Abstract

The electronic vote promises the possibility of a convenient, efficient and safe way to capture and count votes in an election. 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

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, called ballot box.

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 various problems inherently in paper-based systems such as logistics involved in elections in paper, cost and resources involved are very large. Also the time needed for all phases of the electoral process is very huge (e.g. printing votes, tally votes).

1.2 Types of Electronic Voting

Three approaches to the problem of electronic voting have been proposed so far [13]:
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;

Kiosk voting – voting takes place through publicly available terminals. 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 nor the environment can be controlled [14].

Internet voting systems are appealing for several reasons: (i) People are getting more used to work with computers to do all sorts of things, namely sensitive operations such as shopping and home banking; (ii) They allow people to vote far from where they usually live, helping to reduce abstention rates; (iii) 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. 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. 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 Denial of Service 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.

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 [6, 11]:

- 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.

3 Civitas

Civitas is based on a voting protocol by A. Juels et al. (JCJ) [10] and adapts most of its general functionalities.

3.1 Design

Civitas refines and implements a voting scheme, which is referred to it as JCJ because it was developed by Juels, Catalano, and Jakobsson [10].

Agents There are five kinds of agents in the Civitas voting system:

- 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.

**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:

- **Resisting coercion** – The key idea [10] 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.

**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 [4]. 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.

3.2 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.

3.3 Security

The Civitas voting system requires certain assumptions about the trustworthiness of agents and system components [5]. 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.

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.

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.

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.

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 [2].

4 Tools for the Solution

In this section, we will refer the various tools that our architecture needs in order to solve some Civitas’ issues.
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 [9], it is possible to have a high level of trust in a voting client, with reduced impact of trust assumptions. Clarkson et al. [5] 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.

4.1 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 1 describes this new architecture. The changes and new responsibilities of these entities detailed below.

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.

4.2 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 registrar, the danger is greatly reduced. The credentials can be stored in and managed by a smart card, just like the Citizen’s Card, used in Portugal. Smart cards are the key idea for both Civitas and CodeVoting.

This will manage the problem of credential management and adversary simulating a voter in registration.

Remember that these issues come from Civitas, since Clarkson et al. [5] 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 registering himself, giving his personal data to the officer on duty that confirms his identity.

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 and to construct a valid or invalid credential (intentionally).

The voter takes the letter to the local authorities’ office, where he receives the Citizen’s Card. This card will be later used to authenticate the voter and cast a vote.

4.3 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 coэрcer (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.

In order to vote in a remote environment, a voter needs the following requirements: Computer, Internet Connection, Certified Smart Card Reader with Keypad and Display and Smart Card.

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. 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.

4.4 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.

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:

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

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:

\[ \langle \text{Enc}_{CCR}(\text{Enc}(s; K_{TT}), \text{Enc}(v; K_{TT}), P_w, P_k) \rangle \]

Where:
- \( \text{Enc}_{CCR} \) 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 [10] 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.

4.5 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.

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:

\[ \langle \text{Enc}_{CCR}(\text{Enc}(s; K_{TT}), \text{Enc}(v; K_{TT}), P_w, P_k) \rangle \]

The CodeCardReplier checks the ballot boxes for new stored votes and decrypts it with its private key, \( \text{Dec}_{CCR} \). The reply code is:
\( \langle \text{Enc}(v; KT_{TT}) \rangle \)

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 4.4.

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.

### 4.6 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 **DecCCR** and committed to **Bulletin Board**;
4. All **Tabulation tellers** proceed sequentially through the following phases as in section 3.1.

### 4.7 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.

**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 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 may 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 may capture the vote. Without the correct reply code, the voter will vote again in another client voting;
- The adversary may 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 by-passing 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 [10] 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 \( \langle \text{Enc}(s'; K_{TT}), \text{Enc}(v; K_{TT}), P_w, P_k \rangle \);
- If the adversary demands the voter to sell or surrender a credential, then the voter supplies a fake credential. The vote created is \( \langle \text{Enc}(s'; K_{TT}), \text{Enc}(v; K_{TT}), P_w, P_k \rangle \);
- 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 \( \langle \text{Enc}(s'; K_{TT}), \text{Enc}(v; K_{TT}), P_w, P_k \rangle \);

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 \( \text{Dec}(c) = \text{Dec}(c') \), 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.

### 4.8 New Trust Assumptions

Regarding the changes we have made to the Civitas protocol, it is time to review the assumptions that Clarkson et al. [5] 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 earlier. 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 [3, 12] and multi-party computation [8], 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 [7, 12, 15] would increase the difficulty of corrupting all the tellers.

**Assumption 5':** The CodeCardReplier is trustworthy.

Our solution relies on an 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.

## 5 Conclusion

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. 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 Assumption 1 *The adversary cannot simulate a voter during registration* and 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: (i) **Smart card**, to store credentials, compute and construct the vote; (ii) **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; and (iii) **CodeCardReplier**: an 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


