transparency of voter clients, vulnerability of voter clients to malware, and vulnerability of the ballot boxes to denial-of-service attacks that could lead to large-scale or selective disenfranchisement. However, Civitas does address other problems raised by the report: the voter client is not a DRE, trust is distributed over a set of election authorities, voters can verify their votes are counted, spoofing of election authorities is not possible due to the use of digital signatures, vote buying is eliminated by coercion resistance, and election integrity is ensured by verifiability.

In remote voting, all the security properties described before are extremely important to assure. The less trust assumptions our architecture has, the better it will be suited to the real-world usage.

[9] See Appendix D.

6.2 Old Assumptions

Lets go back to the Civitas' trust assumptions and review them in order to find solutions for those which matter most.

Assumption 1: *The adversary cannot simulate a voter during registration.*

One possible defense would be to store private keys on tamper-resistant hardware, which could enforce digital non-transferability of the keys. This is not a completely effective defense, as voters could physically transfer the hardware to the adversary. Preventing such physical transfers is not generally possible, but they could be discouraged by introducing economic disincentives for voters who relinquish their keys. For example, the Estonian ID card, which contains private keys and is used for electronic voting, can be used to produce legally binding cryptographic signatures [48,49]. Voters would be unlikely to sell such cards, although coercion would remain a problem. Another possible defense is to change authentication to use in-person registration as an alternative to private keys.

Each registration teller would either be an online teller, meaning voters register with that teller remotely, or an offline teller, meaning voters must register in person with that teller. Offline registration tellers would be trusted to authenticate voters correctly, preventing the adversary from masquerading as the voter. At least one offline registration teller would need to exist in any election, ensuring that voters register in person with at least one teller. For deployments of Civitas in which this trust assumption does not hold, we recommend requiring in-person registration. This compromises of our goal of a fully remote system. But it is a practical defense, since voting could still be done remotely, registration could be done far in advance of the actual election, and a single credential could be reused for multiple elections.

Assumption 2: *Each voter trusts at least one registration teller, and the channel from the voter to the voter's trusted registration teller is untappable.*

An untappable channel is the weakest known assumption for a coercion-resistant voting system [7,21,34,40,60]. Replacing this with a more practical assumption has been an open problem for at least a decade [22]. Offline registration tellers, discussed with Assumption 1, could ensure an untappable channel by supervising the registration process.

Assumption 3: *Voters trust their voting client.*

Current research aims to solve this problem by changing how voters enter their votes [37]. The voting client is decomposed into multiple (hardware and software) components, and the voter interacts with each component to complete the voting process. For example, voting might require interacting with a smart card to obtain a randomized ballot, then interacting with a client to submit a vote on that ballot. Now the voter need not trust a single client, but instead that the components implementing the client will not collude. Complementary

research aims to leverage trusted computing technology [5]. For example, attestation could be used to prove that no level of the hardware or software stack has been changed from a trusted, pre-certified configuration.

Note that this trust assumption does not require all voters to trust a single client implementation. Rather, voters may choose which client they trust. This client could be obtained from an organization the voter trusts, such as their own political party or another social organization. These organizations are free to implement their own Civitas client software on their own hardware, and to make their source code publicly available. This freedom improves upon current direct recording electronic (DRE) voting systems, in which voters are often forced by local election authorities to use particular proprietary (or closed-source) clients that are known to contain vulnerabilities [42]. Another advantage over DREs is that diverse clients, provided by several organizations, could reduce the incentive to attack Civitas by raising the cost of mounting an attack. Requiring trusted voter clients compromises our goal of a remote voting system. Even if voters download a client from a trusted organization, the software stack on a voter's machine might not be trustworthy. Thus voters might need to travel to a location where an organization they trust has provided a client application running on a trustworthy hardware and software platform.

Assumption 4: *The channels on which voters cast their votes are anonymous.*

The current prototype of Civitas does not implement its own anonymous channel because authors thought that the construction of trustworthy anonymous channels was an orthogonal research problem. It seems likely that existing anonymizing networks, such as Tor[10] [25], would suffice if made sufficiently reliable.

Assumption 5: *At least one of each type of authority is honest.*

At least one of the ballot boxes to which the voter submits his vote is correct. This is weaker than the standard assumption (less than a third of the ballot boxes fail) made for Byzantine fault tolerance [14,46] and multi-party computation [31], which both require more expensive protocols.

There exists at least one honest tabulation teller. This assumption is not needed for verifiability, even if all the tellers collude or are corrupted – the proofs posted by tellers during tabulation will reveal any attempt to cheat. Fault tolerance techniques [27,46,61] would increase the difficulty of corrupting all the tellers.

Assumption 6: *Cryptography works.*

Cryptography must work, guaranteeing that encryptions are secret, signatures authenticate. The Decision Diffie-Hellman (DDH) and RSA assumptions hold, and SHA-256 implements a random oracle.

[10] See Appendix C for more information.

Attacks on election authorities

Assumptions 2 and 5 allow all but one election authority of each kind to be corrupted. But certain attacks might still be mounted:

- A corrupt registration teller might fail to issue a valid credential share to a voter. The voter can detect this, but coercion resistance requires that the voter cannot prove that a share is valid or invalid to a third party. Defending against this could involve the voter and another election authority, perhaps an external auditor, jointly attempting to re-register the voter. The auditor could then attest to the misbehavior of a registration teller;
- The bulletin board might attempt to alter messages. But this is detectable since messages are signed. A bulletin board might also delete messages. This is an attack on availability;
- A corrupt registrar might add fictitious voters or remove legitimate voters from the electoral roll. Each tabulation teller can defend against this by refusing to tabulate unless the electoral roll is correct according to some external policy;
- A corrupt supervisor might post an incorrect ballot design, stop an election early, or even attempt to simulate an election with only one real voter. Voters and tabulation tellers should cease to participate in the election once the supervisor exhibits such behavior.

All election authorities might be simultaneously corrupted if they all run the same software. For example, an insider working at the software supplier might hide malicious code in the tabulation teller software. As discussed in Assumption 5, this attack could violate coercion resistance, but it could not violate verifiability. To defend against insider attacks, election authorities should use diverse implementations of the Civitas protocols.

6.3 Goals

Now that we reviewed and analyse Civitas' assumptions, we are aware of its problems. The most important assumption to be solved is the one related to trust in voting client. As our goal is to create up a remote voting system, this assumption need to be removed, in order to make Civitas usable to our proposes.

7 Trusting in Voting Clients

This section will describe solutions for the trust in voting client problem and discuss its applicability to this work.

7.1 Unreliable Vote Client

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 it is presented an overview of the approaches proposed to mitigate the problem of the unreliable vote client platform [54].

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 it be claimed that a CD-ROM is clean? And to which extent?
- Is it clean enough to provide a secure voting platform?

With all these questions and problems pending, it is wise to think that this approach can not be successfully used to enable secure voting from any computer with an Internet connection.

Secure Hardware to connect to the 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. [66] 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.

Closed Secure Devices The use of closed secure devices was one of the proposals made by the California Internet Voting Task Force in 2000. 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 farming attacks.

Secure Operation 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. 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 [59] and concerning the revocation of cracked machines [12]. 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 [65].

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. 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. 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.

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. Kutylowski and Zagórski [45] 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 $\frac{1}{2}$ of 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. [62] 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.

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 were made in 2001 by Chaum [15], 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. A similar system was also proposed by Helbach and Schwenk [32] 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.

7.2 CodeVoting

CodeVoting was proposed as solution/system to prevent vote manipulation at client side while allowing the use of cryptographic voting protocols to protect the election's integrity at server side. CodeVoting is a mix of the *special security PC hardware* and *code sheet* approaches mentioned above. In Figure 12 it is presented an overview of the CodeVoting components.

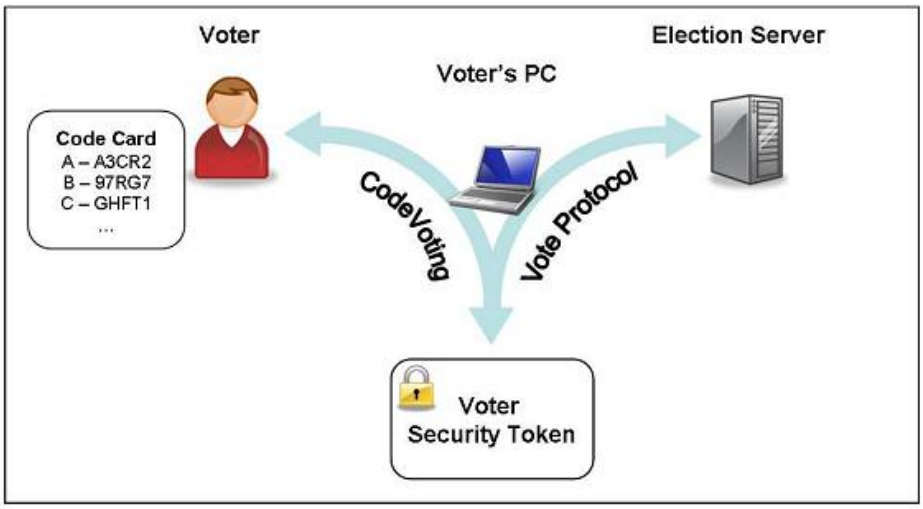

Fig. 12. CodeVoting components.

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, the CodeVoting consists in the following steps:

1. the voter expresses her/his vote as a secret code;
2. the secret code is translated into the corresponding candidate identifier (ID) (clear vote);
3. the clear vote is used in a cryptographic voting protocol (e.g. Civitas);
4. once a cryptographic proof of the correct vote delivery is received, a successful vote delivery code is released to the voter.

Voter Security Token The VST is the component in charge of the vote code translation to the clear vote as printed in the CodeCard. 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. The authors suggest the use of the VST to enable such authentication mechanism. The VST should be protected by a PIN[11] to prevent unauthorized access.

It was proposed 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.

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, there is a problem to solve: how to create the CodeCard, associate it with the VST and give it to the voter without leaking the codes.

It was proposed to generate each voter's CodeCard within the VST. This is a viable option because the CodeCard becomes automatically associated with the

[11] *PIN* stands for Personal Identification Number.

corresponding VST and no other entity besides the VST has access to the codes on the CodeCard. However, still have the problem of how to secretly print the CodeCard, i.e. how to give it to the voter without leaking the codes. Authors [38] believed 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. Authors 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.

Details CodeVoting can be seen as a rearrangement of the ideas presented by Chaum [15]. 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.

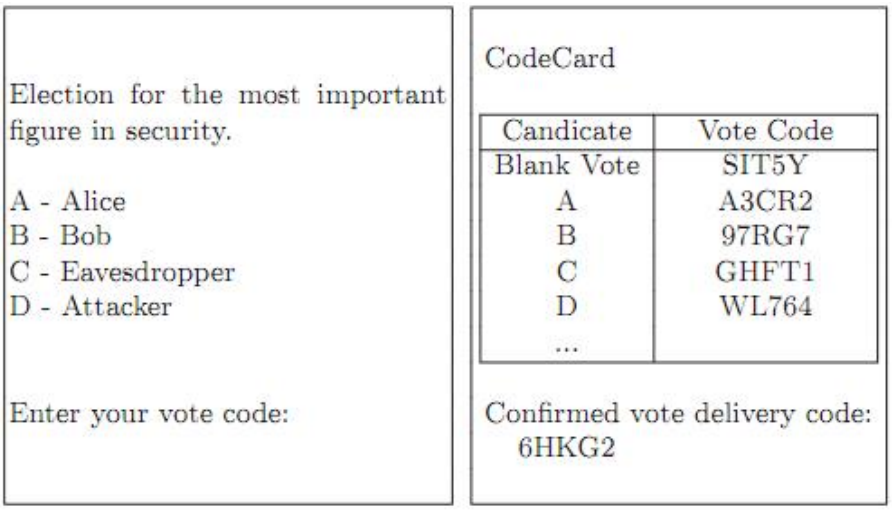

Fig. 13. Example of a ballot (left) and a CodeCard (right).

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 Figure 13, who wishes to vote for a candidate D only has to enter WL764 as the vote code[12].

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 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 plug-in to a voting system that protects the voter's choice from manipulation. Note that CodeVoting do not use different reply codes for each candidate. The reason for this is simple, 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:

1. publicly exposing the ballot some time before the election;
2. forcing the sorting of the candidates on the ballot, and the corresponding candidate IDs, in a verifiable way, e.g. alphabetical sorting;
3. using an image that is hard to forge/modify as a ballot.

Assumptions Authors argue that CodeVoting protects against vote manipulations at the voter's PC under the following assumptions:

[12] Image has taken from original CodeVoting paper.

1. The VST performs the protocols correctly and cannot be manipulated by the voter's computer;
2. The CodeCard is generated in a secure and controlled environment by the VST (the voter is the only person there);
3. The voter keeps her/his CodeCard secret;
4. 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. Currently, the authors of CodeVoting are implementing MarkPledge's technic, based on Neff's work [52].

8 Tools for the Solution

In this section, we will refer the various tools that our architecture needs in order to solve some Civitas' issues presented in section 6.1.

We would like to give as much freedom as possible to our adversary as this allows a more powerful adversary. This is the case if one removes all trust assumptions, however this is not possible in real life. Solving all problems and assumptions in one take is very complex, hence the focus needs to be in a few assumptions and solve them properly.

The main purpose of this work is to improve the *trust in voting client*. The latter is the main problem for remote voting and if we want to create a system that allows voting from anywhere in the world through the internet, our efforts should go to solve this.

As we have seen here in CodeVoting [37] (See section 7.2.), it is possible to have a high level of trust in a voting client, with reduced impact of trust assumptions. Clarkson et al. [18] did have this problem in mind when they created Civitas. They solved the problem with a trust assumption, assumption which we intent eliminate now.

Other assumption in Civitas is *the adversary cannot simulate a voter during registration*. This can be easily fixed with the use of in-person registration.

In this section, the Civitas' protocol with a new architecture will be described.

8.1 New and Changed Entities

In Civitas, there are five different groups of entities or agents: a supervisor, a registrar, voters, registration talliers, tabulation talliers, ballot boxes and a bulletin board. In this new solution, a new entity is created: a CodeCardReplier.

Figure 14 describes this new architecture. The changes and new responsibilities of these entities detailed below.

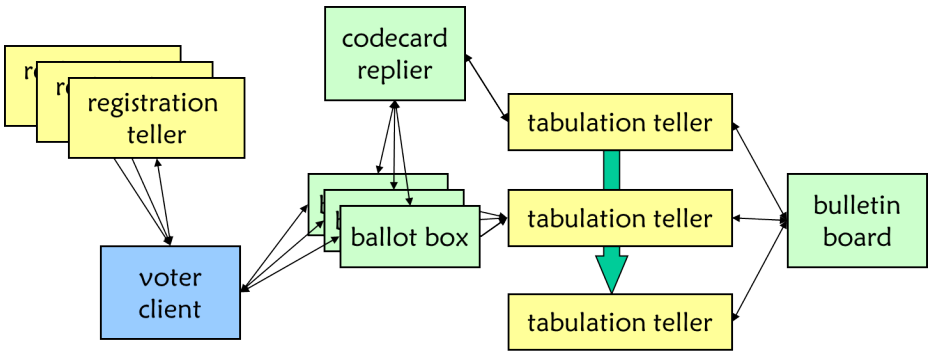

Fig. 14. New architecture based in Civitas.

The entities described below remain unchanged from Civitas:

- *Supervisor*: The supervisor is the administrator of an election. He is responsible for selecting the participating registration and tabulation talliers, for setting up the ballot design and for starting and stopping the election;
- *Voters*: Voters register for voting, get their credentials and cast a vote;
- *Tabulation Talliers*: The tabulation talliers are responsible for tallying the votes while preserving the anonymity of voters;
- *Bulletin Board*: The bulletin board is used by election authorities to record all the information needed for verifiability of the election.

A new entity was added to these which is described below:

- *CodeCardReplier*: This entity has the role of replying to each voter the correct reply code, decrypting the vote with its private key, reading the votes from the ballot boxes.

The entities below were changed to support the new entity:

- *Registrar*: The registrar authorizes voters and now is based in in-person registration. The ID system is based in smart card technology;
- *Registration Talliers*: The registration talliers together generate the credentials that are used by the voters for voting and one of the credentials needed is already inside the smart card issued by *registrar*;
- *Ballot Box*: The ballot boxes are instances of an insert-only log service. They are used by voters to cast their votes and have one additional function, reporting their contents to the CodeCardReplier and at the end of an election.

These agents use an underlying log service that implements publicly readable, insert-only storage. Integrity of messages in a log is ensured by digital signatures. Agents must sign messages they insert, ensuring that the log service cannot forge new messages. The log service must sign its responses to reads, ensuring that attempts to present different views of log contents to different readers can be detected. Multiple instances of the log service are used in a single election.

8.2 Registration

Civitas uses a remote agent *registrar* in the *setup phase* to authorize who can vote or not. Authorized voters need to get their credentials from this agent and doing this remotely can be dangerous, because an adversary can impersonate a voter and steal his credential. The latter is what distinguishes a voter from an adversary and therefore it must be kept private.

If we change the remote agent *registrar* by an *in-person registar*, the danger is greatly reduced. The credentials can be stored in and managed by a smart card[13], just like the *Citizen's Card*, used in Portugal[14]. Smart cards are the key idea for both Civitas and CodeVoting.

This will manage the problem of:

[13] See Appendix A for more detailed information about smart cards.
[14] See Appendix B for more information regarding Portugal ID card.

– *credential management*;

– *adversary simulating a voter in registration.*

Remember that these issues come from Civitas, since Clarkson et al. [18] did not address them in Civitas, rather put them in trust assumptions.

Much like in the Citizen's card enrolment, the voter must prove his identity, which is checked both by systems and humans. This reduces greatly the probability of identity theft. All relevant personal data and credentials are stored inside the Citizen's card and just similar smart card's application, both its hardware and software are certified and its architecture is public, for the sake of transparency.

Registration begins with the potential voter going to the local authorities' office and registrating himself, giving his personal data to the officer on duty that confirms his identity.

Fig. 15. Portuguese citizen's card.

After the voter is registered, the data that he provided is processed and analysed, to confirm the identity of the voter. After this process is completed, the voter will receive a letter in his address with the personal codes (PIN) of the smart card. The received letter is private and should not be shared with anyone, under the risk of compromising privacy. The PIN codes are used for:

– Accessing to stored data inside the Citizen's card;

– construct a valid or invalid credential (intentionally), as it will be described later.

Afterwards, the voter takes the letter to the local authorities' office, where he receives the Citizen's Card (Figure 15). This card will be later used to authenticate the voter and cast a vote.

8.3 Trust in Voting Client

This is the primary goal that we want to solve in Civitas. Adding the idea of code cards to Civitas' protocol will allow the voter to confirm that his vote was registered correctly in the ballot box. It is still needed to convince the coercer (who could be right next to the voter), that the vote is valid and that it has arrived correctly to its destination. So it is needed to maintain the credentials faking ability from Civitas, and allow the voter to resist coercion and use codes as a receipt confirmation.

Why confirmation? The voter needs to have sure that his vote is correctly captured by the election's server. Lacking this the vote could be lost, stolen or tampered without the voter's knowledge.

Voting remotely In order to vote in a remote environment, a voter needs the following requirements:

1. *Computer*;
2. *Internet Connection*;
3. *Certified Smart Card Reader with Keypad and Display*;
4. *Smart Card*.

The *first* and *second* requirements are easy to explain and understand. This protocol is founded on remote voting, and the best way to do it is to use the most wide spread network: the internet. The computer can be any machine which has ability to connect to the certified smart card reader with keypad and screen. If this device is *bluetooth* enable, then the *second* requirement can be almost any nowadays system, like a smartphone or a tablet, which generally have *bluetooth*.

The *third* requirement is more specific, since it can not use an ordinary smart card reader. This is required because the hardware that reads the smart card must be certified. It will also need to receive the input from the voter without it being changed by any means, in particular an infected computer host. Figure 16 shows an example, its hardware and software are fully certified and its architecture public. This smart card reader ensures that all the input is correctly captured:

- *Display*: Shows inputs from voters and outputs from the smart card;
- *Numeric keypad*: Allows numeric inputs for voter's candidate choice, input PIN;
- *Hardware and software certified*: Ensures that no tampering is possible.

This certified smart card reader would be distributed at the same time that voters obtained his Citizen's card at the local authorities' office. The only action needed for it to work is to plug it in the computer. All its software and drivers must be inside and prepared to plug and work on typical desktops' OS (like *Microsoft Windows*, *Apple Mac OS* or *Linux*) or mobile's OS (like *Google Android*, *Windows Phone 7* or *Apple iOS*).

Fig. 16. Smart Card Reader with keypad and screen.

8.4 Remote Voting

In the election period, the voter has to decide on which candidate he is going to vote. It is time to cast the vote and, to do so, the voter logs on the web page of the elections, using the smart card (after the certified smart card reader is connected to the voter's computer), which recognises the smart card's holder as a person eligible to vote. This page will have a connector to the smart card reader driver and will allow both to communicate. Without the correct recognition of the smart card by the PC or the web page, the voter cannot proceed with the voting process[15].

After the connecting is established, the web page presents the voter with the choices for the elections running. The voter inserts his smart card into the reader and authenticates himself into the voting client, through the certified hardware device. In order to cast a vote, the voter must choose a candidate and type in his choice. The smart card will compute the vote as:

$\langle Enc(s; K_{TT}), Enc(v; K_{TT}), P_w, P_k \rangle$

Where:

- s is the private credential;
- v is the voter's choice;
- P_w is the zero-knowledge proof that the vote is well-formed with respect to the ballot design of the election;

[15] There's much to talk about this connections, but it is out of the scope of this work.

– P_k is the zero-knowledge proof that shows that the submitter simultaneously knows s and v.

After this, the vote is again encrypted with the CodeCardReplier's key:

$\langle Enc_{CCR}(Enc(s; K_{TT}), Enc(v; K_{TT}), P_w, P_k)\rangle$

Where:

– Enc_{CCR} is the CodeCardReplier's encrypt key.

The smart card computes the vote and asks for a PIN. If this key is wrongly inputted, the vote will be casted but it will be invalid, since the credential will be fake. This is the Civitas' ability to construct a fake credential, by making the smart card running a local algorithm to produce a fake credential in the vote.

The key idea [40] that enables voters to resist coercion (*coercion Resistance*), and defeats vote selling, is that voters can substitute fake credentials for their real credentials, then behave however the adversary demands:

– If the adversary demands the voter *to submit a particular vote*, then the voter *does so with a fake credential*;
– If the adversary demands the voter *to sell or surrender a credential*, then the voter *supplies a fake credential*;
– If the adversary demands the voter *to abstain*, then the voter *supplies a fake credential to the adversary and votes with a real one.*

To construct a fake credential, the voter's smart card runs an algorithm to produce fake private credential shares that, to an adversary, are indistinguishable from real shares. The faking algorithm requires the voter's private designation key, which is a PIN. The voter's smart card combines these shares to produce a fake private credential; the voter's public credential remains unchanged.

The vote will appear real and valid to everyone, except the voter, who casted the wrong *private credential* on purpose to cheat on the coercer. Both the coercer and voter will receive the confirmation code from the *CodeCardReplier*, that matches the code that is visible on the smart card's display. This is much like CodeVoting using code cards, but without having them printed. This allows the system to be more simple and the voting process more smooth.

Putting the right *private credential*, makes the smart card cast a vote with the valid private credential. All the rest remains the same as previous example.

8.5 Ballot Box and CodeCardReplier

As described before, the *ballot boxes* are insert-only entities in Civitas, that are distributed and eventually compromised. In order to have the confirmation delivery code sent back to the voter's computer, it is needed a new server side entity to reply back to client voting and shows the confirmation code that matches

with the one displayed on the displayed on the *certified smart card reader*. This confirmation allows the voter to know that the vote has reached an entity that knows the correct reply code.

Since we do not want to change the *ballot box* properties from Civitas, which are insert once and read-only, we propose a new trustworthy entity that its only function is to read the contents of the ballot boxes (remember from Civitas that the votes casted are still encrypted) and reply to the vote's senders. This way all Civitas assumptions regarding the ballot boxes remain intact.

CodeCardReplier, the trustworthy entity This new entity has the role of replying to each voter client the correct reply code.

The vote that the ballot box receives is in this format:

$\langle Enc_{CCR}(Enc(s; K_{TT}), Enc(v; K_{TT}), P_w, P_k)\rangle$

The *CodeCardReplier* checks the ballot boxes for new stored votes and decrypts it with its private key, Dec_{CCR}. The reply code is:

$\langle Enc(v; K_{TT})\rangle$

As it can be seen, the reply code is just the voter's choice encoded as in original Civitas. This works because the vote can be split in its four parts as described in section 8.4.

This way, no printed CodeCards are needed and every time a voter sends a vote, the reply code is different, either the credential is fake or not. This saves time and complexity in voting process.

8.6 Tabulation

After the elections close, the votes must be tallied by the tabulation tellers. These are the steps needed to do:

1. All *Ballot boxes* post commit(received votes) on *CodeCardReplier*;
2. The *Supervisor* post signed copy of all received *Ballot boxes* commitments;
3. All received votes are Dec_{CCR} and committed to *Bulletin Board*;
4. All *Tabulation tellers* proceed sequentially through the following phases[16]. The *Bulletin Board* is used as a public broadcast channel for all posts and all this posts must be be signed, and all messages retrieved from it should have their signatures verified:
   (a) *Retrieve Votes* – Retrieve all votes from all endorsed *Ballot boxes*. Verify the *Ballot boxes* commitments. Let the list of votes be A;
   (b) *Check Proofs* – Verify all P_k and P_w in retrieved votes. Eliminate any votes with an invalid proof. Let the resulting list be B;

[16] Each phase has a list (e.g., A, B, etc.) as output. In each phase that uses such a list as input, verify that all other tellers are using the same list.

(c) *Duplicate Elimination* – Run PET(s_i, s_j)[17] for all $1 \leq i < j \leq |B|$, where s_x is the encrypted credential in vote B[x]. Eliminate any votes for which the PET returns 1 according to a revoting policy; let the remaining votes be C;

(d) *Mix Votes* – Run MixNet(C) and let the anonymized vote list be D;

(e) *Mix Credentials* – Retrieve all credentials from *Bulletin Board* and let this list be E. Run MixNet(E) and let the anonymized credential list be F;

(f) *Invalid Elimination* – Run PET(s_i, t_j) for all $1 \leq i \leq |F|$ and $1 \leq j \leq |D|$, where $s_i = F[i]$ and $t_j = D[j]$. Eliminate any votes (from D) for which the PET returns 0. Let the remaining votes be G;

(g) *Decrypt* – Run *DistDec* on all encrypted choices in G. Output the decryptions as H, the votes to be tallied;

(h) *Tally* – Compute tally of H using an election method specified on *Bulletin Board* by the *Supervisor*. Verify tally from all other tellers.

5. *Supervisor* endorse tally.

Only the beginning of the tabulation phase is new, this the votes being Dec_{CCR} in order to be tally in the correct format, just like in Civitas.

8.7 Coercion-Resistance

The removal of the *voting client trust* must not imply the loss of the *coercion-resistance* property. Here we will explain why this will not happen. The formal definition requires Civitas to defend against attacks in which the adversary demands secrets known to the voter, and attacks in which the adversary demands the voter to submit a value chosen by the adversary. This value might be a legitimate vote or a random value. The adversary may even demand the voter to abstain by submitting no value at all.

It will be checked, based on Civitas and on our own implementation, what can the adversary do and perform, and if coercion-resistance holds or not.

Threat model from Civitas From Civitas we have seen that authors required it to be secure with respect to an adversary with the following capabilities:

– The adversary may corrupt a threshold of the election authorities, mutually distrusting agents who conduct an election. Agents might be humans, organizations, or software components;

[17] Plaintext equivalence test (PET) is used to compare ciphertexts. Given c and c', a PET reveals whether $Dec(c) = Dec(c')$, but nothing more about the plaintexts of c and c'.

- The adversary may coerce voters, demand their secrets, and demand any behaviour of them – remotely or in the physical presence of voters. But the adversary may not control a voter throughout an entire election, otherwise the voter could never register or vote;
- The adversary may control all public channels on the network. However, it is assumed the existence of some anonymous channels, on which the adversary cannot identify the sender, and some untappable channels, which the adversary cannot use at all;
- The adversary may perform any polynomial-time computation.

These are the threats models that original Civitas allows the adversary to perform.

What can the Coercer do? Besides what was mention before, the coercer can perform these additional actions:

- The adversary can see the reply code on the smart card's display. It only proves that the vote casted by the voter arrived at the ballot box. When trying to cheat the adversary, it is needed to convince him that the vote arrived correctly as the adversary wanted;
- The adversary can capture the vote. Without the correct reply code, the voter will vote again in another client voting;
- The adversary can capture the reply code from the *CodeCardReplier* and choose not to show it to the voter or tamper it. Either the options the voter will cast his vote again in another client voting;
- The adversary can control the voting client. He can see and change all traffic coming in or out the smart card reader;
- The adversary can try to assume the voter's identification, by registrating himself instead of the voter. The cost and complexity of bypassing the local authorities' officer is very high;
- The adversary can force the voter to vote in any option.

Recall that coercion resistance is a strong form of privacy in which it is assumed that the adversary may interact with voters, in which case, the adversary may instruct targeted voters to divulge their private keys, or may specify that these voters cast votes in a particular form.

The key idea [40] that enables voters to resist coercion (*coercion Resistance*), and defeats vote selling, is that voters can substitute fake credentials for their real credentials, then behave however the adversary demands:

- If the adversary demands the voter *to submit a particular vote*, then the voter *does so with a fake credential.*
  The vote created is $\langle Enc(s'; K_{TT}), Enc(v; K_{TT}), P_w, P_k \rangle$;

- If the adversary demands the voter *to sell or surrender a credential*, then the voter *supplies a fake credential.*
  The vote created is $\langle Enc(s'; K_{TT}), Enc(v; K_{TT}), P_w, P_k \rangle$;
- If the adversary demands the voter *to abstain*, then the voter *supplies a fake credential to the adversary and votes with a real one.*
  The vote created is $\langle Enc(s'; K_{TT}), Enc(v; K_{TT}), P_w, P_k \rangle$;

Where:

- *s'* is the fake private credential.

It would be easy to eliminate votes containing duplicate or invalid credentials if credentials could be decrypted in tabulation phase, but this would fail to be coercion-resistant, because voters' private credentials would be revealed. A zero-knowledge protocol called a plaintext equivalence test (PET) is used to compare ciphertexts. Given *c* and *c'*, a PET reveals whether $Dec(c) = Dec(c')$, but nothing more about the plaintexts of *c* and *c'*. For duplicate elimination, a PET must be performed on each pair of submitted credentials. To eliminate invalid credentials, PETs must be performed to compare each submitted credential with every authorized credential.

Hence a coercion-resistant voting system is one in which the user can deceive the adversary into thinking that she has behaved as instructed, when the voter has in fact cast a vote according to her own intentions. That's what happens in this new architecture, as long as the trust assumptions hold.

8.8 Alternative Solutions

In this section is explained the reasons for some solution's choices, their alternative solutions and why the chosen ones are better than the alternative ones.

Encrypt voter's choice In new architecture, the vote is computed in the following way:

$\langle Enc(s; K_{TT}), Enc(v; K_{TT}), P_w, P_k \rangle$

Where:

- s is the private credential;
- v is the voter's choice;
- P_w is the zero-knowledge proof that the vote is well-formed with respect to the ballot design of the election;
- P_k is the zero-knowledge proof that shows that the submitter simultaneously knows s and v.

After this, the vote is again encrypted with the CodeCardReplier's key:

$\langle Enc_{CCR}(Enc(s; K_{TT}), Enc(v; K_{TT}), P_w, P_k)\rangle$

Where:

– Enc_{CCR} is the CodeCardReplier's encrypt key.

If the vote was Enc_{CCR} partially, like:

$\langle Enc_{CCR}(Enc(v; K_{TT}))\rangle$

The submitted vote would be like this:

$\langle Enc(s; K_{TT}), Enc_{CCR}(Enc(v; K_{TT})), P_w, P_k\rangle$

An adversary could capture the vote, tampering the vote in which the vote would be invalided when tallied but the reply code was correctly delivered. The adversary damaged the vote by changing parts of the vote, like $Enc(s; K_{TT})$ or the zero-knowledge proofs (P_w, P_k).

Printed CodeCards The solution presented in this thesis did not went for printed codes because they would imply the voter to obtain them previously to the voting phase. Even if the codecards were not connected to the voter's credentials, it would still be needed to create the codecards some time before the elections' time because of the matching candidates to codes.

This alternative solution that is used in CodeVoting, increased the complexity for the voter. With this work's solution the voter just needs to have his smart card (which is his country standard ID card) to vote.

8.9 New Trust Assumptions

Regarding the changes we have made to the Civitas protocol, it is time to review the assumptions that Clarkson et al. [18] have created for their voting protocol. The trust assumptions that were added in the new solution are marked with a X'. The trust assumptions that remain unchanged keep the same number used to describe it in section 6.2. The assumptions that were removed are with its description without italic.

Assumption 1: The adversary cannot simulate a voter during registration.

This assumption is removed with the new solution, since the registration is made *in-person*. This choice has a downside, the system not fully remote. The upside is that credentials can be used in many elections.

Assumption 2: Each voter trusts at least one registration teller, and the channel from the voter to the voter's trusted registration teller is untappable.

This assumption is removed with the new solution, since the credentials are already inside the smart card. The need to in one registration teller is contained in Assumption 5.

Assumption 2': *The credentials inside the smart card cannot be corrupted by any means.*

All computation done inside the smart card is correct and by no means can be corrupted, in a useful time line. This is an assumption that is valid for almost every smart card implementation. It's always a matter of time and we need to keep in mind that solutions that today seem unbeatable, tomorrow can be defeated easily. Electronic voting is coming, either we like it or not, so it is necessary to always have that in mind and try always to improve what already is good.

Assumption 3: Voters trust their voting client.

This assumption is removed with the new solution, by using reply codes and a new entity that is trustworthy: *CodeCardReplier*.

Assumption 4: *The channels on which voters cast their votes are anonymous.*

This assumption comes right from Civitas and remains unchanged. This is only necessary to hide which computers voted and how many times.

Assumption 5: *At least one of each type of authority is honest.*

At least one of the ballot boxes to which the voter submits his vote is correct. This is weaker than the standard assumption (less than a third of the ballot boxes fail) made for Byzantine fault tolerance [14,46] and multi-party computation [31], which both require more expensive protocols.

There exists at least one honest tabulation teller. This assumption is not needed for verifiability, even if all the tellers collude or are corrupted – the proofs posted by tellers during tabulation will reveal any attempt to cheat. Fault tolerance techniques [27,46,61] would increase the difficulty of corrupting all the tellers.

Assumption 5': *The CodeCardReplier is trustworthy.*

Our solution relies on a entity that replies to the voter's client with the reply code. This entity must be trustworthy because without it, the reply codes could be tampered. One could assume the *CodeCardReplier* identity and reply correctly to the voter, destroying the vote. This would allow to voter to think that his vote was correctly captured, when it was not.

The more entities share the trust, the greater the effort must be made by an adversary to coerce the elections.

Assumption 6: *Cryptography works.*

Cryptography must work, guaranteeing that encryptions are secret, signatures authenticate. The Decision Diffie-Hellman (DDH) and RSA assumptions hold, and SHA-256 implements a random oracle.

8.10 Voting Steps

Here are described the steps the voter needs to follow in the new architecture, in order to cast a vote:

1. The voter needs to register to get a Citizen's card, in the presence of the local authorities' officer (*the registar*). This document is a standard ID to any citizen from the voter's country;
2. The voter receives in his address the letter with the PIN keys for accessing his Citizen's card;
3. The Citizen's card will come with one credential's private share, that will allow the voter to create a vote;
4. Within the voting period, the voter used a computer with a certified smart card reader and connection to the internet. There the voter uses the election's web page to cast the vote:

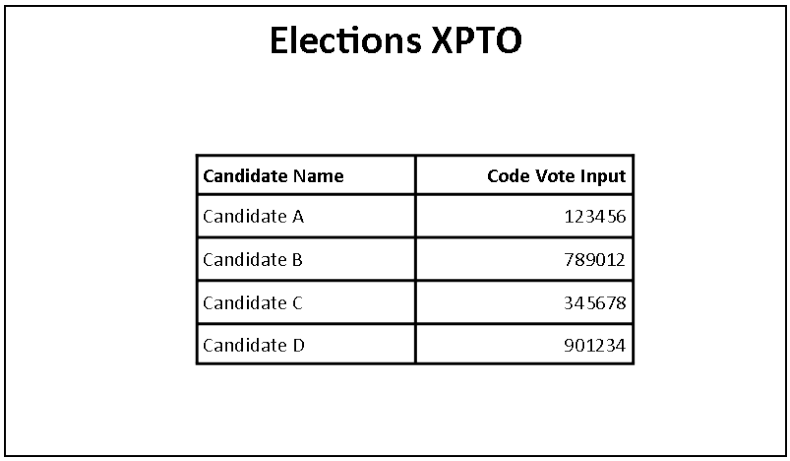

Fig. 17. Election's web page with candidates.

(a) The Citizen's card is recognized and enables the voter to cast a vote;

(b) Each one of the candidates presented to the voter has a unique code, the voter just needs to type the selected code in the certified smart card reader (see Figure 17);

(c) In order to confirm his choice, the voter types the PIN:

   i. *Correctly* to generate a valid credential;

   ii. *Wrongly* to generate a fake credential;

(d) The vote is encrypted by the smart card using the Civitas' protocol, using the credential generated in the step before;

(e) The vote is encrypted using the *CodeCardReplier's* key;

(f) The vote is sent to the election's servers (*ballot box*);

(g) The smart card reader displays the expected reply code. This code is the vote encrypted by Civitas' protocol created in step (d);

(h) The *ballot box* receives the vote. The *CodeCardReplier* receives the vote from the ballot box and decrypts it. Send the reply code and posts it in the election's web page;

(i) The voter waits for the reply code. When it arrives, he matchs the code displayed in the election's web page with the one displayed in the smart card reader;

5. The voter has sure that is vote was captured correctly.

9 Future Work

After all the steps made to increase the trust in Civitas, it is time to mention the opportunities this work presents to all those who want to continue it. There are several areas where work can be done and we describe some of them here.

9.1 Usability

The usability of Civitas has not been investigated, although usability is more important than security to some voters. The interface whom users should be analysed and taken into account since usability is more important than security to some voters [8,33]. This study requires a look into several systems that interact with people and obtain feedback from them, in order to create a interface and system that is pleasant to use by the voters.

9.2 Anonymity Network

Full anonymity on the Internet is not guaranteed since IP addresses can be tracked, allowing to identify the computer from which a certain post was made, albeit not the actual user. Tor [25] can be an excellent addition to the architecture and solve one of the most annoying problems for all protocol that need privacy and use internet as communication channel. As told before, with this problem solved, one important assumption would be removed and closed the security gap that today's remote systems have.

9.3 Removing hardware dependency

This work assumes that the used smart card reader is trustworthy and reliable, however, the best solution would be not being dependent of the latter. Although the purpose of trustworthy hardware was to increase the trust in the solution, having no trusted hardware increases the adversary's power, which is good as long as the solution remains trustable. This is another open opportunity to explore.

9.4 High Availability

Address availability is still not solved. To improve the availability of election authorities, they could be implemented as Byzantine fault-tolerant services [39]. Also, the encryption scheme used by Civitas could be generalized from the current distributed scheme to a threshold scheme[18]. This would enable election results to be computed even if some tabulation tellers become unresponsive or exhibit faulty behavior, such as posting invalid zero-knowledge proofs.

For a threshold scheme requiring k out of n tabulation tellers to participate in decryption, no more than $k - 1$ tellers may be corrupted, otherwise coercion resistance could be violated. For availability, a new trust assumption must be added: At least k tellers do not fail.

[18] See Appendix D for more detailed information.

10 Conclusion

It is time to review the work done in this thesis. We have seen the properties we wanted to obtain with this architecture.

This thesis describes the design of Civitas, together with the key ideas from CodeVoting. Civitas was chosen because it provides stronger security than previously implemented electronic voting systems, whose underlying voting scheme is proved secure under carefully articulated trust assumptions, and CodeVoting is a system which prevents the manipulation of the voter's vote.

The key idea that enables voters to resist coercion, and defeats vote selling, is that voters can substitute fake credentials for their real credentials, then behave however the adversary demands:

- If the adversary demands the voter *to submit a particular vote*, then the voter *does so with a fake credential*;
- If the adversary demands the voter *to sell or surrender a credential*, then the voter *supplies a fake credential*;
- If the adversary demands the voter *to abstain*, then the voter *supplies a fake credential to the adversary and votes with a real one.*

After analysing Civitas we discovered that it was based in six trust assumptions, one of which became this thesis' objective to remove:

Assumption 3: *Voters trust their voting client.*

Automatic vote manipulation at client side is one of the biggest dangers that prevent the widespread of Internet voting. This thesis also solved two other assumptions:

Assumption 1: *The adversary cannot simulate a voter during registration.*

Assumption 2: *Each voter trusts at least one registration teller, and the channel from the voter to the voter's trusted registration teller is untappable.*

The use of CodeVoting's ideas in combination with Civitas, a cryptographic voting protocol that runs completely inside the smart card will prevent the vote's manipulation from an adversary, as long as the new trust assumptions holds. Also the scheme used prevents reply codes to be tampered.

This new architecture adds three items to original Civitas which make trust in client voting not necessary:

- *Smart card*: to store credentials, compute and construct the vote;
- *Smart Card Reader with keypad and display*: this certified hardware is needed to use the smart card correctly and to display the expected confirmation code;

– *CodeCardReplier*: a trusted entity that is the central key for the confirmation codes.

The solution detailed here used the idea of code cards, but instead of having them printed them like in CodeVoting, in this solution the codes are just displayed on the smart card's display. This makes simpler the voting process and allows us to have one reply code for each vote submitted by the same voter.

This solution also makes a trade-off, instead of having to trust in all client voting (e.g. desktops, laptops, smartphones), the trust is upon a certified hardware, the smart card reader and the smart card itself. It's a fair trade since it is better to trust in a single certified hardware, which is easier to control, than all the client voting.

We think that a new step towards a secure voting system. We are optimistic about the future of electronic voting systems constructed. It is only a matter of time until the future arrives, and the future is remote electronic voting.

References

1. *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*. IEEE Computer Society, 2008.
2. Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.
3. Ben Adida, Olivier Pereira, Olivier De Marneffe, and Jean jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *In Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE*, 2009.
4. Ben Adida and Ronald L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In Ari Juels and Marianne Winslett, editors, *WPES*, pages 29–40. ACM, 2006.
5. Ammar Alkassar and Melanie Volkamer, editors. *E-Voting and Identity, First International Conference, VOTE-ID 2007, Bochum, Germany, October 4-5, 2007, Revised Selected Papers*, volume 4896 of *Lecture Notes in Computer Science*. Springer, 2007.
6. Michael Backes, Matteo Maffei, and Dominique Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *IEEE Symposium on Security and Privacy* [1], pages 202–215.
7. Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *PODC*, pages 274–283, 2001.
8. Benjamin B. Bederson, Bongshin Lee, Robert M. Sherman, Paul S. Herrnson, and Richard G. Niemi. Electronic voting system usability issues. In Gilbert Cockton and Panu Korhonen, editors, *CHI*, pages 145–152. ACM, 2003.
9. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
10. Josh Benaloh. Simple verifiable elections. In *EVT'06: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, pages 5–5, Berkeley, CA, USA, 2006. USENIX Association.
11. Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *EVT'07: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, pages 14–14, Berkeley, CA, USA, 2007. USENIX Association.
12. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 132–145. ACM, 2004.
13. James Burk and Amilcar Brunazo Filho. Brazil: The perfect electoral crime (ii). *http://portland.indymedia.org/en/2006/10/347846.shtml*, October 2006.
14. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186, 1999.
15. David Chaum. Surevote, international patent wo 01/55940 a1, June 2001.
16. David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.
17. David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

18. Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy* [1], pages 354–368.
19. Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *FOCS*, pages 372–382. IEEE, 1985.
20. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.
21. Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT*, pages 72–83, 1996.
22. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*, pages 103–118, 1997.
23. Adam M. Davis, Dmitri Chmelev, Michael R. Clarkson, Adam M. Davis, Dmitri Chmelev, and Michael R. Clarkson. Civitas: Implementation of a threshold, 2008.
24. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
25. Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.
26. Niels Ferguson and Bruce Schneier. *Practical Cryptography*. John Wiley & Sons, Inc., New York, NY, USA, 2003.
27. Stephanie Forrest, Anil Somayaji, and David H. Ackley. Building diverse computer systems. In *Workshop on Hot Topics in Operating Systems*, pages 67–72, 1997.
28. Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Active traffic analysis attacks and countermeasures. *Computer Networks and Mobile Computing, International Conference on*, 0:31+, 2003.
29. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.
30. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptology*, 20(1):51–83, 2007.
31. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.
32. Jörg Helbach and Jörg Schwenk. Secure internet voting with code sheets. In Alkassar and Volkamer [5], pages 166–177.
33. Michael C. Herron. Voting technology: The not-so-simple act of casting a ballot - by paul s. herrnson, richard g. niemi, michael j. hanmer, benjamin b. bederson, and frederick c. conrad. *Review of Policy Research*, 25(4):379–380, 2008.
34. Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT*, pages 539–556, 2000.
35. Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2000.
36. David R. Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner. Analyzing internet voting security. *Commun. ACM*, 47(10):59–64, 2004.

37. Rui Joaquim and Carlos Ribeiro. Codevoting: protecting against malicious vote manipulation at the voter's pc. In David Chaum, Miroslaw Kutylowski, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Frontiers of Electronic Voting*, volume 07311 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
38. Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira. Improving remote voting security with codevoting. In David Chaum, Markus Jakobsson, Ronald Rivest, Peter Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors, *Towards Trustworthy Elections*, volume 6000 of *Lecture Notes in Computer Science*, pages 310–329. Springer Berlin / Heidelberg, 2010.
39. Rui Joaquim, André Zúquete, and Paulo Ferreira. REVS - a robust electronic voting system. *IADIS*, December 2003.
40. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In Vijay Atluri, Sabrina De Capitani di Vimercati, and Roger Dingledine, editors, *WPES*, pages 61–70. ACM, 2005.
41. Aggelos Kiayias, Michael Korman, and David Walluck. An internet voting system supporting user privacy. In *ACSAC*, pages 165–174. IEEE Computer Society, 2006.
42. Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy*, pages 27–. IEEE Computer Society, 2004.
43. Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *ESORICS*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 2010.
44. R. Küsters, T. Truderung, and A. Vogt. Accountabiliy: Definition and Relationship to Verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010)*. ACM Press, 2010. To appear.
45. Miroslaw Kutylowski and Filip Zagórski. Verifiable internet voting solving secure platform problem. In Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg, editors, *IWSEC*, volume 4752 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2007.
46. Ricardo Lebre, Rui Joaquim, André Zúquete, and Paulo Ferreira. REVS - a robust electronic voting system. *International Conference on Applied Computing*, March 2004.
47. Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC*, volume 2587 of *Lecture Notes in Computer Science*, pages 389–406. Springer, 2002.
48. Epp Maaten. Towards remote e-voting: Estonian case. In Alexander Prosser and Robert Krimmer, editors, *Electronic Voting in Europe*, volume 47 of *LNI*, pages 83–100. GI, 2004.
49. Epp Maaten and Thad Hall. Improving the transparency of remote e-voting: The estonian experience. In Robert Krimmer and Rüdiger Grimm, editors, *Electronic Voting*, volume 131 of *LNI*, pages 31–46. GI, 2008.
50. Steven J. Murdoch and George Danezis. Low-cost traffic analysis of tor. In *IEEE Symposium on Security and Privacy*, pages 183–195. IEEE Computer Society, 2005.
51. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conference on Computer and Communications Security*, pages 116–125, 2001.
52. C. Andrew Neff. Practical high certainty intent verification for encrypted votes, 2004.
53. Lukasz Nitschke. Secure remote voting using paper ballots. *CoRR*, abs/0804.2349, 2008.

54. Rolf Oppliger. How to address the secure platform problem for remote internet voting. *Proceedings 5th Conf. on Security in Information Systems*, SIS 2002:153–173, 2002.
55. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
56. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *EUROCRYPT*, pages 522–526, 1991.
57. Aviel D. Rubin. Security considerations for remote electronic voting over the internet. *CoRR*, cs.CR/0108017, 2001.
58. Peter Y. A. Ryan and Steve A. Schneider. Prêt à voter with re-encryption mixes. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *ESORICS*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326. Springer, 2006.
59. Ahmad-Reza Sadeghi, Marcel Selhorst, Christian Stüble, Christian Wachsmann, and Marcel Winandy. Tcg inside?: a note on tpm specification compliance. In Ari Juels, Gene Tsudik, Shouhuai Xu, and Moti Yung, editors, *STC*, pages 47–56. ACM, 2006.
60. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *EUROCRYPT*, pages 393–403, 1995.
61. Fred B. Schneider and Lidong Zhou. Implementing trustworthy services using replicated state machines. *IEEE Security & Privacy*, 3(5):34–43, 2005.
62. Gerhard Skagestein, Are Vegard Haug, Einar Nødtvedt, and Judith E. Y. Rossebø. How to create trust in electronic voting over an untrusted platform. In Robert Krimmer, editor, *Electronic Voting*, volume 86 of *LNI*, pages 107–116. GI, 2006.
63. Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pages 25–25, Berkeley, CA, USA, 2001. USENIX Association.
64. CACM Staff. Seven principles for secure e-voting. *Commun. ACM*, 52(2):8–9, 2009.
65. Melanie Volkamer, Ammar Alkassar, Ahmad reza Sadeghi, Stefan Schulz, and Sirrix Ag. Enabling the application of open systems like pcs for online voting. In *In Proc. of Workshop on Frontiers in Electronic Elections*, 2006.
66. André Zúquete, Carlos Costa, and Miguel Romão. An intrusion-tolerant e-voting client system. *1st Workshop on Recent Advances on Intrusion-Tolerant Systems*, March 2007.

A Smart Cards

Smart cards are everywhere nowadays. They are simple and ordinary plastic card, just at the size of a credit card, with a microprocessor and memory embedded inside. Beside its tiny little structure it has many uses and wide variety of applications ranging from phone cards to digital identification of the individuals.

A.1 Types of Smart Cards

The term *smart card* is loosely used to describe any card that is capable of relating information to a particular application such as magnetic stripe cards, optical cards, memory cards, and microprocessor cards. It is correct to refer to memory and microprocessor cards as smart cards:

- Magnetic stripe cards. A magnetic stripe card has a strip of magnetic tape material attached to its surface. This is the standard technology used for bank cards and can only store data which cannot be updated;
- Optical cards. Optical cards use some form of laser to read and write to the card;
- Memory cards. Memory cards can store a variety of data, including financial, personal, and specialized information, but cannot process information.
- Microprocessor cards. Smart cards with microprocessors look like standard plastic cards, but are equipped with an embedded Integrated Circuit (IC) chip. They can store information, carry out local processing on the data stored, and perform complex calculations. These cards take the form of either *contact* cards (which require a card reader) or *contactless* cards (which use radio frequency signals to operate).

A.2 The Microprocessor Smart Card

The microprocessor smart card is defined as an IC chip contact card with a microprocessor and memory. The size of a credit card, this smart card contains a dime-sized microchip that can process and store thousands of bits of electronic data. Unlike passive devices (such as a memory card or magnetic stripe card) that can only store information, the microprocessor smart card is active and able to process data in reaction to a given situation.

This capability to record and modify information in its own non-volatile, physically protected memory makes the smart card a powerful and practical tool (smart cards are small and portable), they can interact with computers and other automated systems, and the data they carry can be updated instantaneously.

A.3 The Micromodule

As mentioned, smart cards are credit card-sized and made of flexible plastic, usually polyvinyl chloride[19]. They are embedded with a micromodule containing a single silicon integrated circuit chip with memory and microprocessor. The micromodule has eight metallic pads on its surface (see Figure 18), each designed to international standards:

- VCC – power supply voltage;
- RST – used to reset the microprocessor of the smart card;
- CLK – clock signal;
- GND – ground;
- VPP – programming or write voltage;
- I/O – serial input/output line.

The two remaining contacts are AUX1 and AUX2 respectively, and used for USB interfaces and other uses. Only the I/O and GND contacts are mandatory on a card to meet international standards; the others are optional.

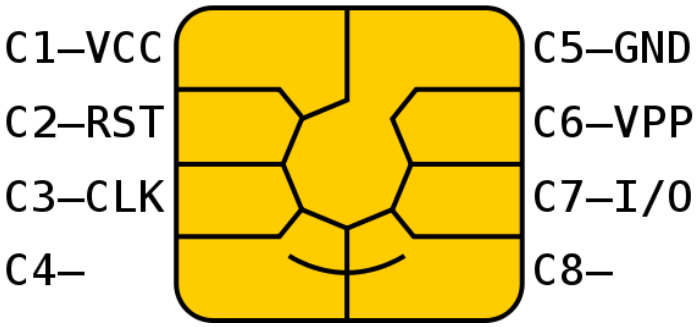

Fig. 18. Smart card pinout.

When a smart card is inserted into a Card Acceptance Device (CAD), such as a point-of-sale terminal, the metallic pads come into contact with the CAD's corresponding metallic pins, thus allowing the card and CAD to communicate. Smart cards are always reset when they are inserted into a CAD. This action causes the smart card to respond by sending an *Answer-to-Reset* (ATR) message, which informs the CAD, what rules govern communication with the card and the processing of a transaction.

The micromodule on board the smart card is made up of certain key components that allow it to execute instructions supporting the card's functionality.

[19] PolyVinyl Chloride usually known as *PVC*.

The Microprocessor Unit (MPU) executes programmed instructions. Typically, older version smart cards are based on relatively slow, 8-bit embedded microcontrollers. The trend has been toward using customized controllers with a 32-bit Reduced Instruction Set Computing (RISC) processor running at 25 to 32 MHz. The I/O Controller manages the flow of data between the Card Acceptance Device (CAD) and the microprocessor.

Read Only Memory (ROM) or Program Memory is where the instructions are permanently burned into memory by the silicon manufacturer. These instructions (such as when the power supply is activated and the program that manages the password) are the fundamentals of the Chip Operating System (COS) also known as the *Mask*.

Random Access Memory (RAM) or Working Memory serves as a temporary storage of results from calculations or input/output communications. RAM is a volatile memory and loses information immediately when the power supply is switched off.

Application Memory, which today is almost always double E-PROM (Electrically Erasable Programmable Read-Only Memory), can be erased electronically and rewritten. By international standards, this memory should retain data for up to 10 years without electrical power and should support at least 10,000 read-write actions during the life of the card. Application memory is used by an executing application to store information on the card.

A.4 Smart Card Operating System

Every smart card has an operating system (OS). The OS is the hardware-specific firmware that provides basic functionality as secure access to on-card storage, authentication and encryption. Only a few cards allow writing programs that are loaded onto the smart card – just like programs on a computer. This is a great way to extend the basic functionality of the smart card OS.

The smart card's Chip Operating System (COS), also referred to as the *Mask*, is a sequence of instructions permanently embedded in the ROM of the smart card. Like the familiar PC DOS or Windows Operating System, COS instructions are not dependent on any particular application, but are frequently used by most applications.

Chip Operating Systems are divided into two families:

- The general purpose COS. The general purpose COS has a generic command set in which the various sequences cover most applications;
- The dedicated COS. The dedicated COS has commands designed for specific applications and can contain the application itself (an example would be a card designed to specifically support an electronic purse application).

The baseline functions of the COS, common across all smart card products, include:

- Management of interchanges between cards and the outside world, primarily in terms of interchange protocol;
- Management of the files and data held in memory;
- Access control to information and functions (e.g. select file, read, write, and update data);
- Management of card security and the cryptographic algorithm procedures. Maintaining reliability, particularly in terms of data integrity;
- Management of various phases of the card's life cycle (e.g. microchip fabrication, personalization, active life, and end of life).

Generally, a card issuer must commit to a specific application developer, operating system and chip for each service. This leaves little flexibility to change any of these components without having to invest funds into a new software and/or hardware implementation. Early smart cards were costly and inflexible, but now, the trend is toward multi-application cards. For on-card application development of programs that run inside the secure environment of the smart card chip, it is recommended operating systems that have bigger market exposure such as JavaCard and MULTOS (multi-application smart card operating system).

B Citizen's Card

From the physical point of view the citizen's card has a *smart card* format and will replace the existing identity card, taxpayer card, Social Security card, voter's card and National Health Service user's card.

Fig. 19. Portuguese Citizen's card.

From the visual point of view (see Figure 19), the front of the card displays the holder's photograph and basic personal details. The back lists the numbers under which the holder is registered with the different bodies whose cards the citizen's card combines and replaces. The back also contains an optical reader and the chip.

From the electronic point of view the card has a contact chip, with digital certificates (for electronic authentication and signature purposes). The chip also holds the same information as the physical card itself, together with other data such as the holder's address.

The citizen's card will allow citizens to use a multichannel system in their interactions with public and private services, as follows:

- *Internet Channel / Site*: the site provides card-based access to electronic services, in order to give people a privileged channel for dematerialised interaction with public and private services. New online services will be made available, particularly in relation to house purchases and changes of address, which will only be possible using strong authentication. The site will also use the *single sign-on* concept as part of citizens' relationships with the Public Administration;
- *Telephone Channel / Contact Center*: this channel will enable people to obtain services by telephone, using a one-time password (which they will

get via the Citizen's Card and its reader system) to identify themselves and authenticate the transaction;

– *Personal Contact Channel / Others*: the Citizen's Card will interact with other services, particularly those that involve personal contact, thereby implementing the vision of the integration of back-offices and personal reception channels which underlies the *single contact point* concept.

B.1 Chip Physical Characteristics

In Citizen's card it was used a Infineon chip, namely the SLE66CX680PE model. Next are some of its characteristics:

– 0.22 μ technology;

– 244 KBytes ROM, 6144 Bytes RAM, 68 KBytes EEPROM;

– RSA 1024, 2048 bits;

– DES, TDES, AES;

– CC EAL5+;

– Supply voltage range:1.62 V, 3.0 V, 5.0 V;

– Operating Temperature range: -25 to +85°C;

– 500,000 write/erase cycles;

– Typical data retention 10 years.

The operation system used here is the OS ICitizen V2 64K, from the manufacturer *Axalto*. Its main characteristics are the following:

– Java Card 2.2.1 (http://java.sun.com/products/javacard/);

– Global Platform 2.1 (http://www.globalplatform.org/);

– 64K of memory for applications and data;

– T=0 e T=1;

– Multiple PIN Mangement;

– Dynamic Memory Management.

C Traffic Analysis

Traffic analysis is the process of intercepting and examining messages in order to deduce information from patterns in communication. It can be performed even when the messages are encrypted and cannot be decrypted. In general, the greater the number of messages observed, or even intercepted and stored, the more can be inferred from the traffic. Traffic analysis can be performed in the context of military intelligence or counter-intelligence, and is a concern in computer security.

Traffic analysis is also a concern in computer security. An attacker can gain important information by monitoring the frequency and timing of network packets. A timing attack on the SSH protocol can use timing information to deduce information about passwords since, during interactive session, SSH transmits each keystroke as a message [63]. The time between keystroke messages can be studied using hidden Markov models. Song, et al. [63] claim that it can recover the password fifty times faster than a brute force attack.

Onion routing systems are used to gain anonymity. Traffic analysis can be used to attack anonymous communication systems like the Tor anonymity network [25]. Steven J. Murdoch and George Danezis from University of Cambridge presented [50] research showing that traffic-analysis allows adversaries to infer which nodes relay the anonymous streams. This reduces the anonymity provided by Tor. They have shown that otherwise unrelated streams can be linked back to the same initiator.

C.1 Countermeasures

It is difficult to defeat traffic analysis without both encrypting messages and masking the channel. When no actual messages are being sent, the channel can be masked [28] by sending dummy traffic, similar to the encrypted traffic, thereby keeping bandwidth usage constant [26]. The military-versus-civilian problems applies in situations where the user is charged for the volume of information sent.

Even for Internet access, where there is not a per-packet charge, ISPs make statistical assumption that connections from user sites will not be busy 100% of the time. The user cannot simply increase the bandwidth of the link, since masking would fill that as well. If masking, which often can be built into end-to-end encryptors, becomes common practice, ISPs will have to change their traffic assumptions.

C.2 Onion Routing

Onion routing is a technique for anonymous communication over a computer network. Messages are repeatedly encrypted and then sent through several network nodes called onion routers. Each onion router removes a layer of encryption to uncover routing instructions, and sends the message to the next router where

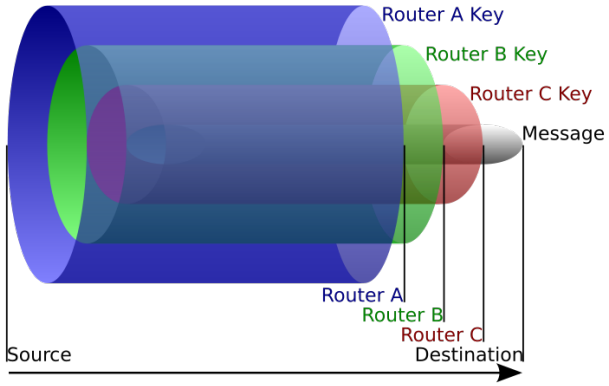

Fig. 20. Onion routing.

this is repeated. This prevents these intermediary nodes from knowing the origin, destination, and contents of the message.

The idea of onion routing (OR) is to protect the privacy of the sender and recipient of a message, while also providing protection for message content as it traverses a network.

Onion routing accomplishes this according to the principle of Chaum's mix cascades [17]: messages travel from source to destination via a sequence of proxies (*onion routers*), which re-route messages in an unpredictable path. To prevent an adversary from eavesdropping[20] on message content, messages are encrypted between routers. The advantage of onion routing (and mix cascades in general) is that it is not necessary to trust each cooperating router; if one or more routers are compromised, anonymous communication can still be achieved. This is because each router in an OR network accepts messages, re-encrypts them, and transmits to another onion router. An attacker with the ability to monitor every onion router in a network might be able to trace the path of a message through the network, but an attacker with more limited capabilities will have difficulty even if he or she controls one or more onion routers on the message's path.

Onion routing does not provide perfect sender or receiver anonymity against all possible eavesdroppers – that is, it is possible for a local eavesdropper to observe that an individual has sent or received a message. It does provide for a strong degree of *unlinkability*, the notion that an eavesdropper cannot easily determine both the sender and receiver of a given message. Even within these confines, onion routing does not provide any absolute guarantee of privacy; rather, it provides a continuum in which the degree of privacy is generally a function

[20] Eavesdropping is the act of secretly listening to the private conversation of others without their consent.

of the number of participating routers versus the number of compromised or malicious routers.

C.3 Tor, Anonymity Network

Tor is a implementation of onion routing[21] enabling Internet anonymity by thwarting network traffic analysis and surveillance. It is an open network that is intended to protect users' personal freedom, privacy, and ability to conduct confidential business, by keeping their internet activities from being monitored.

Tor works by bouncing communications around a distributed network of relays run by volunteers around the world. In doing so, it can prevent a third party with the capacity to monitor a user's internet connection from learning which sites they visit, and it prevents the sites they visit from learning the user's physical location. By keeping some of the network entry points hidden Tor is also able to evade many internet censorship systems, even ones specifically targeting Tor.

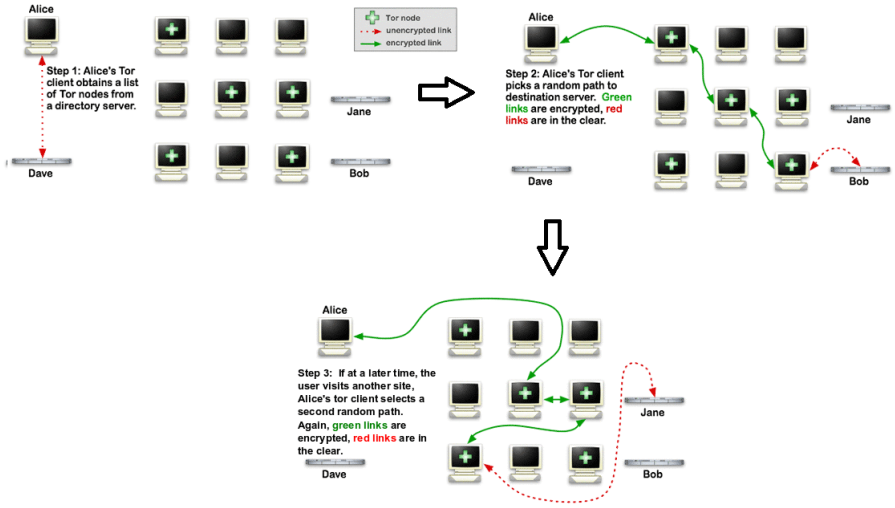

Fig. 21. How Tor works.

How Tor works Tor helps to reduce the risks of both simple and sophisticated traffic analysis by distributing your transactions over several places on the Internet, so no single point can link you to your destination. The idea is similar to using a twisty, hard-to-follow route in order to throw off somebody who is

[21] Onion routing is a technique for anonymous communication over a computer network.

tailing you – and then periodically erasing your footprints. Instead of taking a direct route from source to destination, data packets on the Tor network take a random pathway through several relays that cover your tracks so no observer at any single point can tell where the data came from or where it's going.

To create a private network pathway with Tor, the user's software or client incrementally builds a circuit of encrypted connections through relays on the network. The circuit is extended one hop at a time, and each relay along the way knows only which relay gave it data and which relay it is giving data to. No individual relay ever knows the complete path that a data packet has taken. The client negotiates a separate set of encryption keys for each hop along the circuit to ensure that each hop can't trace these connections as they pass through.

Once a circuit has been established, many kinds of data can be exchanged and several different sorts of software applications can be deployed over the Tor network. Because each relay sees no more than one hop in the circuit, neither an eavesdropper nor a compromised relay can use traffic analysis to link the connection's source and destination. Tor only works for TCP streams and can be used by any application with SOCKS support.

For efficiency, the Tor software uses the same circuit for connections that happen within the same ten minutes or so. Later requests are given a new circuit, to keep people from linking your earlier actions to the new ones.

D Threshold cryptosystem

In cryptography, a cryptosystem is called a *threshold cryptosystem* [23], if in order to decrypt an encrypted message a number of parties exceeding a threshold is required to cooperate in the decryption protocol. The message is encrypted using a public key and the corresponding private key is shared among the participating parties. Let n be the number of parties. Such a system is called *(t,n)*-threshold, if at least t of these parties can efficiently decrypt the ciphertext, while less than t have no useful information. Similarly it is possible to define *(t,n)*-threshold signature scheme, where at least t parties are required for creating a signature. Threshold versions of encryption schemes can be built for many public encryption schemes. The natural goal of such schemes is to be as secure as the original scheme. Such threshold versions have been defined for RSA and El-Gamal [29], just to mention some.