An efficient end to end verifiable voting system

Léonie TAMO MAMTIO[1] **and Gilbert TINDO**[1]

[1]Department of Computer Sciences,University Of Yaoundé I, Cameroon

*E-mail : tamoleonie@gmail.com

DOI : 10.46298/arima.6442
Submitted on 30 Avril 2020 - Published on 23 Juin 2021
Volume : **33** - Year : **2020**
Special Issue : **CRI 2019**
Editors : *Eric Badouel, Paulin Melatagia Yonta, René Ndoundam, Maurice Tchuenté*

Abstract

Electronic voting systems have become a powerful technology for the improvement of democracy by reducing the cost of elections, increasing voter turn-out and even allowing voters to directly check the entire electoral process. End-to-end (E2E) verifiability has been widely identified as a critical property for the adoption of such voting systems for electoral procedures. Moreover, one of the pillars of any vote, apart from the secret of the vote and the integrity of the result, lies in the transparency of the process, the possibility for the voters "to understand the underlying system" without resorting to the competences techniques. The end-to-end verifiable electronic voting systems proposed in the literature do not always guarantee it because they require additional configuration hypotheses, for example the existence of a trusted third party as a random source or the existence of a random beacon. Hence, building a reliable verifiable end-to-end voting system offering confidentiality and integrity remains an open research problem. In this work, we are presenting a new verifiable end-to-end electronic voting system requiring only the existence of a coherent voting board, fault-tolerant, which stores all election-related information and allows any party as well as voters to read and verify the entire election process. The property of our system is information guaranteed given the existence of the bulletin board, the involvement of the voters and the political parties in the process. This involvement does not compromise the confidentiality nor the integrity of the elections and does not require cryptographic operations on the voters account.

Keywords

Electronic voting, End-to-End verifiability, Confidentiality, Integrity, Hash Function

---

I INTRODUCTION

In an end-to-end verifiable electronic system (E2E), voters have the ability to verify that their vote has been properly emitted, recorded and counted in the election result. Intuitively, the security property this system must provide is the ability of voters to detect fraud in the electoral process.

End-to-end verification requires the voter to be able to obtain a receipt at the end of the vote that will allow him to verify that his vote has been cast as expected, recorded as such and counted in the results. In addition, any external third party should be able to verify that the election procedure has been carried out correctly. Indeed, it is imperative that receipts from an E2E system be delegable, that is, that the voter can outsource the verification task to any interested third party, for example an independent, confident organization that performs a global check. This requirement, as well as the fact that it would be impossible for the voter to use his receipt as evidence of who he voted for (to avoid buying votes) make the design of verifiable end to end a difficult problem.

The well-known e-voting systems that offers end-to-end verifiability (examples: Remotegrity [19], EVIV [18]) generally ensure this only under certain configuration assumptions, such as the existence of a trusted third party for key generation and as a source of chance, a model or machine for generating random values [21]. Indeed a limitation to the use of certain hypotheses to ensure end-to-end verifiability is the fact that one must have faith in that and therefore to the result of the elections. This causes a problem because it is not easy for electoral authorities to unequivocally convince voters that the election is correct. In addition, the ability of voters to easily understand the different components of the system gives rise to controversies. The results of the election can be the subject of several contestations.

Motivated by the foregoing, we are propose a new verifiable end-to-end electronic voting system that only requires the existence of a consistent voting bulletin board that provides a comprehensive and coherent view of the election. In addition, the proposed system does not require any cryptographic knowledge on the voter's side. End-to-end verification can be achieved through the use of transparent procedures, the voting bulletin board and the involvement of the different political parties concerned by the election.

In the rest of this work, section 2 presents a state of art of the verifiable end-to-end systems. Section 3 highlights the new end-to-end verifiable voting system. An evaluation of this system is proposed in section 4. Section 5 concludes our work and presents the prospects for improvement.

II RELATED WORK

Electronic devices have become an indispensable part of our life with the evolution of computing and technology. Voting systems also benefit from these technological developments. Indeed, votes are frequent; it is therefore necessary to ensure that the voting procedure is pleasant and non-binding. Electronic voting provides many benefits to both voters and administrations, including speed, accessibility, accuracy, convenience, flexibility and mobility. The argument generally used for electronic voting is that it reduces the cost of elections, gives more accurate results, speeds up the counting process, modernizes the electoral system, offers a wider time slot during which voters can cast their votes and improves voter turnout.

In view of all these advantages, several works have been carried out with the aim of proposing secure electronic voting systems. It follows that an electronic voting system must meet the following requirements [2]:

1. Integrity: it is impossible to modify, add or delete a vote;

2. Democracy: only authorized persons can vote once and only once.

3. Confidentiality: the vote and the voter are secret;

4. Individual verifiability: a voter checks his vote in the count;

5. Universal verifiability: the published results are verifiable by all;

6. Partial Mobility: the system only takes into account the votes of the authorized offices;

7. Total Mobility: The system takes into account votes cast anywhere.

8. Availability: the system works correctly throughout the voting period;

9. Robustness (protocol and system): the system can satisfy the voting properties even under abnormal operating conditions.

The confidentiality, integrity, democracy and verifiability of elections are the main research objectives in the field of electronic voting, which has been in existence for some decades. The voting protocols resulting from this work are classified into three broad categories according to the technique used: mix-nets [14, 15], blind signatures [1, 3] and homomorphic voting systems [8, 15]. All of these protocols are based on heavy cryptographic components and require cryptographic skills for voters and political parties. As a result, in 2004, Chaum [4] and Neff [5] identified that the verifiability offered by voting systems relies too much on cryptography to be of practical use to voters. From these works emerges a new paradigm of research in electronic voting: E2E voting systems. The goal of end-to-end verifiable systems is to provide the opportunity for any voter to easily check his or her vote and that all the results are verifiable universally. The system proposed by Chaum [4] was the base of many end-to-end verifiable systems such as: Prêt à Voter [6], Punchscan [7], Scantegrity [11], Aperio [12], Eperio [13]. These systems are exposed to many problems: complexity, confidentiality, and usability. The verifiability is still argued but not assured.

Helios [10] was the first used verifiable end-to-end voting system. It targets the low risk elections by implementing existing ideas into a system. Helios uses a simplified version of the Benaloh challenge [9](considered acceptable for a low risk coercion environment) to achieve verifiability. However, it is exposed to several attacks related to privacy due to the malleability of the cryptographic scheme used [17]. As a result, other systems have emerged to truly achieve end-to-end verifiability.

This is the case for the Remotegrity [19], based on a lock code that is provided to the voter on a scratchable surface to allow him to check his vote, detect and prove unauthorized changes to their ballots made either by malicious client software or a corrupt election authority. In addition, Remotegrity is designed for a specific municipal election: Takoma Park, Maryland responds to a certain number of requirements and is an extension of the existing Scantegrity system [11], which constrains its conception. We realise that, the number of codes an voter must enter during the voting process is high. This number could be revised downward. For example, the serial numbers of the ballot and the authorization card can be harmonized to the same value.

At the same time, EVIV [18] has been developed with the objective of offering complete mobility to the voter and ensuring end-to-end verifiability while preserving confidentiality. In other to solve the problem of malicious software and authority, EVIV combines the voting code to the encryption technique MarkPledge [16]. For the verification, the voter must match alphanumeric strings that detect and protect against voting manipulations on both the unstable voting client platform and the election server. In EVIV, each voter has a Voter Security Token (VST), which is responsible for encrypting the vote and to which the voter communicates his selection of candidates. With the help of VST, each voter generates the voting codes at home, which facilitates the logistics of the election and allows a complete voting process online and on mobile. However, the limited computing capabilities of VST limit the use of EVIV in elections with a small number of candidates. EVIV nonetheless requires the integration of coercion resistance mechanisms and improved usability and verifiability mechanism.

Rabin and Rivest [20] proposes an end-to-end verifiable voting system based on the distribution of vote count and the creation of a verifiable proof of the exactitude of server combined with a random representation of the integers used in the system. At the end of the count, a random permutation of the different recorded votes is published without revealing the identity of the voters. However, the system requires mechanisms of resistance to constraints and turns out to be complex.

Kiayias et Al. [21] finds that all known e-voting systems that offer end-to-end verifiability can only achieve this under configuration assumptions such as the existence of a trusting party that provides random values or of a machine as a random source. In fact, this verifiability can be argued, but not formally proven due to the fact that a leap of faith will be required in order to accept the setup assumption and thus the election result. For this purpose, the authors propose a verifiable end-to-end system without hypothesis except for the existence of a bulletin board. However, the system is based on the ElGamal cryptosystem with the Decisional Diffie-Hellman (DDH) basic hypothesis, which is a difficult cryptographic problem; its commitment regime is homomorphic and uses entropy that is generated by the interaction between voters and the system.

In sum, the end-to-end verifiable voting systems proposed to this day allow voters to effectively check their votes. But however they remain based on a set of assumptions and cryptographic components that are sometimes heavy. What remains a problem in the process of verifiability of the voting system, namely: the ability for the political parties and voters concerned by the election to understand the voting system that is proposed to them and to be involved. This condition is necessary to increase verifiability and further enhance the reliability of the system. Hence the purpose of the protocol we propose.

III END-TO-END VERIFIABILITY PROTOCOL

In this section, we present different actors involved in the electoral process and the different phases of the voting process ranging from the enrollment of voters to the publication of the results.

Throughout the process, we will use the following notations:

E: set of registered voters

$I = I_1, \ldots, I_{N_2}$ the set consisting of all the authentication information of the registered voters such that card (E) = card (I) = q,

N_1 the number of individuals able to vote,

N_2 individuals actually registered

I_j : set of information about an individual j,

R the set of random values provided by the voters

v all lock codes

V all vote casts

Id_{Office} : all the identifiers of the polls stations

B: all the ballot papers

$Y = Y_1, ...Y_{N_2}$: the vote database

The hash functions used during the voting process are SHA-256 functions.

RSA is used for asymmetric encryption/decryption and the key size is 1024 bits.

3.1 The entities of the system

The system comprises the following different entities:

Enrollment service: entity responsible for registering voters on voter list, drawing voter registration cards, candidate databases and voter lists before election day.

Election authentication: handles voter authentication on election day. This service is the responsibility of the Central Authority for Legitimation (**CAL**).

Voting booth: electronic entity used to store certain voting data.

Central Accounting Authority (CAA): entity in charge of counting of the ballot papers, the counting of votes by candidate and the publication of the results. The counting of votes is done centrally.

Bulletin Board: entity responsible for the publication of the public data of the vote and the results of the ballot.

Voting service : it includes a software unit for voting, a ballot validation unit and a voting data transfer unit in random order to the CAA.

Verification service: it is an entity responsible for the verification and validation of data and voting results. **Voters**, **Political parties**, and **Independent verification organizations** can run each instance of this service.

The architecture of the system and voting sequence can be summarized as described respectively in Figure 1 and Figure 2.

3.2 Voter enrollment phase

Months before the poll, the organization responsible for managing the elections organizes the registration of voters on the voter lists for a specific period. The entity in charge of registration (Enrollment service) is set up and any citizen of voting age can register. This phase is carried out through the following steps:

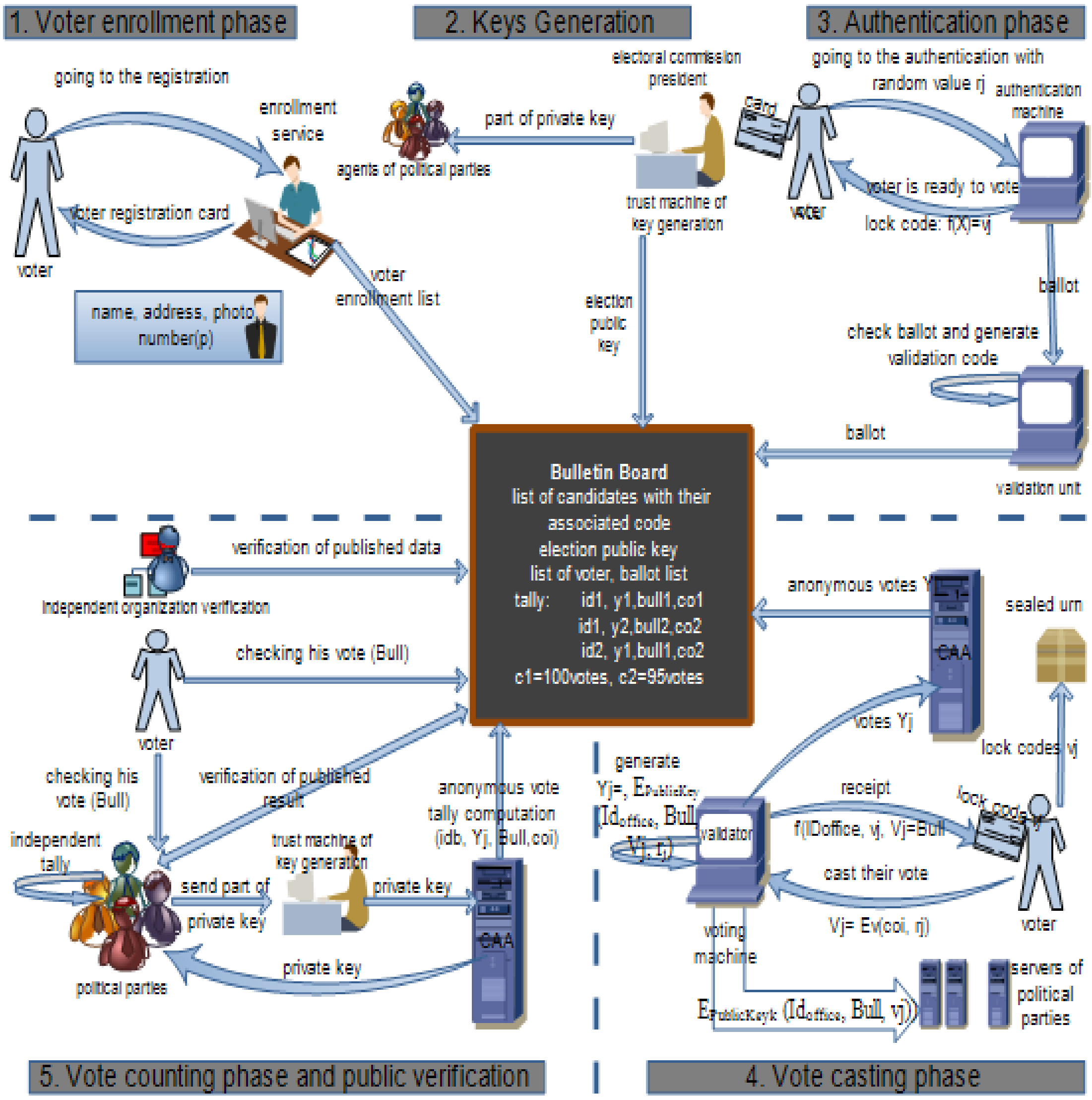


Figure 1: Architecture Example

3.2.1 *Voter to Enrollment service*

At this stage, the voter provides the Enrollment service with all the information necessary for its identification with the aim of obtaining a voter's card. For this purpose, he presents himself at a registration station equipped with a piece of civil status (national identity card or birth certificate) allowing identification. The following information concerning to the voter: first and last names, date and place of birth, father's name, mother's name, (national identity card number. This information is unique for each voter, that is, there cannot be two individuals for whom all of this information is identical. Formally,

$\forall j \in 1,\dots,N_2, I_j = i_j^1, i_j^2, \dots, i_j^k$ and $\forall l, j \in 1,\dots,N_2, \exists x \in 1,\dots,k, l \neq j, i_l^x \neq i_j^x$, where I_j is all the information of voter j and i_j^k is the k^{th} information of voter j.

3.2.2 *Enrollment service to Voter*

The recording machine generates 4 disposable masks which will be used to encode the personal information of the voter that will be used for authentication before storage in the database. The choice of 4 masks is done in order to further reinforce the secrecy around the voter information

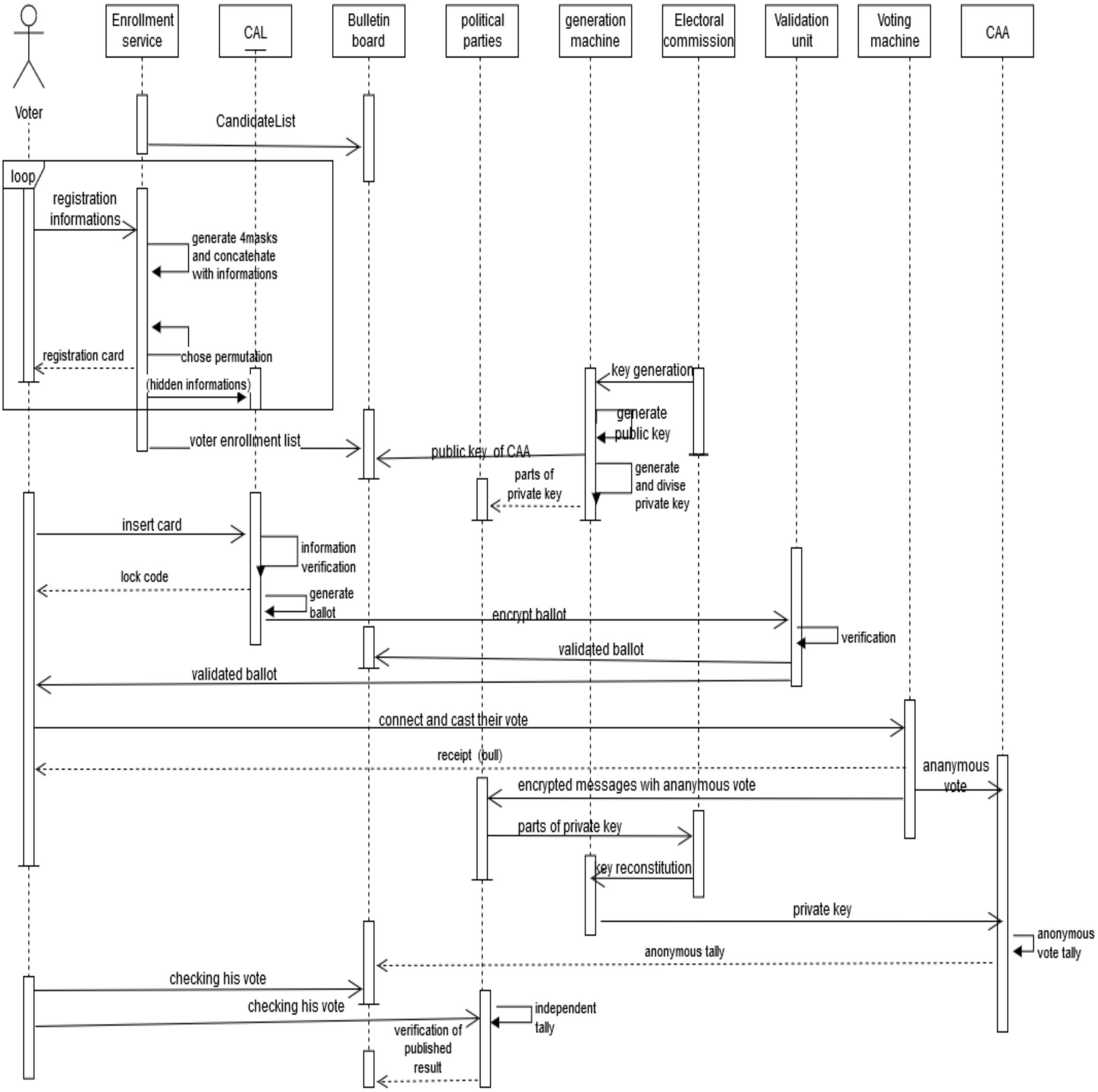


Figure 2: Voting interactions

and make the decoding task more difficult. After generating the masks, the information is coded with the masks as follows: Let be:

n the number of bytes necessary to code all the information (I_j) of a voter j,

I_{jy} is the part of the information I_j contained in the byte y, with $0 \leq y \leq n-1, I_j = I_{j0}, \ldots, I_{j(n-1)}$

M_1, M_2, M_3, M_4 the 4 generated masks, with $M_q = m_q^0, m_q^1, \ldots, m_q^{(n-1)}, 1 \leq q \leq 4$

We carry out an X-OR between I_j and M_1 the result is carried out one X-OR with M_2 and so on. The result obtained is $I_j \oplus M_1 \oplus M_2 \oplus M_3 \oplus M_4$.

Then, we perform permutations according to the permutation number chosen among the $n!$ possibilities on the result $I_j \oplus M_1 \oplus M_2 \oplus M_3 \oplus M_4$. The number of the permutation is number (p), $0 \leq number(p) \leq n! - 1$. In order to make the detection task of the disposable mask more complex, number $(p) \geq 4$. number (p) is associated with the voter's information and put on the voter's card. The masked information of the voter and the different masks are encrypted with the public key of the authentication machine (CAL).

At the end of the registration process, the enrollment service performs the redundancy check in order to eliminate duplicates. In the case of duplicates, that is to say that an voter has been registered twice, the most recent registration is retained.

3.3 Key generation

Throughout the process, several encryption / decryption keys are manipulated. The CAL and the ballot validation unit has each other a key pair to encrypt/decrypt the information from the record. Each political party k has a key pair ($Publickey_k$/$Privatekey_k$) to encrypt/decrypt the voting data that will be transmitted to it. In order to achieve end-to-end verification, we involve the different political parties in the election process. Each party must have a function (action) of the key for decrypting the ballots and can perform the count independently. The election management organization selects a brand and model of machines to use for keys generation. One month before the election, political parties and the media are invited to the key generation event. Each party sends an expert to attest the generation machines and an officer. Once the experts have certified the machines, the public key of the CAA is generated, signed and published. The corresponding private key is generated and then divided into pieces according to Shamir's secret-sharing technique as indicated by algorithm 1 below and according to the number of political parties involved in the election. Each party receives a function from the key. The threshold (k, n), k less than or equal to n which represents the number of political parties, is fixed for the reconstitution of the key. From k parts we can find the key. k = n implies that all parts are needed to find the key.

Algorithm III.1 Generation & SharingKey

```
Input: $c_1, c_2, ..., c_n$; (k,n) the threshold chosen for sharing keys.
Output: $p_1, p_2, ..., p_n$
$Pairkey \leftarrow$ generationPairkey ( );
$Publickey \leftarrow$ Pairkey.getPublickey ( );
$Privatekey \leftarrow$ Pairkey.getPrivatekey ( );
// It takes k points to define a polynomial of degree k−1
Generate randomly k−1 coefficients $r_1, r_2, ..., r_{k-1}$;
Let $r_0 = Privatekey$;
Build the polynomial $f(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{k-1} x^{k-1}$;
for $j \leftarrow 1$ to $n$
    Build the share $p_j = (x_j = j, y_j = f(j))$, where $p_j$ (a pair of antecedent and the corresponding
     image by the polynomial function) is the $j^{th}$ part of the private key $Privatekey$;
    Give the share $p_j$ to the agent of the $j^{th}$ party.
Given a subset k of these pairs $p_j$, the polynomial interpolation makes it possible to find
 the coefficients of the polynomial whose constant term is the secret $Privatekey$;
```

3.4 Authentication phase

Voter authentication on election day is done by the CAL. The voter presents himself with his voter's card which he introduces into the authentication machine. The machine retrieves number (p) and calculates the inverse permutation to find the permutation performed on the hidden information. The CAL decrypts the registration information. Then it verifies the voter's identity from the database and in case of compliance, a unique lock code is generated for the voter as proof that the authentication has succeeded. The voter information is unmasked once and gets X= $I_j \oplus M_1 \oplus M_2 \oplus M_3$.

Let f be the authentication function. f is a collision-resistant one-way hash function that provides for each voter its unique lock code, f(X)= v_j. The triplet (id_{Office},X,v_j) is sent to the electronic voting booth. The voting booth holds the function f. It retrieves X and calculates f(X) and verifies that f(X) = v_j. In case of equality, the machine checks in the set of stored triplets that it does not already exist this v_j. If this is the case the v_j is validated and the triple (id_{Office},X,v_j) is registered in the voting booth. v_j is given to the voter and posted on the voting bulletin board. In the case where v_j is already contained in the voting booth, access to the voting machine is refused to the voter because that supposes that he has already cast his vote.

3.5 Vote casting phase

This phase gathers all the steps of the actual voting from the creation of the ballot to the receipt marking the proof of registration of the vote of an voter.

3.5.1 Creation of the ballot

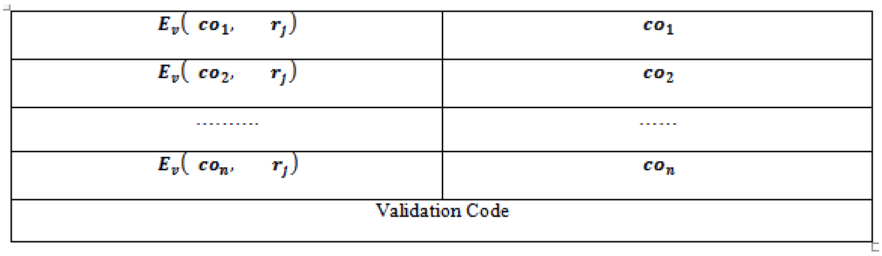
Once the authentication is complete, the voter goes to the voting machine and accesses it via his code. Each election candidate has a unique code to identify himself and will be used at the polls by voters to make their choice. Formally, let C be the set of candidates and C_o be the set of candidate codes such that card (C) = card (C_o) = n. The bijection h which associates a unique code to each candidate is defined as follows: $h : \; C \rightarrow C_o$

$$c_i \mapsto h(c_i) = co_i$$

The voter provides a random value r_j. This value is concatenated to each of the candidate codes and then encrypted with the public key of the ballot validation authority to produce a single ballot for that voter. Let E_v be the encryption function,

$$\begin{aligned} E_v : \; & C_0 \times ... \times C_0 \times R \rightarrow \sharp C_o \times ... \times \sharp C_0 \\ & (c_{o1}, ..., c_{on}, r_j) \mapsto E_v(c_{o1}, ..., c_{on}, r_j) = (\sharp r_j c_{o1}, ..., \sharp r_j c_{on}) \end{aligned}$$

The resulting ballot is of the form described in Figure 2.



$E_v(co_1, \quad r_j)$	co_1
$E_v(co_2, \quad r_j)$	co_2
....	......
$E_v(co_n, \quad r_j)$	co_n
Validation Code	

Figure 3: Ballot Example.

At this stage, the validation code is empty. He will be informed by the ballot validation unit of the ballot papers to certify that the ballot is correct. This built ballot is sent to the validation unit of the ballots for validation.

3.5.2 Validation of the ballot

Upon receipt of the ballot created, the ballot validation unit, decrypts with its private key, all the encrypted content in the bulletin and verifies that c_{oi} contained in each encrypted corresponds to that provided at the considered line and that it is a valid candidate code. In case of compliance, it validates the ballot by generating a unique validation code for it. The validated ballot is stored in the voting database (for verification) and sent to the voter.

3.5.3 Voting stage

To vote, the voter chooses the encryption corresponding to the code of the candidate for which he wishes to vote and builds his unique vote V_j= E_v (c_{oi},r_j).

$\forall j \in 1,\dots,N_2, \exists! i \in 1,\dots,n | V_j = \sharp r_j c_{oi}$

Let f' be the voting function, f' is an irreversible hash function which for a given voter recovers his choice and produces a single ballot corresponding to his vote and his Bull receipt,

$$\begin{aligned} f' :\ & Id_{Office} \times v \times V \to B \\ & (id_{Office}, v_j, V_j) \mapsto f'(id_{Office}, v_j, V_j) = Bull \end{aligned}$$

Bull is given to the voter as proof of validation of his vote and will be used to verify his vote in the final count. After each vote, the voting unit builds a message consisting of Bull, v_j and id_{Office}, office identifier that is encrypted for each political party k with its public key ($Publickey_k$) and sent to the headquarters of said party. Let $E_{PublicKeyk}$ this encryption function,

$$\begin{aligned} E_{PublicKeyk} :\ & Id_{Office} \times B \times v \to \sharp B \\ & (id_{Office}, Bull, v_j) \mapsto E_{PublicKeyk}(id_{Office}, Bull, v_j) = \sharp Bull \end{aligned}$$

Each party also has the function f'.

The voting unit also builds for each vote made by an voter a message consisting of Bull, V_j, r_j and id_{Office} that it encrypts with the public key (PublicKey) of the CAA and gets the encrypted Y_j. Let $E_{PublicKey}$ this encryption function,

$$\begin{aligned} E_{PublicKey} :\ & Id_{Office} \times B \times V \times R \to Y \\ & (id_{Office}, Bull, V_j, r_j) \mapsto E_{PublicKey}(id_{Office}, Bull, V_j, r_j) = Y_j \end{aligned}$$

The $Y_j s$ are transmitted by the voting data transfer unit to the CAA via a secure tunnel. The deciphering of the different Y_j will make it possible to count votes by candidates and obtain the results of the election.

3.6 Vote counting phase and public verification

At the end of the elections, the CAA has the Y_j vote database. However, the counting requires the use of the CAA private key Privatekey. This requires a step of reconstituting this key before the tally. In addition, each political party counts independently for the purpose of verifying the concordance of results published by the CAA.

3.6.1 Key reconstitution

The election management organization organizes the event to reconstitute the private key and summons all the actors, including the different political parties. Each party sends an officer with the function of the key held by the political party and an expert for possible controls. The different parts are recovered, introduced into the machine and reassembly performed according to the Shamir decoding principle described in algorithm 2 below. The reconstituted private key is given to the CAA and the different political parties (to perform the count).

Algorithm III.2 KeyReconstitution

```
Input: $p_1, p_2,..., p_n$; (k,n) the threshold chosen for sharing keys.
Output: $Privatekey$
For each political party $j \leftarrow 1$ to $n$
    recover $p_j$ by the CAA via secure communication;
Use k sum of the parts $(p_1, p_2,..., p_n)$, interpolate with Lagrange contained in the Shamir
  scheme to generate the polynomial $f(x)$;
The associated Lagrange polynomial is written : $f(x) = \sum_{j=0}^{k-1} y_j l_j(x)$
  where $l_j$ are the basic polynomials of Lagrange and $y_j$ the images by the polynomial
  function from $p_j = (x_j, y_j)$.
The $l_j$ are defined as follows : $\forall i \in \{0, ..., k-1\} | i \neq j, l_j(x) = \prod \frac{(x - x_i)}{(x_j - x_i)}$ ;
The constant term in the polynomial $f(x)$ is the private key $privateKey$.
```

3.6.2 Vote tally

The CAA decrypts the Y_j using Privatekey and extracts all the vote casts (id_{Office}, Bull, V_j, r_j). For each vote, using the voting code, it finds the corresponding ballot in the ballot database and verifies that the ballot actually contains V_j. It then retrieves the private key of the validation unit, decrypts all the entries of the ballot and from r_j extracts the different c_{oi}, checks that they are correct to ensure that the validation unit didn't fraud. It also verifies that the co_i code in V_j corresponds to that of the line considered in the ballot. In case of compliance, he adds this vote in the list of counts that are in the form of tuples (Id_{Office}, Y_j, Bull, c_{oi}). For each candidate c_i, the sum of the tuples containing c_{oi} gives the number of votes he has obtained. The CAA publishes the list of results on the voting board and vote counts for each candidate. The CAA sends the Y_j database to the headquarters of each political party.

3.6.3 Vote tally verification

Each party can also count independently and compare its results to those published by the CAA for verification and validation of results. Indeed each party has the following elements: the voting hash function f', the database of encrypted messages ($\sharp Bull$), the database of Y_j, the private key Privatekey, the private key of the validation unit and his own private key $Privatekey_k$.

To perform the count, it decrypts the Y_j with Privatekey, and extracts the tuples (id_{Office}, Bull, V_j, r_j). Then, using the private key of the validation unit and r_j, it decrypts the V_j, extract the c_{oi} and builds the database D1 = (id_{Office}, Bull, V_j, r_j, c_{oi}) which is then sorted according to the key (id_{Office}, Bull). Then, the encrypted messages $E_PublicKeyk$ (id_{Office}, Bull, v_j) are deciphered and extracted to form the database D2 = (id_{Office}, Bull, v_j). Similarly, D2 is sorted according to the key (id_{Office}, Bull). The join of D1 and D2 gives the database D whose each tuple is of the form (id_{Office}, Bull,v_j, V_j, r_j, c_{oi}). It calculates for each tuple, f'(id_{Office},v_j,V_j)

= Bull and compare the value obtained with that of Bull of the considered tuple. It verifies that the c_{oi} corresponding to this Bull published by the CAA is the same as that contained in D and that it is a correct c_{oi}.

At the end, each party has the count of votes it has made and it can compare to the published one. In case of inconsistency the party may appeal for electoral fraud and a recount is organized to detect it. Each voter can go to the seat of his party to verify their vote or directly on the bulletin board or through an independent organization that he trusts by means of his receipt Bull. The verification consists of ensuring that he finds a tuple containing his Bull in all the published tuples and that the associated c_{oi} really corresponds to the code of their candidate. At the same time, any independent organization may also check the published voting data and track count using copies of the receipts from the voters. Anyone else can check the count because the check is based solely on the data posted on the bulletin board.

IV EVALUATION

The published voting results are anonymous, encrypted and hashed and therefore do not reveal the voter's identity. In addition, the receipt provides to the voter is hashed with a one-way, collision-resistant hash function. It is very difficult to know for whom an voter voted. Only the voter has this information and shouldn't reveal it to a third person.

4.1 Properties

The proposed protocol guarantees a set of properties required for a verifiable voting protocol. In the following we will be presenting these different properties.

-Integrity: No vote can be added, deleted or modified without detection.

Let's assume that an election authority is malicious and wants to vote for voters who abstained or voters who are absent. It is blocked at a first level because the personal information of the voters stored in the machine are locked using the disposable masks since the enrollment. Only the voters hold this masks and the information is required for authentication. It is therefore impossible for an authority to know exactly all the voters who did not vote. To obtain them, authority must be in agreement with the voters. Suppose an authority has skipped this step and generates a random lock code v_j as a result of authentication and will vote. However, the authority has no assurance that this v_j has not been used yet. The vote it will make will be easily detected as fraud because the lock codes v_j are deposited by each voter in a sealed urn in the voting booth immediately after his vote. The auditory trace of the v_j will show that the v_j used by the authority is not correct in case we do not find this v_j because it cannot have identical v_j especially as the function f to obtain them is a collision-resistant hash function. In addition, the malicious authority must provide a r_j that is necessary for the creation of a ballot. Thus, the addition of a vote requires the disposable mask, the values r_j and v_j correct. These elements are difficult to obtain without the voter because they depend on the voter and his personal information. Since the votes are published on the voting bulletin board, the voter can easily check that his vote has not been deleted. Also, the paper trace audit can detect any deletion.

Moreover, the false ballots are detectable because the ballots are validated by the validation unit. Indeed, the validation unit upon receipt of the ballots created decrypts all the encrypted $E_v(c_{oi}, r_j)$ to ensure that the codes c_{oi} of candidates contained in these ciphered didn't modified when the ballot was created. Because the voters are called to make their choice from the encrypted in order to keep their vote secret. Because of this, any modification of the c_{oi} in the ciphered can cause a modification of the choice of the voter. The validation step of the ballot allows overcoming this type of fraud.

Suppose that the ballot validation unit is malicious and tries to rig the ballots with the false ciphered. For each vote, using the voting code, the CAA finds the corresponding ballot in the ballot database and verifies that the ballot actually contains V_j provided by the candidate. It then retrieves the private key of the validation unit, decrypts all the entries of the ballot and with r_j extracts the different c_{oi} and verifies that they are correct. It also verifies that the code c_{oi} in V_j corresponds to that of the line considered in the ballot. Thus, any modification made by the validation unit is detected by the CAA . In addition, the voting unit cannot modify a voter's vote because it is encrypted with the public key of the ballot validation unit. However, suppose that an external attacker intercepts the votes (Y_j) when sent to the CAA by the transfer unit. It is impossible for this attacker to know the contents of Y_j because he does not have the private key of the CAA . The functions of this key are held by the political parties and the key restored in the counting phase.

Suppose the CAA has changed, added or duplicated a vote. The Yj database of votes is held by both the central counting authority and the different political parties. An equivalence test ensures that everyone has the same database. Each political party constructs the database of the tuples results D1 = (id_{Office}, Bull, V_j, r_j, c_{oi}) and the database of messages D2 = (id_{Office}, Bull, v_j). The join of D1 and D2 makes it possible to obtain the database D whose each tuple is of the form (id_{Office}, Bull,v_j, V_j, r_j, c_{oi}). It calculates for each tuple, $f'(id_{Office}, v_j, V_j) = Bull$ and compares the value obtained with that of Bull of the tuple considered to ensure that the value of Bull has not been modified by the CAA. It also verifies that the c_{oi} corresponding to this Bull published by the CAA is the same as that contained in its database D and that it is a correct c_{oi}. Each party can verify the accuracy of the results published by the CAA. The CAA publishes the results on the voting bulletin board. Each voter can check with his Bull receipt that his vote has been taken into account. The verification consists in ensuring that he finds a tuple containing his Bull receipt in all the published tuples and that the associated c_{oi} actually corresponds to the code of the candidate of his choice. Any fraud of the CAA is detectable by the political parties. Similarly, any fraud by one of the political parties is detectable using the results of others or the CAA. One can easily check the concordance of the different results obtained at the end of the counts made by each of the parties.

Assume the voter is malicious and tries to lie about their vote during verification. The CAA and different political parties have the information to reconstruct the vote. Indeed, if the voter complains, their Bull receipt is recovered for verification and with this Bull we can prove that they are wrong. For each Bull, the corresponding v_j is known. v_j is the result of authentication which is a collision resistant hash function and therefore unique. The CAA has the Yjs which allow it to find the corresponding c_{oi} for each Bull. Each c_{oi} is associated with a precise candidate (application h) and is known. Thus, we know for which candidate Bull was issued. In addition, each party has the database D whose tuple is of the form (id_{Office}, Bull,v_j, V_j, r_j, c_{oi}). He can calculate for each tuple, $f'(id_{Office}, v_j, V_j) = Bull$ and compare the value obtained with that of Bull of the tuple considered. It also checks that the c_{oi} corresponding to this Bull published by the CAA is the same as that contained in its database D and that it is a correct c_{oi}. Also, each party finds for the Bull provided by the voter the candidate chooses.

Besides, it is difficult to add, delete, modify or duplicate an voter's vote in the proposed system. Indeed the voting process involves the different actors of an election and requires several key elements related to the voter. In addition, the protocol uses a public verification process.

- Democracy: Only authorized persons can vote once and only once.

We have the population ie the number of people (N_2) actually registered. Anyone wishing to vote must go through the registration phase and have at the end of which we have his disposable mask that is unique because there is a correspondence between his information and the mask provided.

- Confidentiality: Only the voter knows for which candidate he has voted.

The identifier and the personal information relating to a voter are locked using the disposable mask held solely by the voter. They cannot be known without the agreement of the voter. The ballot Bull, corresponding to the vote made by a voter and also used as a receipt for the voter, is the result of a collision-resistant and irreversible hash function. It is therefore impossible to make a correspondence between a voter and his vote. However, for each vote, the CAA knows the c_{oi} code of the chosen candidate and the random value r_j provided by the voter. The knowledge of these elements does not reveal to him the identity of the voter. As for the political parties, for each vote they are aware of the following elements: $v_j, V_j, r_j, c_{oi}, f'$. Each party has the elements that make it possible to reconstitute the vote (Bull) of a voter but in no case to reveal his identity. Although the lock code v_j is directly linked to the voter, it does not allow knowing his identity because v_j is obtained after authentication of the voter by an irreversible hash function of which the party is not aware. In addition, the voter cannot prove with his receipt to a third person for whom he has voted because the receipt is the result of a hash function. In sum, the use of the disposable mask and one-way hashing functions make it possible to ensure that it is difficult to match an voter and his vote and thereby guarantee confidentiality.

Verifiability is therefore ensured by the proposed protocol. Voters and political parties are involved in the voting process and are key elements of it, increasing confidence in the voting system. The system is based on simple processes that are understandable by all actors, transparent and public, which makes it reliable. In addition, the system provides voters and political parties with the ability to check votes while ensuring confidentiality. This property limits the purchase of votes and ensures the integrity of elections.

4.2 Complexity

The different phases of voting require a set of elements to complete. It is assumed that the SHA-256 hash and the RSA encryption are known and RSA is a elementary instruction. The complexity of SHA-256 hash is linear. The complexity calculation of the voting phases is summarized as follows:

-Voter enrollment phase

The voter information is obtained in one operation. The voter informations is encoded with the 4 masks by performing X-OR; this in 4 operations. The choice of the permutation number is done in one operation. We perform number (p) permutations. The hidden information of the voter and the different masks are encrypted with the public key of the authentication machine (CAL) in 5 operations. The complexity of this step is of the order of the number of permutations performed O (number (p)) and therefore linear.

-Key generation

The CAA public / private key generation and sharing algorithm is executed in O (n).

-Authentication phase

numero (p) is used to calculate the reverse permutation in order to find the permutation performed on the hidden information of the voter. The CAL decrypts the registration information in one operation. The construction of the authentication hash function f is done in one instruction. The calculation of the hash by the voting booth and its verification are done in constant time. The complexity of this phase is linear.

-Vote casting phase

. Creation of the ballot: The number of candidates concerned by the vote is n. To build the ballot, we encrypt the c_{oi} of each candidate concatenated to the random value provided by the voter. This construction is done in O (n).

. Validation of the ballot: The validation unit decrypts the bulletin in O (n) and enters the validation code in one operation.

. Voting stage: The choice of cipher V_j and the construction of Bull are each done in one operation. The construction of $\sharp Bull$ is done in O (n) and that of Y_j in one instruction. The complexity is in O (n).

- Vote counting phase and public verification

. Key reconstitution: The reconstitution of the key is done in O (n).

. Vote tally: The CAA decrypts the Y_j and verifies the V_j in one operation each. The decryption of the ballot and the verification of the codes is done in O (n). Then we add the ballot to the count. All of these operations are repeated for the N2 votes cast. The complexity is in O (n x N_2).

. Vote tally verification: The decryption of Y_js, V_js and messages using the political party's private key is done in O (N_2) each. The calculation of the Bulls by each political party is also done O (N_2).

V CONCLUSION AND PERSPECTIVES

The proposed system offers the possibility to check the vote. Voters, political parties have the opportunity to attest the published results as they are involved in the process. Voting procedures used are transparent; do not require heavy cryptographic skills on the voters' side. The system is understandable by the different users. However, the proposed system, like any other network protocol, can be exposed to attacks related to the network infrastructure. Although the system can not specify any security measures to counter this type of security attack, it has properties that can simplify the design of these security measures.

REFERENCES

Publications

[1] M. Ohkubo, F. Miura, M. Abe, A. Fujioka, and T. Okamoto. “An improvement on a practical secret voting scheme”. In: *International Workshop on Information Security*. Springer. 1999, pages 225–234.

[2] S. Ikonomopoulos, C. Lambrinoudakis, D. Gritzalis, S. Kokolakis, and K. Vassiliou. “Functional requirements for a secure electronic voting system”. In: *Security in the information society*. Springer, 2002, pages 507–519.

[3] R. Joaquim, A. Zúquete, and P. Ferreira. “REVS–a robust electronic voting system”. In: *IADIS International Journal of WWW/Internet* 1.2 (2003), pages 47–63.

[4] D. Chaum. “Secret-ballot receipts: True voter-verifiable elections”. In: *IEEE security & privacy* 2.1 (2004), pages 38–47.

[5] C. A. Neff. *Practical high certainty intent verification for encrypted votes*. 2004.

[6] D. Chaum, P. Y. Ryan, and S. Schneider. “A practical voter-verifiable election scheme”. In: *European Symposium on Research in Computer Security*. Springer. 2005, pages 118–139.

[7] K. Fisher, R. Carback, and A. T. Sherman. “Punchscan: Introduction and system definition of a high-integrity election system”. In: *Proceedings of Workshop on Trustworthy Elections*. 2006, pages 19–29.

[8] A. Kiayias, M. Korman, and D. Walluck. “An internet voting system supporting user privacy”. In: *2006 22nd Annual Computer Security Applications Conference (ACSAC’06)*. IEEE. 2006, pages 165–174.

[9] J. Benaloh. “Ballot Casting Assurance via Voter-Initiated Poll Station Auditing.” In: *EVT* 7 (2007), pages 14–14.

[10] B. Adida. “Helios: Web-based Open-Audit Voting.” In: *USENIX security symposium*. Volume 17. 2008, pages 335–348.

[11] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. “Scantegrity: End-to-end voter-verifiable optical-scan voting”. In: *IEEE Security & Privacy* 6.3 (2008), pages 40–46.

[12] A. Essex, J. Clark, and C. Adams. “Aperio: High integrity elections for developing countries”. In: *Towards Trustworthy Elections*. Springer, 2010, pages 388–401.

[13] A. Essex, J. Clark, U. Hengartner, and C. Adams. “Eperio: Mitigating Technical Complexity in Cryptographic Election Verification.” In: *EVT/WOTE* 13 (2010), pages 116–135.

[14] J. Furukawa, K. Mori, and K. Sako. “An implementation of a mix-net based network voting scheme and its use in a private organization”. In: *Towards trustworthy elections*. Springer, 2010, pages 141–154.

[15] A. Juels, D. Catalano, and M. Jakobsson. “Coercion-resistant electronic elections”. In: *Towards Trustworthy Elections*. Springer, 2010, pages 37–63.

[16] R. Joaquim and C. Ribeiro. “An efficient and highly sound voter verification technique and its implementation”. In: *International Conference on E-Voting and Identity*. Springer. 2011, pages 104–121.

[17] V. Cortier and B. Smyth. “Attacking and fixing Helios: An analysis of ballot secrecy”. In: *Journal of Computer Security* 21.1 (2013), pages 89–148.

[18] R. Joaquim, P. Ferreira, and C. Ribeiro. “EVIV: An end-to-end verifiable Internet voting system”. In: *Computers & Security* 32 (2013), pages 170–191.

[19] F. Zagórski, R. T. Carback, D. Chaum, J. Clark, A. Essex, and P. L. Vora. “Remotegrity: Design and use of an end-to-end verifiable remote voting system”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2013, pages 441–457.

[20] M. O. Rabin and R. L. Rivest. “Efficient end to end verifiable electronic voting employing split value representations”. In: *EVOTE 2014* (2014).

[21] A. Kiayias, T. Zacharias, and B. Zhang. “An efficient E2E verifiable e-voting system without setup assumptions”. In: *IEEE Security & Privacy* 15.3 (2017), pages 14–23.