# CIVIS - A Coercion-Resistant Election System

**Roberto Araujo**[1], **André Neto**[1], **Jacques Traoré**[2]

[1]Faculdade de Computação – Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

[2]Orange Labs
42 rue des Coutures, BP 6243, 14066
Caen Cedex, France

rsa@ufpa.br, andre.neto@icen.ufpa.br, jacques.traore@orange.com

***Abstract.*** *Coercion is an intrinsic problem of Internet elections that certainly prevent its wide use. Although there is no optimal solution for this problem, modern cryptographic election schemes can mitigate it. Most of these proposals, however, were never used to carry out real elections due to the lack of software implementations. As a result, it is not possible to test these schemes in realistic election scenarios and so their interest remain purely theoretical. In this context, this work introduces the CIVIS election system. CIVIS is a web-based system that implements ideas to fight coercive attacks. It is based on a secure coercion-resistant election scheme and it shows the applicability of this scheme to accomplish realistic elections. Besides introducing CIVIS, we show that a famous Internet-based election system used in Brazil does not satisfy important security properties for voting.*

## 1. Introduction

Internet elections have many inherent benefits. Voters, for instance, can cast their votes from any connected place, using their own devices. These benefits probably motivated countries such as Switzerland [Serdult et al. 2015] and Estonia [The National Election Committee 2017] to adopt it in government elections. However, ensuring security in Internet elections is not an easy task and many researchers [Participants of the 2007 Dagstuhl Conference on Frontiers of E-Voting 2007] do not recommend them. The lack of a controlled place to cast votes and the intrinsic insecurity of the Internet environment certainly increase the attack vectors.

One of the main challenges of Internet elections is to solve the coercion problem. Because adversaries (coercers or vote buyers) can observe voters while voting, they can easily influence voters to select their candidates. Although there is no optimal solution for this problem, Juels, Catalano, and Jakobsson [Juels et al. 2005] (JCJ) introduced an idea to mitigate it. In their idea, each eligible voter receives a credential (i.e. a random password) that she uses to cast her vote. This credential identifies votes to be counted in the final tally. A voter under coercion, however, could use a fake credential (i.e. a fake password) to vote. Both credentials are indistinguishable to the adversary, but fake credentials will be identified and removed from the tally without enabling the adversary to determine which one was discarded. Based on JCJ's ideas, several schemes were proposed. Most of them, however, remain theoretical as they were never implemented. As a

result, it is impossible either to evaluate these schemes or to test them in practical scenarios. The scheme due to ABRTY [Araújo et al. 2010] is an example. Their proposal is one of the most efficient coercion-resistant schemes for Internet elections nowadays, but to the best of our knowledge there is no previous implementation of this scheme.

**Contributions**. SIGEleição is an Internet-based voting system that has been used to carry out elections in many Brazilian universities. In this work, we first show that SIGEleição relies on strong assumptions to satisfy security properties. After that, we introduce the CIVIS election system. CIVIS is a web-based system developed from the ABRTY election scheme. It aims at showing the practicability of this protocol in real elections. This way, it can be used for tests purposes and to carry out elections. CIVIS was inspired by the Helios voting system [Adida 2008], so it borrows its main characteristics such as execution in a single server and management of multiple elections. CIVIS allows voters to deceive adversaries using JCJ ideas and its current version follows the design of ABRTY cryptographic election scheme. Although it is currently alpha software, the system is functional and will be available soon as open-source software. The CIVIS current version was conceived for small-scale elections. However, due to its design, the system can be adjusted to perform larger elections. To show the applicability of CIVIS, we present the results of a simulated election carried out in a university department.

**Related work**. This work is related to many secure election systems. CIVIS shares similarities with Helios voting system [Adida 2008]. However, the two systems have different purposes. Helios focuses on auditability and it was not conceived to defeat coercive attacks. This system allows voters to verify their votes are correctly formed by means of Benaloh's challenge [Benaloh 2006]. Differently, CIVIS mitigates coercive attacks. CIVITAS [Clarkson et al. 2008] and Mosaic [Abdellatif and Adouani 2014] are election schemes that deal with this problem properly. CIVITAS is an implementation of JCJ original scheme with improvements related to efficiency. However, CIVITAS is clearly impracticable as the complexity of its tallying phase is quadratic in the number of submitted ballots and therefore it could not be used for large real-world elections. Mosaic is based on a more efficient protocol than CIVITAS. Unfortunately, Mosaic relies on a cryptographic voting protocol that is not formally proved secure. This work is also related to the systems used in Estonia [The National Election Committee 2017] and in Switzerland [Serdult et al. 2015]. However, these systems do not mitigate all coercive attacks (e.g. forced abstention attack) as JCJ based proposals do.

**Organization**. This paper is organized as follows. The next section recalls the SIGEleição voting system and shows that it relies on strong assumptions to satisfy security properties. After that, Section 3 details the CIVIS election system. It presents an overview of CIVIS architecture as well as its main web and cryptographic building blocks. Section 4 presents the results of the simulated election that employs CIVIS. Finally, we present our conclusions and future works in Section 5.

## 2. The Sig-Eleição Voting system

### 2.1. Overview of SIGEleição Voting system

The SIGEleição voting system [Santos et al. 2017] is part of a broader management system (i.e. Sistemas Institucionais Integrados de Gestão - SIG) that is employed

by a number of Brazilian universities. This voting system is web-based and it was conceived for conducting secure elections. SIGEleição is famous in universities that uses the management system that it belongs to and has been used to carrying out election in many of these universities. However, unfortunately there is no complete description of SIGEleição voting protocol in the literature and this system is not open-source. Recently, [Santos et al. 2017] published a paper about this system that contains some important details about it. Based on this work, we describe next the SIGEleição system.

The system requires the generation of a random *secret key* for an election authority (EA) before the election. This authority keeps it in secrecy and should not reveal it. EA enters the secret key on the SIGEleição server before the voting period starts and the server stores it in its memory. SIGEleição uses this key in the generation of cryptographic hashes later on. At time of voting, once SIGEleição server receives a vote, it stores information about the voter and her vote apart. For this, the system maintains two database tables: one for the votes (i.e. the *Voto* table) and another for the voters (i.e. the *VotoEleitor* table). For each vote, the system stores in the *Voto* table a unique vote identification number (*id_voto*) and a number that identifies the election (*id_eleicao*) that the vote belongs to. Also, it stores *in plaintext* the candidate number (*id_chapa*) that the voter has voted for (without identifying the voter herself), and a value (*valor*) about the state of the vote (i.e. valid, white, or null). The system also computes a cryptographic hash that takes as input the *secret key* and the former data. The resulting hash as well as the other data are stored in the same line of the *Voto* table. The system stores the vote information in a random order to avoid the association between the voter and her vote. In a line of the *VotoEleitor* table, the system stores the *same* election identification number (*id_eleicao*) of the *Voto* table, the identification number of the voter (*id_pessoa*), the date that the system received her vote, and the ip address of the voter computer (*ip_eleitor*). Taking as input all these data and the EA *secret key*, it computes a cryptographic hash and stores the resulting hash value in the same line of *VotoEleitor* table. The system also generates a general hash of the election data. At the end of the election, EA enters his secret key to generate the hashes again and compares the new hash values to the values stored before. Finally, the system counts the votes that match both.

### 2.2. Security Properties for Voting

Cryptographic voting protocols are the core of any secure voting system and are designed to satisfy a set of security properties. Most of these properties come from general principles conventionally employed in elections. Others are more specific as coercion-resistance. We recall next security properties for voting that are widely discussed in the literature (see, for instance, the works of [Langer et al. 2009] and [Delaune et al. 2010]). **Eligibility.** Only eligible voters can cast valid votes and only one vote per eligible voter is counted. **Fairness.** All votes remain secret until the voting is completed. **Accuracy**. Upon cast, a vote cannot be altered, duplicated, or eliminated and all valid votes are counted correctly. **Vote Privacy.** A vote cannot be associated to the voter who cast it. **Universal and Individual Verifiability.** Universal verifiability means that anyone can verify the correctness of the results. Individual verifiability means that a voter can verify her vote was correctly counted. **Receipt-freeness.** A voter cannot obtain any information (i.e. a receipt) that can be used as a prove of her vote. **Coercion-resistance.** A voter cannot convince a coercer about her true vote intention.

### 2.3. A Sketch Analysis of SIGEleição

The design of a secure voting protocol is the first step towards the development of a secure voting system. To be secure, a protocol must fulfil at least the Eligibility, the Fairness, the Vote Privacy, and the Universal Verifiability properties. However, the other requirements are highly recommended as they allow more transparency and security for elections. In the next, we present a sketch analysis of SIGEleição with regards these properties. Because there is no complete description of SIGEleição publicly available, our analysis is based on the SIGEleição system presented by [Santos et al. 2017] (see Section 2.1). In this analysis, we abstract implementation details and consider that the communications channels are secure and authenticated.

If SIGEleição performs correctly, it receives a vote and stores its data in the *Voto* table and voter data in the *VotoEleitor* table. By looking at the field id_pessoa of the *VotoEleitor* table, the Election Authority (EA) can verify that only eligible voters voted. In addition, by looking at the *Voto* table, it is possible to verify that there is one vote per voter. This would satisfy the eligibility property. However, if a malicious system deviates the protocol, it could store the voter data in the *VotoEleitor* table exactly as posted by the voter, but it could store a different vote in the *Voto* table. By means of these tables, it is not possible to identify this malicious behavior. SIGEleição does not rely on any cryptographic primitive to keep the votes in secrecy until the end of the election. The votes are not encrypted, and the system stores them in *plaintext* in the field id_chapa of the *Voto* table. By looking at this table, an adversary (e.g. a malicious EA) can easily identify the winner option before the end of the election. In this case, the system does not satisfy fairness as well. Upon receiving a vote, SIGEleição stores it along with a cryptographic hash. The hash value is computed using the EA secret key that is stored in the server memory. Because the vote, the secret key, and other data are hashed together, an adversary would need this key to alter or to duplicate a vote stored in the *Voto* table. Also, the security of the cryptographic hash function ensures that the vote cannot be changed. If the EA is trustworthy and the secret key cannot be obtained from the server memory, the adversary cannot succeed. However, a malicious EA can easily alter or duplicate votes without being noted. In addition, because the secret key is stored in the server memory during the voting period, it is difficult to ensure that it cannot be obtained by adversaries. A hacker could compromise the server during the voting.

SIGEleição does not employ any standard cryptographic primitive to disassociate the vote from the voter. Instead, it relies on unrelated tables to disassociate them. The system stores data related to the vote in the *Voto* table and data related to the voter in the *VotoEleitor* table. Because SIGEleição does not use cryptographic primitives for ensuring vote privacy, the system must be trustworthy. Otherwise, it does not ensure vote privacy. Because the system knows the vote and the voter data, a malicious system can easily link a voter to her vote. To be universally verifiable, a system (and its corresponding protocol) must allow the verification of the properties above. In other words, anyone (voters, authorities, third parties, etc.) must be able to verify the correctness of the results. Supposing that the EA publicly reveals its secret key after the election, anyone can generate the hash values again and compare them to the hashes stored in the tables. However, as presented above, because there is no way to check if the data saved in the tables were not changed by the EA or by the system, the system is not universally verifiable. In addition, the system is not receipt-free. The voter may save all transcripts of her communication

with the system (e.g. session keys used in the TLS protocol) and reveal it to an adversary. These transcripts are used as a receipt to prove the vote to an adversary. He is now able to decrypt the communication data and verify the vote. Because SIGEleição is not receipt-free, it is also not coercion resistant. In addition, it does not satisfy individual verifiability as it does not provide any mechanism for voters to verify their votes.

In a perfect world, the SIGEleição system and its users would follow all the steps required. However, when designing a voting protocol and implementing it later, it must be considered that adversaries may show up and perform maliciously. SIGEleição relies strongly on its system and on the Election Authority. If they do not follow the protocol, the system does not satisfy important security requirements as described above.

## 3. The CIVIS Election System

As presented, SIGEleição strongly relies on one authority and on the system itself. In the next, we introduce a system with better security guarantees: the CIVIS voting system. CIVIS is based on the secure cryptographic coercion-resistant voting protocol due to ABRTY [Araújo et al. 2010].

### 3.1. Architecture Overview

The CIVIS architecture has three main parts: administration, voting, and public information. The administration part controls all the election phases and their different steps. The voting part is related to the vote casting. The public information part is responsible to make public all data that it receives. Each part has one or more modules. These modules can be viewed as independent applications that can be run in parallel or turned off as required. For instance, the voting module can be turned on just during the voting period.

The election map, setup, registration, tallying, and mix net modules compose the administration part. The election map allows the management of all election phases and their steps. It calls all the other administration modules and also controls the voting. By means of it, administrators may initialize or end a phase after executing their corresponding steps. Because the protocol that CIVIS is based on requires a number of steps to run an election, the system introduces the idea of a map. It makes election management more user-friendly. The remaining modules contain all steps necessary to generate the election, to register voters, and to tally votes. They use the properly cryptographic tools to accomplish their steps. The tallying module, in particular, controls a special module called mix net. This module manages the mix net cryptographic primitive. The voting part is composed of a unique module of the same name. It is the part of the system that deals with the voting phase. This module contains all steps related to the generation of the vote and it uses the suitable cryptographic tools for this purpose. The administration as well as the voting modules depend on the web bulletin board module (WBB) to publish any public information (e.g. encrypted votes). This module allows anyone to verify and to attest any public information about the election, e.g. the encrypted votes sent to the system. The mix net also relies on this module to broadcast information used to verify the mixing process.

The access to the administration modules depends on a preliminary authentication. In the current version of the system, this is performed via login and password. Different from the administration modules, the voting module does not require any previous authentication. Although this allows anyone to cast a vote using the system, this is a requirement

of JCJ based schemes. If a system requires voter authentication, then an adversary could easily force a voter to reveal her login and password, and vote on her behalf later on. Figure 1 illustrates the CIVIS architecture.

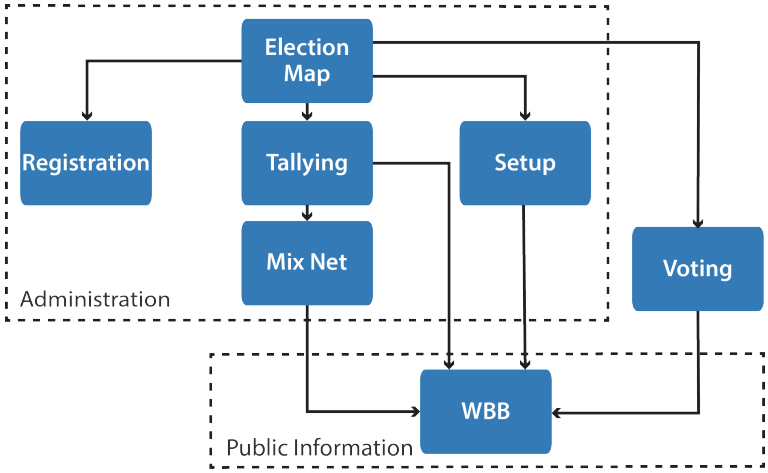


**Figura 1. Overview of the CIVIS architecture.**

### 3.2. Technological Building Blocks

CIVIS depends on a set of technologies to accomplish a web-based coercion-resistant election system. These mechanisms perform either on the user web browser (client side) or on the server side. We present them next.

**Python and the Django framework**. Most of the web part of CIVIS relies on the Django web framework [DSF 2017]. Django is a Python [PSF 2018] based open-source framework that follows the MVC (model, view, controller) concept. To perform computation on server side, CIVIS also relies on Python programming language. **JavaScript and jQuery Library**. The system employs cryptographic mechanisms that require computation on the client side. To perform this, CIVIS relies on the JavaScript [Eich 2005] programming language. It uses jQuery to simplify a number of JavaScript tasks.

In addition to these web components, CIVIS relies on JavaScript Object Notation (JSON) [IETF 2014] to represent all its data. This notation is lightweight and can be used to interchange data between Python and JavaScript languages. In the system, it represents all cryptographic material such as keys (public and private) and NIZKPs as well as races, candidates, and all other information related to an election. Because election data belongs to a unique election, all JSON data includes an election ID (ElecID) and a description (Desc) fields. The former field identifies the election that the data belongs to. The latter describes for what the data is intended for. For instance, ”Desc”:”Credential” indicates that the data is a credential. The CIVIS system performs cryptographic computations on the server and on the client sides. Key generation, encryptions and decryptions, and zero-knowledge proofs computations, for instance, take place on the user web browser. Due to the characteristic of the current version that executes the whole system in a single machine, it establishes the underlying algebraic group on the server side and runs the mix net on the same server. In the sequel, we present CIVIS cryptographic components.

**Cryptosystems**. The current version of CIVIS implements the El Gamal cryptosystem [Gamal 1985] as well as its threshold version due to [Cramer et al. 1997]. In the threshold version, the El Gamal public key and its corresponding private key are cooperatively generated by $n$ parties, but the private key is shared among a set of $n$ parties. In order to decrypt a ciphertext, a minimal number of $t$ out of $n$ parties is necessary. Both El Gamal versions rely on a cyclic group $G$ of order $p$ where the Decision Diffie-Hellman problem (see [Boneh 1998] for details) is hard. To establish this group, the system randomly selects two primes $p, q$ such that $p = 2q + 1$. We could also adapt the system to use the elliptic curve variant of El Gamal. It generates two pair of keys. Let $g_1, g_3 \in \mathbb{Z}_p^*$ be two random generators and $x_1, y \in_R \mathbb{Z}_q^*$ be two random numbers. The first El Gamal public key is $T = \langle p, q, g_1, h_1 = g_1^{x_1} \rangle$ and its corresponding private key is $\widehat{T} = x_1$. The second public key is $R = \langle p, q, g_3, g_3^y \rangle$ and its corresponding private key is $\widehat{R} = y$. In the threshold version, the pair of keys ($T$,$\widehat{T}$) is generated such that $\widehat{T}$ is shared among the set of $n$ parties. The El Gamal cryptosystem is semantically secure [Goldwasser and Micali 1984], i.e., its ciphertexts do not reveal any information about the corresponding plaintexts. Also, it is homomorphic under multiplication. This way, let $E_T[m_1]$ and $E_T[m_2]$ be the El Gamal ciphertexts of the messages $m_1$ and $m_2$ using a public key $T$, $E_T[m_1] \cdot E_T[m_2] = E_T[m_1 \cdot m_2]$. Although the use of the threshold version is recommended, CIVIS allows the use of the conventional El Gamal version for flexibility. This way, it is possible to use the system in less restricted elections.

**Non-Interactive Zero-Knowledge Proofs (NIZKPs)**. The system implements a set of proofs of knowledge (PoK). In order to make ciphertexts plaintext aware (i.e. to prove knowledge of the underlying plaintext), it employs the proof of knowledge of a discrete logarithm due to [Schnorr 1991]. By means of this primitive, the party that makes the ciphertext proves that he is aware of what he is encrypting. The system also uses the proof of validity of an encrypted vote due to [Lee and Kim 2002]. It allows a party to prove that an encrypted vote belongs to the list of valid voting options. Besides these primitives, CIVIS implements the discrete logarithm equality test owing to [Chaum and Pedersen 1992]. It is used to ensure the correct key generation and that the ciphertexts are correctly decrypted. CIVIS implements the protocol for proving knowledge of a representation proposed by [Okamoto 1992]. It proves that, for two random group generators $g, h \in \mathbb{Z}_p$ (i.e. two bases), the representation of $z$ is $r, s \in_R \mathbb{Z}_q$ if $g^r h^s = z$. The system uses this proof during the vote generation. The system also implements a plaintext equivalence test. This primitive verifies whether two ciphertexts encrypt the same plaintexts or not without leaking any bit about the corresponding plaintexts. Particularly, here we verify a ciphertext encrypts the value 1 or not using the same technique. That is, for a ciphertext $C$ computed using a public key $T$, the system selects random number $z \in \mathbb{Z}_q$ and computes $C^z$. After decrypting $C^z$ using the private key $\widehat{T}$, the result is equal to 1 (if $C$ encrypted the value 1) or to a random number. CIVIS uses Fiat-Shamir heuristic [Fiat and Shamir 1986] to make these PoK non-interactive (NIZKPs).

**Universally Verifiable Mix net and the Web Bulletin Board**. A mix net is a cryptographic primitive that is composed of a set of servers (mixers). Each server performs by receiving as input a set of ciphertexts (e.g. encrypted votes) and outputs an anonymized set (i.e. by re-encrypting and permuting the ciphertexts of the original set). This way, the next mixer receives as input the output of the previous mixer. As long as

at least one mixer does not reveal its secrets (i.e. the random values used to re-encrypt the ciphertexts as well as the secret permutation) used to anonymize the set, no one can learn any correspondence between the messages on both sets (inputs and output). Because mixers may be malicious, for instance, by replacing any ciphertexts in the input set by a different one, CIVIS applies the scheme due to [Furukawa and Sako 2001] to allow verification of mix net work. That is, each mixer should prove that the ciphertexts on both sets (input and output) correspond to the same plaintexts without revealing any correspondence about them. As the proofs are public, anyone can verify them. In the current version, the mix net module contains the mix net primitive and it runs on the server side. Let $E_T[\cdot]'$ be an anonymized ciphertext. The bulletin board (WBB) is another important tool of the CIVIS system. It is a communication model that allows anyone to publish information. Once it receives a data, it makes the data public, and no one can delete or alter it. The current version of the system, however, models the bulletin board as a set of web pages along with a database by means of the WBB module. Through the WBB, anyone can verify public information about an election such as the encrypted votes.

### 3.3. Election Phases and Participants

CIVIS has a preliminary phase that creates an election. In addition, it has four phases that follow ABRTY's scheme. All phases are related to a module in the architecture. The setup is the phase that configures an election. In the registration phase, eligible voters receive valid credentials for voting. Voters cast their votes in the voting phase. The system identifies votes cast with valid credentials and tallies these votes in the tallying phase. Authorities, voters, and verifiers interact with the administration, the voting, and the bulletin board parts. The authorities create and manage the election by means of the administration part. The system has three types of authorities. The election authority (EA), the registration authority (RA), and the tallying authority (TA). EA is responsible for managing the election. It conducts the election and acts mainly in the preliminary phase. For instance, by enrolling eligible voters and the other authorities. RA issues credentials to the voters enrolled by EA and helps TA to identify these credentials in the tallying phase. Tallying votes and publishing the election results is the work of TA. For this, it identifies the valid credentials with the help of RA. The system allows to set one or more roles for an authority. In other words, the same entity can have the EA, RA, or TA roles, or one or two of them. As showed in Section 3.2, the system uses two El Gamal key pairs. TA computes her pair of El Gamal keys $T = \langle p, q, g_1, h_1 = g_1^{x_1} \rangle$ and $\widehat{T} = x_1$. RA computes her pair $R = \langle p, q, g_3, g_3^y \rangle$ and $\widehat{R} = y$.

### 3.4. Credential and Vote Formats

JCJ based schemes such as ABRTY use credentials to authorize votes from eligible voters while keeping their identities anonymous. For this, one (or more in cooperation for better security) authority issues a unique and exclusive credential to each voter. We refer to this credential as *valid*. The voter uses her valid credential to cast a vote that will appear in the final tally. In addition to valid credentials, these schemes also employ *fake credentials*. Different from the valid ones, fake credentials identify votes that will be removed from the tally and thus will not appear in the results. Voters under coercion use fake credentials to deceive adversaries. Both (valid and fake) credentials are indistinguishable to adversaries. This way, even if a voter transfers her valid credential to an adversary, the latter will not

be able to verify whether it is a valid one or a fake one. When under a coercive attack, the voter uses a fake credential instead of the valid one or give it to the adversary. Another defense against coercive attacks is the possibility to vote multiple times. That is, voters can vote more than once using the *same* credential (valid or fake). As a result, a voter can give a fake credential to an adversary and, when she is not under coercion, she can vote again using this time her valid credential. CIVIS computes a valid credential from a set of parameters via the registration module. A credential has a mathematical structure and it is composed of three values: $\langle A = (g_1 g_3^x)^{\frac{1}{y+r}}, r, x \rangle$, where $g_1, g_3 \in \mathbb{Z}_p^*$ are two random group generators, $r, x \in_R \mathbb{Z}_q^*$ are two random numbers, and $\widehat{R} = y$ is the RA secret key. After generating a valid credential, the system encodes it in JSON data format. Figure 2 shows an example of encoded credential. In order to deceive adversaries during coercive attacks, the voter needs a fake credential. In CIVIS, the voter generates a fake credential by replacing the value $x$ of her valid credential by a new randomly generated value $x'$. The system will discard votes corresponding to fake credentials.

Just as with credentials, CIVIS represents votes in JSON format. A vote is composed of two JSON data. The first one contains the ciphertexts related to the chosen candidates as well as data related to the credential. The second one is related to the NIZKPs. In order to encode the first JSON data, the system receives as input the chosen candidates as well as a credential in JSON format. From this credential, it randomly chooses a secret value $s$ and computes the parameters $\langle B = A^s, B^{s^{-1}}, B^{rs^{-1}}, g_3^x, o^x \rangle$, where $g_3, o \in \mathbb{Z}_p^*$ are group generators (see Section 3.2). The system then encrypts the candidate and the parameters except the value $B$ using TA public key, and encodes these values in JSON format. As a result, the first JSON data includes $\langle\ E_T[v], B, E_T[B^{s^{-1}}],\ E_T[B^{rs^{-1}}],\ E_T[g_3^x], o^x\ \rangle$, for a chosen candidate $v$. From the candidate and parameters, the system also computes the proof of knowledge of a discrete logarithm for each ciphertext, a proof of validity of an encrypted vote for the vote ciphertext and the proof of knowledge of a representation (see Section 3.2). The system encodes these NIZKPs in the second JSON data. In order to link the set of NIZKPs to the vote, the second JSON data includes a VoteID field. This field contains a cryptographic hash of the JSON that contains the vote (i.e. the first JSON data). The system makes the vote in the voter web browser.

### 3.5. Running an Election

In the sections above, we introduced the CIVIS components. As described, these components perform cryptographic operations on the voter web browser and on the server. We present next all steps necessary for running an election using the system.

**Creating an Election**. In order to run an election using CIVIS, the first step is to create it. For this, the system requires the registration of an election authority (EA). In the current version, EA registers directly in the system by informing her name, her e-mail address, and a password. Once registered, EA can log into the system. After a successful log in, EA inserts global information about the election. For this, EA enters the election title and an optional short text describing the election. She also selects whether the election will have a distributed tallying or not. By selecting it, the system will use the threshold El Gamal cryptosystem. Otherwise, it will use the conventional version of this cryptosystem. Then, EA enters the races and their corresponding candidates. After saving this data, EA enrolls the registration authority (RA) and the tallying authority (TA) as well as the eligible voters next by entering their names and their e-mail addresses. The

current system allows the registration of more than one TA, but it registers only one RA. These TAs forms the set of $n$ parties for the threshold El Gamal.

**Setup Phase**. After successfully logging into the system, RA and TA access the election map (Figure 3 exemplifies part of the election map). By means of this map, RA and TA generate their cryptographic material. In other words, the system computes the elements that define the secure cyclic group, that is, the primes $p$, $q$, and the generator $g$. From this group, if EA selected distributed tallying before, then the set of $n$ TAs in cooperation runs threshold ElGamal cryptosystem to generate a key pair, where the private key is shared among them. Otherwise, the system uses a single TA and this authority generates her conventional El Gamal key pair. RA also generates her pair of keys. As next steps, EA computes the election generator number $o$. The value $o \in \mathbb{Z}_p^*$ is a random group generator and it will be used during the vote generation later on. To finish this phase, EA publishes all public election data on WBB. This includes general election data, authorities' data, races and candidates' data, and public keys. All data is JSON encoded.

**Registration Phase**. In this phase, all eligible voters receive their valid credentials. Ideally, voters have to receive their valid credentials from RA in secrecy. To accomplish this, the interaction between each voter and this authority takes place through a untappable channel. In practice, because untappable channels are difficult to accomplish, this channel is replaced by a registration place. That is, a secure place (e.g. the City hall for a political election) that the voter should visit to obtain her credential after proving that she is eligible. Although CIVIS can be adjusted to issue valid credentials in a secure place, this would limit its use as it would not be possible to use the entirely system via Internet, in a way similar to the Helios system. To allow this, the system can send valid credentials by e-mail. On one hand, because e-mail is an insecure channel, this undermines the coercion-resistance of the system. A coercer could easily obtain a valid credential by eavesdropping this channel. On the other hand, the submission of valid credentials by e-mail allows the use of the system in simple scenarios and to perform test elections without requiring voters to visit registration places. In order to generate a valid credential, EA enters her private key and the system generates a valid credential (see Section 3.4).

**Voting Phase**. In order to vote, the voter first visits the system and selects the election that she wants to vote. There is no previous authentication to access an election. After that, the system presents to the voter the races and their corresponding candidates. After selecting her candidates, the voter enters her credential in JSON format as illustrated in Figure 2. After that, the system computes the two parts that compose the vote (see Section 3.4) and encodes them. To finish, the system asks the voter to confirm her vote. After receiving the confirmation, the system publishes it on WBB. Upon publishing the vote on WBB, the system verifies whether its corresponding NIZKPs proofs are all valid. If one of the proofs is invalid, the system invalidates the vote. Otherwise, it identifies the vote as well formed on WBB.

**Tallying Phase**. After the voting period ends, the authorities can begin to tally the votes posted on WBB. For this, TA and RA access the election map as illustrated in Figure 3. From all votes (i.e. the first part of the vote) that passed the NIZKPs verification, TA firstly instructs the system to remove duplicated votes (step 8), that is, votes posted with the same credential. The system identifies these votes by comparing all values $o^x$ on the first part of the vote. Indeed, votes posted using the same credential will

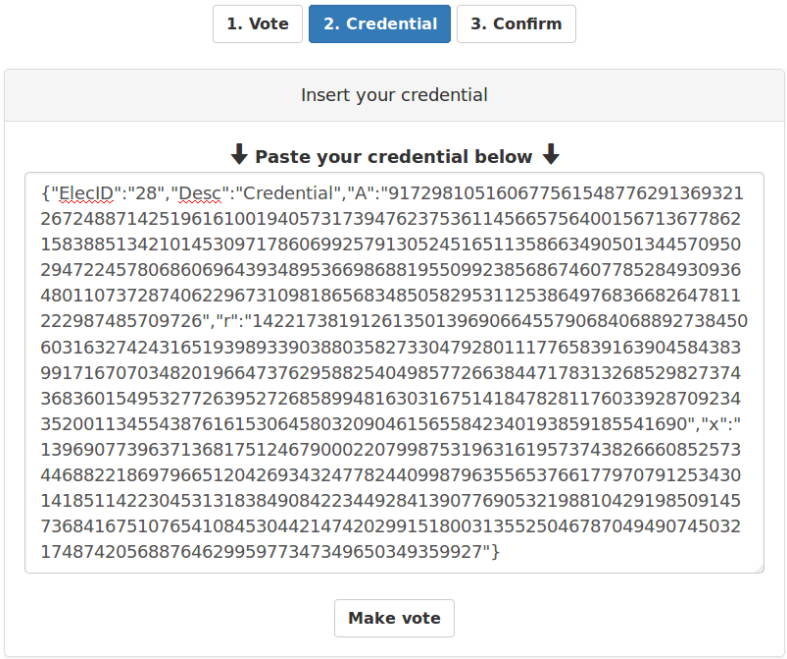


**Figura 2. Voter enters her JSON encoded credential into the system.**

all share the same value $o^x$. From these votes, the system removes all but the last posted one based on the data/time of posting on WBB. After that, TA instructs the system to send all remaining votes (non-duplicated votes and the last posted duplicated votes) to the verifiable mix net (step 9). This starts the verifiable mix net on the server (see Section 3.2). After anonymizing the votes, the mix net publishes them on WBB as well as the proofs that it behaved correctly. From the anonymized votes, RA and TA begin the identification of the votes posted with valid credentials. For this, RA enters her private key $\widehat{R} = y$ and the system computes $E_T[B^{s^{-1}}]'^y = E_T[B^{ys^{-1}}]'$ and $E_T[B^{ys^{-1}}]' \cdot E_T[B^{rs^{-1}}]' = E_T[B^{ys^{-1}+rs^{-1}}]'$ (step 10). The system publishes $E_T[B^{ys^{-1}+rs^{-1}}]'$ on WBB. TA now enters her private key $\widehat{T}$ and reads the values $E_T[B^{ys^{-1}+rs^{-1}}]'$ from WBB. She then computes $C = E_T[B^{ys^{-1}+rs^{-1}} g_1^{-1} g_3^{-x}]'$ from $E_T[B^{ys^{-1}+rs^{-1}}]'$, $E_T[g_3^x]'$, and the public group generator $g_1$. Still in this step, the system next runs the Plaintext Equivalence Test (see Section 3.2) to verify whether $C$ is an encryption of the plaintext 1 or not (step 11). If the resulting plaintext is equal to the value 1, then $E_T[v]$ corresponds to a valid credential. The system then decrypts all $E_T[v]$ related to the valid credentials (step 12), count the votes and publishes the results on WBB. Again, all computations are performed locally on TA and on RA web browsers.

## 4. Election Simulation

In order to show the applicability of CIVIS, we present below the results of a first small-scale election that employed this system. The simulation was carried out in a university environment. We invited and registered a total of 200 eligible voters between professors and students of a computer science department. The election was created with two races. One for choosing the student union president and the other for choosing the college director. Each race contained three options. Because the election was configured with an EA authority that also performs the roles of RA and of TA, it used the conventional El Gamal

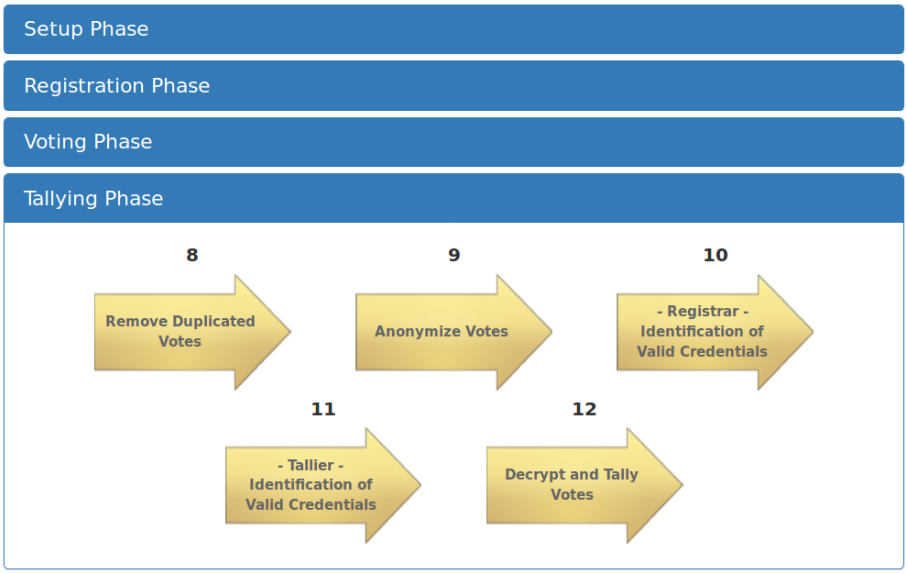


**Figura 3. Steps of the talling phase on the Election map.**

cryptosystem. The system generated 1024-bits El Gamal keys and it was installed on a Linux virtual machine (VM). Eligible voters received their valid credentials via e-mail address. Because we were carrying out a test election, we expected that most of the voters would not visit a registration place to receive their valid credentials. Thus, we decided to send credentials by e-mail addresses. Notice that we used this just for test purposes.

After configuring the election and registering the eligible voters, EA started the voting phase and it was available for a week. During this period, the system received 202 votes. When the voting period ended, TA started the tallying. From 202 votes, 8 votes were discarded due to invalid NIZKPs. For test purposes, the CIVIS team submitted these discarded votes using different fake credentials. Each credential were generated by selecting random values for $A'$, $r'$, and $x'$. Then, the system evaluated 194 votes to identify duplicated votes. The running time for this was about 1.38 seconds and 177 votes remained for the next step. The system sent 177 votes to the mix net. The running time to anonymize these votes and to generate the proofs was 32.9 seconds. Next, the system began the identification of the valid credentials. After the RA entered her secret key, the system processed 177 anonymized votes in 8 seconds. It ran for 13 seconds after TA entered its secret key. Thus, the total running time to identify valid credentials from 177 votes was 21 seconds. During this process, the system identified 148 votes cast with valid credentials and 29 votes cast with fake credentials. After that, TA decrypted the votes from the valid credentials and published the results on WBB.

## 5. Conclusion and Future Work

The design of a secure system begins by the design of a secure protocol. If the protocol is insecure, even a secure implementation of this protocol may fail to ensure a secure election. Because there are few documents about SIGEleição publicly available, we only sketched an analysis of it. This analysis showed that, to satisfy the secure properties, the election authority and the system itself must be trustworthy. These are very strong

assumptions to ensure security in real word elections. Coercion is intrinsic problem of Internet elections that many election systems ignore or do not treat properly. In this context, we introduced CIVIS, an election system that fights coercive attacks. CIVIS is based on one of the most efficient coercion-resistant protocols available. It puts a theoretical scheme into action and it allows carrying out elections using advanced technology against coercion problems. We also presented the first results of a simulated election carried out using CIVIS. In this test, the system sent voters credentials by e-mail. We stress that the e-mail is an insecure channel for sending valid credentials. However, the test allowed users to experience the system and presented results about the system performance. Although the current version of CIVIS is functional, it still requires improvements and more tests. It does not include a distributed credential generation. Also, it needs improvements related to its usability. For instance, voters need an easy way to generate fake credentials. We are currently working on these features.

## References


Abdellatif, T. and Adouani, A. (2014). Mosaic: A secure and practical remote voting system. *Int. J. Autonomic Comput.*, 2(1):1–20.

Adida, B. (2008). Helios: Web-based open-audit voting. In van Oorschot, P. C., editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348. USENIX Association.

Araújo, R., Rajeb, N. B., Robbana, R., Traoré, J., and Yousfi, S. (2010). Towards practical and secure coercion-resistant electronic elections. In Heng, S., Wright, R. N., and Goi, B., editors, *Cryptology and Network Security - 9th International Conference, CANS 2010, Kuala Lumpur, Malaysia, December 12-14, 2010. Proceedings*, volume 6467 of *Lecture Notes in Computer Science*, pages 278–297. Springer.

Benaloh, J. (2006). Simple verifiable elections. In Wallach, D. S. and Rivest, R. L., editors, *2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06, Vancouver, BC, Canada, August 1, 2006*. USENIX Association.

Boneh, D. (1998). The decision diffie-hellman problem. In Buhler, J., editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer.

Brickell, E. F., editor (1993). *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*. Springer.

Chaum, D. and Pedersen, T. P. (1992). Wallet databases with observers. In [Brickell 1993], pages 89–105.

Clarkson, M. R., Chong, S., and Myers, A. C. (2008). Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 354–368. IEEE Computer Society.

Cramer, R., Gennaro, R., and Schoenmakers, B. (1997). A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5):481–490.

Delaune, S., Kremer, S., and Ryan, M. (2010). Verifying privacy-type properties of electronic voting protocols: A taster. In Chaum, D., Jakobsson, M., Rivest, R. L., Ryan, P. Y. A., Benaloh, J., Kutylowski, M., and Adida, B., editors, *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 289–309. Springer.

DSF, D. S. F. (2017). Django - The web framework for perfectionists with deadlines. `https://www.djangoproject.com/`. Access March 2018.

Eich, B. (2005). Javascript at ten years. In Danvy, O. and Pierce, B. C., editors, *Proceedings of the 10th ACM SIGPLAN International Conference on Functional Programming, ICFP 2005, Tallinn, Estonia, September 26-28, 2005*, page 129. ACM.

Fiat, A. and Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In Odlyzko, A. M., editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer.

Furukawa, J. and Sako, K. (2001). An efficient scheme for proving a shuffle. In Kilian, J., editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer.

Gamal, T. E. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472.

Goldwasser, S. and Micali, S. (1984). Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299.

IETF (2014). RFC 7159 - The Javascript Object Notation (JSON) Data Interchange Format. `https://tools.ietf.org/html/rfc7159`. Access: March 2018.

Juels, A., Catalano, D., and Jakobsson, M. (2005). Coercion-resistant electronic elections. In Atluri, V., di Vimercati, S. D. C., and Dingledine, R., editors, *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005*, pages 61–70. ACM.

Langer, L., Schmidt, A., Buchmann, J. A., Volkamer, M., and Stolfik, A. (2009). Towards a framework on the security requirements for electronic voting protocols. In *First International Workshop on Requirements Engineering for e-Voting Systems, RE-VOTE 2009, Atlanta, Georgia, USA, August 31, 2009*, pages 61–68. IEEE Computer Society.

Lee, B. and Kim, K. (2002). Receipt-free electronic voting scheme with a tamper-resistant randomizer. In Lee, P. J. and Lim, C. H., editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 389–406. Springer.

Okamoto, T. (1992). Provably secure and practical identification schemes and corresponding signature schemes. In [Brickell 1993], pages 31–53.

Participants of the 2007 Dagstuhl Conference on Frontiers of E-Voting (2007). Dagstuhl Accord. `http://www.dagstuhlaccord.org/`. Access: March 2018.

PSF, P. S. F. (2018). Python language reference. `http://www.python.org/`. Access: March 2018.

Santos, J., Lins, C., and Madruga, M. (2017). SIGEleição - Um Novo Jeito Seguro de Votar. Workshop de Tecnologia da Informação e Comunicação das Instituições Federais de Ensino Superior do Brasil (WTICIFES).

Schnorr, C. (1991). Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174.

Serdult, U., Germann, M., Mendez, F., Portenier, A., and Wellig, C. (2015). Fifteen years of internet voting in switzerland: History, governance and use. In Terán, L. and Meier, A., editors, *2015 Second International Conference on eDemocracy & eGovernment (ICEDEG)*, Quito, Ecuador. IEEE.

The National Election Committee (2017). Internet Voting in Estonia. `http://www.valimised.ee/en/internet-voting/internet-voting-estonia`. Access: March 2018.