International Journal of Computer Applications (0975 - 8887)
Volume 183 - No.44, December 2021

A Verifiable Electronic Voting System with Homomorphic Tallying using Elliptic-curve Cryptography

Charles F. de Barros
Department of Computer Science
Federal University of São João del Rei
Minas Gerais, Brazil

ABSTRACT

In this paper, the use of elliptic-curve cryptography (ECC) to build an end-to-end verifiable electronic voting system with homomorphic tallying is discussed. Verifiability is an important property of electronic voting systems, allowing any interested person to independently check that all votes were correctly recorded and counted. On the other hand, homomorphic tallying allows the votes to be counted without having to be individually decrypted, which reinforces ballot secrecy, another crucial security requirement for voting systems. After the encrypted votes are counted, only the final tally is decrypted in order to reveal the election result. Ballots are encrypted using elliptic-curve cryptography, which has been proven to offer high security levels, while keeping smaller key sizes, in comparison to other well known cryptographic primitives.

General Terms

Electronic voting, verifiability, elliptic-curve cryptography, homomorphic tallying

Keywords

Electronic voting, verifiability, elliptic-curve cryptography, homomorphic tallying

1. INTRODUCTION

Electronic voting systems have been in use around the world for more than two decades. These systems are typically based on the use of an electronic voting machine (EVM), which is basically a computer capable of storing the votes in digital format. An EVM offers an interface (a keyboard or a touchscreen) so that voters can choose their candidates and cast their votes.

In countries like Brazil and India, electronic voting machines are currently used nationwide. In the USA, they are used in a few States, together with other voting systems that comprise hand marked paper ballots and ballot marking devices (BMDs) [1].

Electronic voting machines, also known as Direct Recording Electronic (DRE) systems, store the votes in digital format, using some kind of computer memory. In Brazil, for example, the votes are stored in a removable result memory, which operates as a sort of flash drive, connected to the EVM by a USB port [2]. After the election is closed, the result memory is removed from the EVM and sent to the TRE (Regional Electoral Court) headquarters inside a sealed envelope.

DRE systems have been thoroughly studied, and its security vulnerabilities are well known and documented. In 2006, the California Secretary of State requested a top-to-bottom review of the California voting systems. At the request, the University of California, Berkeley, prepared a report presenting the findings of an analysis of the Diebold voting systems [3], which consisted of the AccuVote-TSX DRE, the AccuVote-OS optical scanner and the GEMS election management system. The report showed that the Diebold system did not provide sufficient security, due to its vulnerability to malicious software and susceptibility to viruses. The study also concluded that the AV-TSX system failed to protect ballot secrecy, because the votes were recorded in the order in which they were cast, and there was a record of the time each vote was cast. Finally, it was found that the GEMS central election management system could not prevent malicious insiders from tampering with ballot definitions and election results.

A security analysis of the Indian EVM also revealed major security flaws [4], that allowed a malicious insider to tamper with the system firmware even before it was built into the CPU, because there was no verification of the software integrity. The changes would be practically undetectable. The CPU could also be substituted by a look-alike CPU containing malicious software, since there was no cryptographic mechanism to identify the original hardware. Finally, additional hardware could be attached to the control unit's circuit board, allowing an adversary to directly read and write the EEPROM chips where the votes were recorded.

In face of the increasing number of vulnerabilities discovered in DRE-based systems, and in order to reinforce the auditability of the electoral process, some EVMs began to be equipped with VVPAT (Voter Verifiable Paper Audit Trail) printers, that allow the voters to confirm their selections on an independent paper record before casting their votes into digital memory. These paper records are preserved and can be used as a mechanism of post-election audit or recount. In India, for example, VVPATs began to be adopted in 2013 [5]. In the USA, EVMs equipped with VVPATs are used in some states. In Brazil, there was a recent proposal for a constitutional amendment in order to determine the use of VVPATs in the Brazilian EVMs, but the proposal was rejected, so

that Brazil remains as the only country in the world using DRE machines without VVPAT nationwide, even after the discovery of several vulnerabilities in the Brazilian EVM [6].
Although the use of VVPATs increase the auditability of voting systems, the voter is still required to trust the integrity of the paper records. End-to-end verifiable voting systems [7] were developed under the premise that voters should not be required to trust neither the election authority, nor the integrity of the voting software, hardware or even paper records. These systems allow every interested person to independently check that all votes were correctly recorded and counted, which represents the strongest notion of security for electronic voting systems.

1.1 Paper outline

This paper is organized as follows: section 2 presents a brief review on the security requirements of electronic voting systems, emphasizing verifiability and homomorphic tallying as a mechanism to reinforce ballot secrecy. In section 3, a theoretical review on elliptic-curve cryptography (ECC) is presented. Section 4 discusses the application of ECC to build a verifiable electronic voting system, with homomorphic tallying using a variant of the El Gamal encryption scheme, based on elliptic curves. Finally, section 5 presents the conclusions and final remarks.

2. SECURITY REQUIREMENTS OF ELECTRONIC VOTING SYSTEMS

This section presents a brief theoretical review on the security requirements of electronic voting systems, such as integrity, reliability, vote secrecy, authentication, among others. Special attention was given to the requirement of verifiability, which underlies end-to-end verifiable systems.

2.1 General security criteria

Any electronic voting system must meet certain criteria in order to be considered secure. Some of these criteria are mutually conflicting, which makes the design of secure electronic voting systems a special challenge. The main security requirements are the following:

(1) System integrity: the voting system, both hardware and software, must be tamper-proof. After the code is certified, no changes are allowed, and any attempt to tamper with the system must leave detectable evidence.
(2) Result integrity: the final tally should reflect voters intention. This means that all votes should be cast as intended, and counted as cast. No votes can be modified after they were cast. Any attempt of tampering with cast votes should be detected.
(3) Reliability: system bugs must be unlikely to occur.
(4) Availability: the voting system must be available during election time.
(5) Ballot secrecy: no voter should be able to reveal how they voted, even if they want to.
(6) Authentication: only authorized voters can cast their votes.
(7) Enfranchisement: all authorized voters must have the opportunity to vote.

Another important security property is **auditability**, according to which the voting system should be capable of generating election records, whose authenticity and integrity should be provable. The use of VVPATs is an attempt to reinforce this property.

As mentioned before, although the use of VVPATs increases auditability by offering the possibility of recounting the ballots, the voter is still required to trust the integrity of the paper records, which may be difficult to guarantee and verify. This is an intrinsic limitation of DRE-based systems, and the reason for the development of verifiable systems, which offer a very strong security property called end-to-end verifiability, discussed in the next section.

2.2 Verifiability as a security requirement

End-to-end verifiability (E2E) captures the notion that voters should not be required to trust election software, hardware or election officials in order to be convinced that the election outcome is correct. This notion goes hand in hand with the concept of software independence [8], which states that an undetected error in the voting system should not be able to cause an undetectable error in the election outcome.
Several E2E verifiable voting systems have been proposed in the literature, like Chaum's solution using visual cryptography [9], Prêt à Voter [10], which uses a randomized candidate list as the core idea, the ThreeBallot system proposed by Rivest [11], Scantegrity II [12], the Helios voting system [13], Belenios [14], among others.
E2E verifiable systems are based upon the idea that, instead of having to trust the integrity of election software, hardware or even paper records, the voter should be able to check the correctness of the election outcome electronically and independently. This can be achieved by producing a chain of evidence that can help check that all votes were:

(1) Cast as intended: this means that each voter is able to verify that their vote was correctly recorded;
(2) Counted as cast: this means that any interested person is able to verify that every recorded vote was included in the final tally.

This chain of evidence is typically published in some sort of public bulletin board by the election authority.

2.3 Encrypting the votes

In order to preserve ballot secrecy, all votes must be encrypted before being posted on the election bulletin board. However, voters must be convinced that the ciphertexts actually correspond to the encryption of their choices. In order to give the voter some guarantee that the system is behaving honestly during ballot encryption (by correctly encrypting the voter's choices), some verifiable systems employ a Benaloh challenge [15]:

(1) The voters are allowed to produce as many ballots as they wish, but only one of these ballots will be actually cast;
(2) The other encrypted ballots are marked as challenge ballots;
(3) The voter randomly selects the challenge ballots and asks the system to decrypt them, revealing all the information that was used to encrypt that vote;
(4) The voters can independently check that all the encrypted ballots were correctly generated.

Since the voting system does not know which ballots will be marked as challenges, and cannot even predict if the voter will choose to challenge or not, there is a high probability of detection of any dishonest behavior(for example, if the system presents the encryption of a choice that is different from the voter's choice).
To give a concrete example of this idea, consider the El Gamal encryption scheme [16], which is used by the Helios voting system. El Gamal is a public key encryption scheme, which uses a very

large prime number p and an integer g as public parameters. The secret key is a randomly selected integer x, while the public key is given by

$$y = g^x \pmod{p}. \tag{1}$$

A vote is encoded as an integer $v \in \{0, 1\}$, where the voter must choose 1 ir order to vote FOR a given candidate, and 0 to vote AGAINST a given candidate. The encryption of the vote, called a ciphertext, is given by the pair of integers (α, β), such that

$$\begin{array}{rcll} \alpha & = & g^r & \pmod{p} \\ \beta & = & g^v y^r & \pmod{p} \end{array} \tag{2}$$

where r is a randomly selected integer, called *ephemeral key*. We remark that this version is called exponential El Gamal, since in its original formulation, the value of β is given by vy^r instead of $g^v y^r$.
Suppose, for example, that the voter must choose between two candidates, say Alice and Bob. Assuming that the vote goes for candidate Alice, the unencrypted ballot looks like this:

Table 1. Unencrypted ballot.

Alice	1
Bob	0

On the other hand, the encryption of this ballot will consist of two El Gamal ciphertexts, one for each of the voter's choices (1 for Alice and 0 for Bob):

Table 2. Encrypted ballot.

Alice	(α_A, β_A)
Bob	(α_B, β_B)

where (α_A, β_A) is the ciphertext encoding the choice for Alice (the encryption of 1), and (α_B, β_B) is the encrypted choice for Bob (the encryption of 0). These ciphertexts were generated using ephemeral keys r_A and r_B, respectively. If the voter chooses to cast the ballot, the system throws away these ephemeral keys and records the ciphertexts. On the other hand, if the voter chooses to challenge the ballot, the system must decrypt the ciphertexts and reveal the ephemeral keys, so that the voter can check that each ciphertext in fact corresponds to the encryption of her choices.
When the voter chooses to cast the ballot, the system may generate some sort of tracking number, typically in the form of a cryptographic hash of the ballot, so that the voter can subsequently confirm that the encrypted ballot was correctly posted on the bulletin board. Spoiled ballots (those that were challenged by voters) can also be posted on the bulletin board for audit purposes. However, these ballots will not be included in the final tally.

2.4 Homomorphic tallying

After the list of encrypted votes is published on the bulletin board, the election authority proceeds to compute the final tally. The tallying process must provide evidence that all the encrypted votes were included in the tally, without revealing how any individual voted. This can be achieved by a technique called homomorphic tallying, which makes it possible to count the encrypted votes without having to decrypt them individually. After the encrypted votes are counted, the final tally is decrypted and the result is published, along with a mathematical proof that the tally matches the previously published encrypted votes. We shall discuss this mathematical proofs in the forthcoming sections.

Back to the El Gamal example, imagine that one must compute the total number of the votes for a given candidate, cast by n voters, in an election with m candidates. Assume that the i-th voter marks a choice for the j-th candidate, given by

$$v_{ij} \in \{0, 1\}. \tag{3}$$

It is clear that the total number of votes for that candidate is given by

$$t_j = \sum_{i=1}^{n} v_{ij}. \tag{4}$$

Since the votes were encrypted using the El Gamal encryption scheme, for each one of them an ephemeral key r_{ij} was used, so that each encrypted vote is given by

$$\begin{array}{rcll} \alpha_{ij} & = & g^{r_{ij}} & \pmod{p} \\ \beta_{ij} & = & g^{v_{ij}} y^{r_{ij}} & \pmod{p}. \end{array} \tag{5}$$

for $i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, m$. For each value of j, multiplying the ciphertexts of the votes cast by all voters yields

$$\begin{array}{rcll} A_j = \prod_{i=1}^{n} \alpha_{ij} & = & g^{r_{1j} + r_{2j} + \cdots + r_{nj}} & \pmod{p} \\ B_j = \prod_{i=1}^{n} \beta_{ij} & = & g^{v_{1j} + \cdots + v_{nj}} y^{r_{1j} + \cdots + r_{nj}} & \pmod{p}. \end{array} \tag{6}$$

Writing $r_j = r_{1j} + \cdots + r_{nj}$, it is clear that the pair (A_j, B_j) corresponds to the encryption of t_j using r_j as ephemeral key. The value of t_j can be decrypted by computing

$$D_j = B_j (A_j^x)^{-1} = g^{t_j} \pmod{p}. \tag{7}$$

In order to find the value of t_j, successive powers of g must be computed until the value of D_j is obtained as a result. Note that this is computationally viable, due to the relatively small values of t_j. The final election result then will be a tuple

$$T = (t_1, t_2, \cdots, t_m) \tag{8}$$

containing the total number of votes received by each candidate.

3. ELLIPTIC-CURVE CRYPTOGRAPHY

Elliptic curves are mathematical objects studied by algebraic geometry, although they have their origins in the field of analysis, namely in the theory of elliptic integrals and elliptic functions. The use of elliptic curves in cryptography was proposed independently by Koblitz [17] and Miller [18]. Currently, elliptic-curve cryptography, or simply ECC, represents one of the best options in terms of security and efficiency, due to its faster operations and smaller key sizes, when compared to other cryptographic primitives like RSA [19].
Mathematically speaking, an elliptic curve is the set of points that satisfy the equation

$$y^2 = x^3 + Ax + B \tag{9}$$

where A and B are constants over a certain field (the real numbers, for example). For cryptographic purposes, elliptic curves over the finite field $\mathbb{F}_p$ (the set of integers modulo p), denoted by $E(\mathbb{F}_p)$, where p is a large prime number, are commonly used. The restraint

$$4A^3 + 27B^2 \neq 0 \tag{10}$$

is imposed, so that the curve is considered a non singular curve. Mathematically speaking, the condition above imposes that the

tangent line is well defined in all points of the curve. A certain operation, a *sum of points*, can be defined over an elliptic curve so that it has a group structure. Recall that a group is an algebraic structure consisting of a set G and an operation $*$ with the following properties:

(1) Associativity: $a * (b * c) = (a * b) * c$ for all $a, b, c \in G$;
(2) Identity element: there is an element $e \in G$ such that $g * e = e * g = g$ for all $g \in G$;
(3) Inverse: every element $g \in G$ has an inverse $g^{-1} \in G$ such that $g * g^{-1} = g^{-1} * g = e$.

The group is said to be Abelian if the operation is commutative. In the case of elliptic curves, the operation is the sum of points denoted by $\oplus$, and the identity element is a special point, called *point at infinity* and represented by $\mathcal{O}$. Furthermore, every point $P = (x, y)$ of an elliptic curve has an inverse given by $P' = (x, -y)$, such that $P \oplus P' = P' \oplus P = \mathcal{O}$. The minus sign must be interpreted as modular subtraction in the case of elliptic curves over the finite field $\mathbb{F}_p$.
For all two points P and Q over an elliptic curve, their sum is given by

$$P \oplus Q = \begin{cases} P & \text{if } Q = \mathcal{O}; \\ Q & \text{if } P = \mathcal{O}; \\ \mathcal{O} & \text{if } P = Q' \\ R = (x_R, y_R) & \text{otherwise} \end{cases} \tag{11}$$

where the coordinates of the point R are given by

$$\begin{cases} x_R = \lambda^2 - x_P - x_Q \\ y_R = -y_P - \lambda(x_R - x_P) \end{cases} \tag{12}$$

The value of λ represents the slope of the straight line containing the points P and Q and is given by

$$\lambda = \begin{cases} \dfrac{y_P - y_Q}{x_P - x_Q} & \text{if } P \neq Q; \\ \dfrac{3x_P^2 + A}{2y_P} & \text{otherwise} \end{cases} \tag{13}$$

Considering elliptic curves over the field of integers modulo p, divisions must be interpreted as modular inversions, which can be efficiently computed by using the extended euclidean algorithm.
In order to compute multiples of a given point, given by

$$nP = \underbrace{P \oplus P \oplus \cdots \oplus P}_{n \text{ times}}, \tag{14}$$

one can employ an algorithm similar to the fast exponentiation method, as described in [20]. Define also $0P = \mathcal{O}$, and $nP = -nP'$ for all $n < 0$. The **order** of a point G is the smallest positive integer k such that

$$kG = \mathcal{O}. \tag{15}$$

For cryptographic purposes, points with very high order must be chosen, so that the underlying instance of the discrete logarithm problem (on whose hardness the security of these primitives depend) is hard to solve.

3.1 Elliptic-Curve El Gamal

In the elliptic-curve variant of the El Gamal cipher (ECEG) [21], a publicly known elliptic curve $E(\mathbb{F}_p)$ over the field $\mathbb{F}_p$, where p is a large prime also publicly known, is given as parameter, together with a point $G \in E(\mathbb{F}_p)$ with very high order. The secret key is a randomly selected integer x, and the public key is the curve point $P = xG$.
An integer v is encrypted as the pair of points over $E(\mathbb{F}_p)$ given by

$$\begin{aligned} R &= rG \\ S &= vG \oplus rP \end{aligned} \tag{16}$$

where r is a randomly chosen ephemeral key. The ciphertext is the pair (R, S). In order to decrypt it, one has to compute

$$(xR)' \oplus S = r(xG)' \oplus vG \oplus rP = vG \tag{17}$$

Assuming that the value of v is reasonably small, simply compute successive values of kG, for $k = 1, 2, 3, \cdots$, until the correct value of v is obtained.

4. A VERIFIABLE VOTING SYSTEM WITH ELLIPTIC-CURVE CRYPTOGRAPHY

In this paper, the use of elliptic-curve cryptography to build a verifiable voting system is proposed. The idea is that each vote is encrypted using the previously described elliptic-curve variant of El Gamal, and all votes are homomorphically counted, so that no individual ballot needs to be decrypted, reinforcing vote secrecy. The proposal is based on the Helios voting system, with some adaptations on the proofs of decryption correctness and ballot integrity.
Prior to the election, candidates and voters are registered. The list of candidates and authorized voters is published in the election bulletin board, hosted at the election official website. Assume that the number of candidates is m, while the number of voters is n. To each candidate and voter is assigned a unique identifier, which for the sake of simplicity is considered simply as an integer $i = 1, 2, \cdots, n$ for the voters, and $j = 1, 2, \cdots, m$ for the candidates.
The election authority generates a random secret key x and the public key, consisting of an elliptic curve $E(\mathbb{F}_p)$ and two points G, P, where $P = xG$. The public key is made available on the bulletin board, while the secret key is securely stored.
On election day, authorized voters are given access to the voting system. After marking their choices (assigning the value 1 for the chosen candidate and 0 to all others), the system encrypts the votes. The choice marked by the i-th voter for the j-th candidate is given by

$$v_{ij} \in \{0, 1\} \tag{18}$$

and the encryption of this choice is an elliptic-curve El Gamal ciphertext given by

$$\begin{aligned} R_{ij} &= r_{ij}G \\ S_{ij} &= v_{ij}G \oplus r_{ij}P \end{aligned} \tag{19}$$

As previously described, a Benaloh challenge may be used to verify that the voting system is behaving honestly during the encryption of the votes (by correctly encrypting the actual choices made by voters). If the voter chooses to challenge the ballot, the ciphertext is decrypted and the ephemeral key is disclosed. Otherwise, the ephemeral key is discarded and the ballot is cast.

4.1 Proving the correctness of each ballot

Although the Benaloh challenge decreases the probability of the system behaving dishonestly without being caught, the ciphertexts contained in the ballot cast by the voter are not disclosed and could correspond, with non-negligible probability, to encryptions of arbitrary values, which could be large numbers to increase the

total votes for a given candidate, or even negative numbers in order to steal votes from a given candidate.

In order to avoid such dishonest behavior, every interested person must be given the opportunity to audit the ciphertexts that encrypt the choices made by the voters, in order to be convinced that they correspond to encryptions of valid votes, which means a value that is either 0 or 1. However, the proof must be given without revealing the plaintext, in order to preserve ballot secrecy.

Systems like Helios and Belenios employ zero-knowledge proofs [22] as a means of proving the correctness of each cast ballot. A zero-knowledge proof is a method by which a prover can mathematically prove the validity of a statement to a verifier, without conveying any information to the verifier, apart from the fact that the statement is, in fact, valid. In this case, the prover is the election authority, and the verifier is any person interested in checking the integrity of the election.

Zero-knowledge proofs can be divided into interactive and non-interactive. An interactive proof consists, as the name itself suggests, of some sort of interaction between the prover and the verifier, typically involving the following steps:

(1) The prover sends a set of *commitments* to the verifier;
(2) The verifier sends a *challenge* to the prover;
(3) The prover sends back a *response*
(4) After some sort of verification, the verifier may accept or reject the proof.

In a voting scheme scenario, interactive proofs are not practical due to the potentially huge number of verifiers. In order to circumvent this problem, one can employ the Fiat-Shamir heuristic [23], which transforms an interactive proof into a non-interactive one. Instead of receiving the challenge from the verifier, the prover computes the challenge as the hash of the commitments, and makes all these pieces of information (commitments, challenge and response) available to the verifier, without the need for any interaction.

The proof adopted by Helios is known as Chaum-Pedersen protocol [24], transformed into a non-interactive proof by using the Fiat-Shamir transform. It is used to convince the verifier that each ciphertext actually corresponds to the encryption of either 0 or 1. However, this kind of proof, also known as *set membership proof*, has some well known pitfalls [25], so that it is possible to encrypt an arbitrary value V and still produce a convincing "proof" that the ciphertext is the encryption of either 0 or 1. In order to avoid these pitfalls, the proposed voting system in this paper makes use of an adaptation of a strong version of the Fiat-Shamir transform, as described in [26].

Given a ciphertext (R_{ij}, S_{ij}), a set membership proof is a zero-knowledge non-interactive proof that it corresponds to an encryption of either 0 or 1. It is given by the set

$$\Pi_{ij} = \left\{(X_{ij}^{(0)}, Y_{ij}^{(0)}, c_{ij}^{(0)}, d_{ij}^{(0)}), (X_{ij}^{(1)}, Y_{ij}^{(1)}, c_{ij}^{(1)}, d_{ij}^{(1)})\right\} \quad (20)$$

where, for $k = 0, 1$:

(1) The commitment $(X_{ij}^{(k)}, Y_{ij}^{(k)})$ is a pair of elliptic curve points;
(2) The challenge $c_{ij}^{(k)}$ and the response $d_{ij}^{(k)}$ are integer numbers;
(3) If $k \neq v_{ij}$, the values of $c_{ij}^{(k)}$ and $d_{ij}^{(k)}$ are chosen at random, and the commitments are given by

$$\begin{aligned} X_{ij}^{(k)} &= d_{ij}^{(k)} G \oplus (-c_{ij}^{(k)}) R_{ij} \\ Y_{ij}^{(k)} &= d_{ij}^{(k)} P \oplus (-c_{ij}^{(k)})(S_{ij} \oplus kG') \end{aligned} \quad (21)$$

(4) If $k = v_{ij}$, an integer w_{ij} is chosen at random, and the commitments are given by

$$\begin{aligned} X_{ij}^{(k)} &= w_{ij} G \\ Y_{ij}^{(k)} &= w_{ij} P \end{aligned} \quad (22)$$

(5) The expected overall challenge is given by

$$c_{ij} = H(G, P, R_{ij}, S_{ij}, X_{ij}^{(0)}, Y_{ij}^{(0)}, X_{ij}^{(1)}, Y_{ij}^{(1)}) \quad (23)$$

for a given cryptographically secure hash function H.

(6) For $k = v_{ij}$, the challenge and the response are respectively given by

$$c_{ij}^{(k)} = c_{ij} - c_{ij}^{(1-k)} \quad (24)$$

and

$$d_{ij}^{(k)} = w_{ij} + r_{ij} c_{ij}^{(k)} \quad (25)$$

For each ciphertext, the verification procedure consists of the following steps:

(1) For $k = 0, 1$, check that

$$\begin{aligned} d_{ij}^{(k)} G &= X_{ij}^{(k)} \oplus c_{ij}^{(k)} R_{ij} \\ d_{ij}^{(k)} P &= Y_{ij}^{(k)} \oplus c_{ij}^{(k)} (S_{ij} \oplus kG') \end{aligned} \quad (26)$$

(2) Check that

$$c_{ij}^{(0)} + c_{ij}^{(1)} = H(G, P, R_{ij}, S_{ij}, X_{ij}^{(0)}, Y_{ij}^{(0)}, X_{ij}^{(1)}, Y_{ij}^{(1)}) \quad (27)$$

It is clear that, if the system behaves honestly, then every ballot will pass verification. In fact, let (R_{ij}, S_{ij}) be a ciphertext encrypting the voter's choice v_{ij}, and $k \in \{0, 1\}$.

(1) If $k = v_{ij}$, then $X_{ij}^{(k)} = w_{ij} G$, $Y_{ij}^{(k)} = w_{ij} P$ and $d_{ij}^{(k)} = w_{ij} + r_{ij} c_{ij}^{(k)}$. Therefore,

$$d_{ij}^{(k)} G = (w_{ij} + r_{ij} c_{ij}^{(k)}) G = X_{ij}^{(k)} \oplus c_{ij}^{(k)} R_{ij} \quad (28)$$

(2) On the other hand, if $k \neq v_{ij}$, then (21) yields

$$d_{ij}^{(k)} G = X_{ij}^{(k)} \oplus c_{ij}^{(k)} R_{ij} \quad (29)$$

and (22) yields

$$d_{ij}^{(k)} P = Y_{ij}^{(k)} \oplus c_{ij}^{(k)} (S_{ij} \oplus kG'). \quad (30)$$

(3) Identity (27) is clearly satisfied because of (23) and (24).

Hence, each ballot consists of m ciphertexts, given by pairs of points (R_{ij}, S_{ij}), and each pair is accompanied by a zero-knowledge proof Π_{ij} that it corresponds to the encryption of either 0 or 1. Additionally, a cryptographic hash of the ballot is presented to the voter, in order to ensure that the ballot will not be modified. The hash can be computed over the concatenation of all encrypted choices, so that

$$\text{Ballot hash } = H((R_{i1}, S_{i1}) || \cdots || (R_{im}, S_{im})) \quad (31)$$

where H is a cryptographically secure hash function. After the election is closed, a list of all encrypted ballots is published on the election bulletin board. Each entry of the bulletin board has the structure shown in Table 3.

Any interested party may independently check the correctness of each ballot by checking the zero-knowledge proofs. Alternatively, each ballot may include the spoiled ciphertexts disclosed to the voters during ballot preparation.

Table 3. Encrypted ballot for the i-th voter, as published in the bulletin board. Each ciphertext is accompanied by a zero-knowledge proof that it encrypts either 0 or 1.

Voter ID	i
Encrypted choice cand. 1	$(R_{i1}, S_{i1}, \Pi_{i1})$
Encrypted choice cand. 2	$(R_{i2}, S_{i2}, \Pi_{i2})$
...	...
Encrypted choice cand. m	$(R_{im}, S_{im}, \Pi_{im})$
Ballot hash	$H((R_{i1}, S_{i1}) \| \cdots \| (R_{im}, S_{im}))$

4.2 Homomorphic tally

Once the election is closed, the election authority proceeds to compute the final tally. Once again, the total number of votes for the j-th candidate is given by

$$t_j = \sum_{i=1}^{n} v_{ij}. \tag{32}$$

Hence, by summing up the ciphertexts (using the previously defined sum of points) that encode the votes cast by all voters, we obtain, for each candidate,

$$\begin{array}{c} R_j = R_{1j} \oplus \cdots \oplus R_{nj} \\ S_j = S_{1j} \oplus \cdots \oplus S_{nj} \end{array} \tag{33}$$

which yields

$$\begin{array}{rcl} R_j & = & (r_{1j} + r_{2j} + \cdots + r_{nj})G \\ S_j & = & (v_{1j} + \cdots + v_{nj})G \oplus (r_{1j} + \cdots + r_{nj})P \end{array} \tag{34}$$

Putting $r_j = r_{1j} + \cdots + r_{nj}$, it is clear that the pair (R_j, S_j) corresponds to the encryption of t_j using r_j as ephemeral key. Hence, the value of t_j can be decrypted by computing

$$Q_j = S_j \oplus (xR_j)' = t_j G. \tag{35}$$

To find the value of t_j, compute successive multiples of G, until the point Q_j is found as a result. The final election result will be given by the tuple

$$T = (t_1, \cdots, t_m). \tag{36}$$

4.3 Proving the correctness of decryption

The election authority must also provide a convincing proof that the decrypted result is correct. Once again, a zero-knowledge Chaum-Pedersen proof must be provided, by means of which the election authority can prove knowledge of the secret decryption key x (without revealing it!).
For each value of $j = 1, 2, \cdots, m$, the election authority computes a *decryption factor* given by

$$F_j = xR_j, \tag{37}$$

where R_j is given by (33), and commits to the points

$$\begin{array}{c} X_j = w_j G \\ Y_j = w_j R_j \end{array} \tag{38}$$

for a randomly chosen w_j. The challenge is the cryptographic hash of the pair (X_j, Y_j), concatenated with the public key and the homomorphic sum of the ciphertexts, given by

$$c_j = \text{hash}(G, P, R_j, S_j, X_j, Y_j) \tag{39}$$

and the response is the integer

$$d_j = w_j + c_j x. \tag{40}$$

The proof of correct decryption is given by the set

$$\Pi_j = \{F_j, X_j, Y_j, c_j, d_j\}. \tag{41}$$

In order to verify that the election authority used the correct secret key to decrypt the election tally, a verifier (any person interested in verifying the election outcome) must check that:

$$d_j G = X_j \oplus c_j P \tag{42}$$

and

$$d_j R_j = Y_j \oplus c_j F_j. \tag{43}$$

It is easy to see that, if the prover (the election authority) is honest, it will pass the verification procedure, because

$$d_j G = w_j G \oplus c_j x G = X_j \oplus c_j P \tag{44}$$

and

$$d_j R_j = w_j R_j \oplus c_j x R_j = Y_j \oplus c_j F_j. \tag{45}$$

A malicious prover, who does not know the secret key x, will be caught with overwhelming probability, because even if a fake decryption factor is presented, verification (42) would fail because $x'G \neq P$ for a value $x' \neq x$.
After the total number of votes t_j for each candidate is published, any verifier can confirm the correctness of the election outcome by verifying identities (42) and (43) and by checking that

$$t_j G = S_j \oplus (F_j)'. \tag{46}$$

4.4 Full election verification procedure

Now, the verification algorithm for the entire election is formally described. The first verification step consists of checking the integrity of each encrypted ballot. The procedure takes as input the voter identifier i and it works as follows:

```
Procedure: ballot_Verify(i)

For j = 1,2,...,m:
    c_ij = 0
    for k = 0,1:
        if d_ij^(k) G ≠ X_ij^(k) ⊕ c_ij^(k) R_ij
            return False
        if d_ij^(k) P ≠ Y_ij^(k) ⊕ c_ij^(k) (S_ij ⊕ kG')
            return False
        c_ij = c_ij + c_ij^(k)
    if c_ij ≠ H(G, P, R_ij, S_ij, X_ij^(0), Y_ij^(0), X_ij^(1), Y_ij^(1))
        return False
return True
```

The second step of verification consists of checking the integrity of the election tally. For each candidate, we must compute the homomorphic sum of the ciphertexts, cast by all voters, and then verify whether identities (42), (43) and (46) hold. The procedure takes as input the candidate identifier j and it works as follows:

```
Procedure: candidate_tally_Verify(j)

R_j = O
S_j = O
For i = 1,2,...,n:
    if not ballot_verify(i)
        return False
```

```
    R_j = R_j ⊕ R_ij
    S_j = S_j ⊕ S_ij
if t_j G ⊕ F_j ≠ S_j :
    return False
if d_j G ≠ X_j ⊕ c_j P
    return False
if d_j R_j ≠ Y_j ⊕ c_j F_j
    return False
return True
```

Hence, the entire election verification procedure, taking as input the number of candidates m and the number of voters n, can be done as follows:

```
Procedure: full_election-verify(m,n)

for j = 1,··· ,m
    if not candidate_tally_verify(j)
        return False
return True
```

5. PRELIMINARY EXPERIMENTAL RESULTS

Although this paper does not address practical implementation aspects of the proposed voting system, a prototype of an implementation in Python language is currently undergoing a series of preliminary tests. The main goal of these tests is to assess the efficiency of elliptic-curve cryptography during the following operations:

(1) Computation of the homomorphic tally;
(2) Decryption of the homomorphic tally.

The prototype was implemented in Python 3.6.9, running over Linux Mint 19.3 on an Intel Core i3-7020U CPU with 2.30GHz and 4 GiB of RAM. In each test, a list of random votes (integers that were either 0 or 1) was generated, then each vote was encrypted with an elliptic-curve El Gamal public key, using the NIST p-256 curve. The results are summarized in Table 4.

Table 4. Efficiency assessment of elliptic-curve cryptography.

Number of random votes	Tally	Hom. tally comp.	Decryption
32	12	0.0044s	0.0281s
100	48	0.0143s	0.0298s
200	109	0.0278s	0.0340s
500	265	0.0669s	0.0439s
1000	490	0.1406s	0.0589s

6. FINAL REMARKS AND FUTURE WORK

This paper presented a proposal for the use of elliptic-curve cryptography to build a verifiable voting system, taking advantage of the fast operations and compact keys offered by elliptic curves. The proposed system allows every interested person to verify the correctness of the election outcome by means of a zero-knowledge proof of the decryption, together with homomorphic tallying, which allows the votes to be counted without having to be decrypted, reinforcing ballot secrecy.
Preliminary tests were performed in order to evaluate the efficiency of elliptic-curve cryptography during the operations of homomorphic tally computation and decryption. Further research on other security aspects of the proposed system, as well as more practical issues related to its implementation, is highly encouraged. Future work must also include the zero-knowledge proofs implementation and verification tests.

7. ACKNOWLEDGMENTS

Our thanks to the referees for their useful comments and suggestions that contributed to the improvement of this paper.

8. REFERENCES

[1] Verifier. The Verifier - Polling Place Equipment. https://verifiedvoting.org/verifier/#mode/navigate/map/ppEquip/mapType/normal/year/2022.

[2] Brazilian Superior Electoral Court (TSE). Brazilian electronic voting machine: 20 years in favor of democracy. http://bibliotecadigital.tse.jus.br/xmlui/handle/bdtse/6137?locale-attribute=en, 2016.

[3] Joseph A. Calandrino, Ariel J. Feldman, J. Alex Halderman, David Wagner, Harnal Yu, and William P. Zeller. Source code review of the Diebold voting system, 07 2007.

[4] Scott Wolchok, Eric Wustrow, J. Alex Halderman, Hari K. Prasad, Arun Kankipati, Sai Krishna Sakhamuri, Vasavya Yagati, and Rop Gonggrijp. Security Analysis of India's Electronic Voting Machines. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 1 – 14, New York, NY, USA, 2010. Association for Computing Machinery.

[5] Election Comission of India. Status paper on Electronic Voting Machine. https://eci.gov.in/files/file/8756-status-paper-on-evm-edition-3/, 2013.

[6] Diego Aranha, Marcelo Karam, André Miranda, and Felipe Scarel. Software vulnerabilities in the brazilian voting machine, 01 2014.

[7] Syed Taha Ali and Judy Murray. An overview of end-to-end verifiable voting systems, 2016.

[8] Ronald L. Rivest and John P. Wack. On the notion of software independence in voting systems. *PHIL. TRANS. R. SOC. A*, pages 3759–3767, 2008.

[9] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38 – 47, January 2004.

[10] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: A voter-verifiable voting system. *Trans. Info. For. Sec.*, 4(4):662 – 673, December 2009.

[11] Ronald Rivest. The threeballot voting system, 11 2006.

[12] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *Proceedings of the Conference on Electronic Voting Technology*, EVT'08, USA, 2008. USENIX Association.

[13] Ben Adida. Helios: Web-based open-audit voting., 01 2008.

[14] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. *Belenios: A Simple Private and Verifiable Electronic Voting System*, pages 214–238. Springer International Publishing, 2019.

[15] Josh Benaloh, Ronald Rivest, Peter Y. A. Ryan, Philip Stark, Vanessa Teague, and Poorvi Vora. End-to-end verifiability, 2015.

[16] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10 – 18, Berlin, Heidelberg, 1985. Springer-Verlag.

[17] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.

[18] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[19] Nelson Josias Gbètoho SAHO and Eugène C. Ezin. Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptography through RSA Algorithm. In *CARI 2020 - Colloque Africain sur la Recherche en Informatique et en Mathématiques Apliquées*, Thiès, Senegal, October 2020.

[20] J. Hoffstein, J. Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography*. Undergraduate Texts in Mathematics. Springer, New York, NJ, USA, 2008.

[21] Kefa Rabah. Elliptic Curve El Gamal Encryption and Signature Schemes. *Information Technology Journal*, 03 2005.

[22] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. Association for Computing Machinery.

[23] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[24] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 89–105, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[25] Véronique Cortier, Pierrick Gaudry, and Quentin Yang. How to fake zero-knowledge proofs, again. In *E-Vote-Id 2020 - The International Conference for Electronic Voting*, Bregenz / virtual, Austria, 2020.

[26] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 626–643, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.