



Article

Distributed Remote E-Voting System Based on Shamir's Secret Sharing Scheme

Marino Tejedor-Romero ^{1,*}, David Orden ^{1,†}, Ivan Marsa-Maestre ^{2,†}, Javier Junquera-Sanchez ^{3,†}
and Jose Manuel Gimenez-Guzman ^{4,†}

¹ Departamento de Física y Matemáticas, Universidad de Alcalá, 28805 Alcalá de Henares, Spain; david.orden@uah.es

² Departamento de Automática, Universidad de Alcalá, 28805 Alcalá de Henares, Spain; ivan.marsa@uah.es

³ Departamento de Ciencias de la Computación, Universidad de Alcalá, 28805 Alcalá de Henares, Spain; javier.junquera@uah.es

⁴ Departamento de Comunicaciones, Universitat Politècnica de València, 46022 Valencia, Spain; jmgimenez@upv.es

* Correspondence: marino.tejedor@uah.es

† These authors contributed equally to this work.

Abstract: A number of e-voting systems have been proposed in the last decades, attracting the interest of the research community. The challenge is far from being fully addressed, especially for remote systems. In this work, we propose DiverSEC, a distributed, remote e-voting system based on Shamir secret sharing, operations in Galois field and *mixnets*, which enables end-to-end vote verification. Parties participate as nodes in the network, protecting their interests and ensuring process integrity due to the conflicting interests. The threat model is very conservative, not letting even the most privileged actors to compromise votes privacy or integrity. Security in depth is implemented, overlapping different mechanisms to offer guarantees even in the most adverse operating conditions. The main contributions of the resulting system are our proposal for secret-sharing among the political parties, which guarantees that no party can compromise the integrity of the ballot without being detected and identified in real time, and the computational and architectural scalability of the proposal, which make it easy to implement.

Keywords: e-voting; remote voting; verifiable voting; secret sharing



check for updates

Citation: Tejedor-Romero, M.; Orden, D.; Marsa-Maestre, I.; Junquera-Sanchez, J.; Gimenez-Guzman, J.M. Distributed Remote E-Voting System Based on Shamir's Secret Sharing Scheme. *Electronics* **2021**, *10*, 3075. <https://doi.org/10.3390/electronics10243075>

Received: 25 October 2021

Accepted: 7 December 2021

Published: 9 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

During the past decades, the e-voting research field, essential for the development of a bigger concept called e-democracy, has been attracting growing interest. First, because elections are a critical requirement for the proper operation of the current representative-democratic government systems. Second, because we have experienced an explosive digital development that enables alternative voting methods. In this moment, there is powerful and efficient hardware with an acceptable cost. At the same time, there are lots of technology-related and highly-capable professionals. Both circumstances enable combining elections and digital automation.

E-Voting systems present immediate advantages over the paper ballot system. They are more inexpensive and time-efficient, because traditional elections require the orchestration of many people and many resources in order to offer security guarantees. Electronic elections can be not only simpler and cheaper, but also more user-friendly.

This idea is not novel. There are many publications about e-voting, and real-world cases of elections powered using digital resources [1,2]. However, this fact does not imply that e-voting has been properly accepted. These cases have been mostly affected by controversy. In fact, several subsequent analysis over these real-world cases have revealed that the protocols and systems chosen by the officers were not designed from a secure

perspective and had serious vulnerabilities [3,4]. For this reason, we need cryptographic techniques and a secure design. Powerful hardware and a developed IT industry are not sufficient by themselves. A correct design is not only meant to guarantee the integrity and privacy of the vote; it is also important in order to gain the trust of users. Specially nowadays, when the popular opinion of e-voting is not completely favorable.

There are already several e-voting proposals, which use cryptography to solve these requirements, but we can still find weaknesses that must be solved before we take another step forward. This paper contributes to bridge this gap by proposing DiverSEC: a distributed, remote e-voting system (REV) which satisfies desirable properties like ballot secrecy, integrity and verifiability, incorporating elements such as Shamir secret sharing and polynomial rings, and a novel architecture based on a *mixnet*, which distributes responsibility of the process among the different political parties. As a result, none of them can hinder the integrity of the ballot without being detected and identified by the others in real time. The proposal is also flexible and easy to implement, with a computational complexity that is linear to the number of voters, which can be easily divided in electoral districts if needed.

The rest of this paper is organized as follows. Section 2 reviews the desirable properties for an e-voting system, and briefly discusses the most relevant proposals to date. Then Section 3 presents the threat model where our proposal operates, while Section 4 describes the actual proposal: the DiverSEC e-voting system. Section 5 discusses the main properties, strengths and limitations of our approach. Finally, Section 6 summarizes our conclusions and sheds light on some future research directions.

2. State-of-the-Art

2.1. Desirable Properties for E-Voting Systems

There are several properties that are required of e-voting systems, from the basic properties extracted from traditional paper ballot elections (e.g., ballot secrecy or legitimacy), to other requirements, inherent to electronic alternatives. These have been already listed and explained by security researchers [5]. In the following, we briefly discuss the ones more relevant to our work:

- **Ballot secrecy** : Only the voter can know the actual vote. In an e-voting system, this means ensuring vote confidentiality against other voters, against attackers, and even against the very entity performing the election.
- **Legitimacy**: Only previously registered participants can vote. That is, in the final ballot there must not be any vote from an unauthorized participant. In electronic systems, this is guaranteed via authentication.
- **Eligibility**: Just as illegitimate voting is not allowed, legitimate voters can emit only one vote.
- **Accuracy**: The final result accounts for all legitimate votes, without any alteration between emission and tally. This is related to vote integrity.
- **Individual verifiability**: The voter can verify the correctness of the election process she has participated in.
- **Coercion resistance**: A voter can freely emit a vote even under coercion from a third party. The usual way to achieve this property is by means of the impossibility for the voter to prove her own vote. In this way, a third party cannot be sure of the final decision of a voter, which indirectly de-incentivize coercion.
- **Robustness**: The system remains dependable even when some actors actively attempt to corrupt the process. The types of corruption attempts are pre-defined. This is an important requirement for an e-voting system, due to the number of actors involved. The system must be designed taking into account the possibility of corrupted input.

These requirements are not completely independent. Some are connected between them (e.g., accuracy and verifiability). Others have a certain degree of incompatibility, like ballot secrecy against verifiability, and individual end-to-end verifiability against coercion resistance. This is because the ability to track your own vote across the system implies

having exclusive information, and voters could offer their private verification information to an attacker. Therefore, this balance is typically the most difficult to satisfy.

Taking this into account, the goal is to achieve a reasonable compromise, adapted to the current needs of the situation and the voters, and to the advantages that we deem most valuable of the e-voting protocols over traditional ballots.

Besides the basic properties, there are other kinds of concerns about e-voting systems we should solve before deploying them, regarding, for instance, electronic system guarantees. Given any e-voting system, the voters have little knowledge of what's happening under-the-hood, beyond their interaction with the system. This is why an open-source public formalization is deeply encouraged. However, an open-source software implementation of the e-voting system is not enough to solve this distrust, because it does not necessarily imply that everyone in the system is running the open-source implementation [6]. Further on, we will describe the threat and adversarial model we have considered for our proposal but, for the moment, the most cautious choice is to be wary of every actor in the system.

Under this assumption, the only way to guarantee the voter's security and trust, along with the tally's integrity is a secure design. An e-voting system is resistant even to corruption coming from inside the system's most privileged actors only if:

- The voter sends only the minimum information to the system for the entire process to be completed correctly. The most critical point here, is that, with that minimum amount of information, a malicious privileged actor should not be capable of compromising the security of the vote.
- A malicious administration is not able to convincingly fake the tally. A usual way to achieve this are distributed tally systems, in which there are no centralized tally authorities. The degree of distribution may vary, ranging from a few nodes, as is our case, to self-tallying systems where every user may perform the tally independently [7].

These two conditions allow the voter to participate securely in the election without checking the software of the rest of the nodes. [5] regards this property as software independence, although the author also refers to unintentional errors, not only intentional corruption.

2.2. Protocols and E-Voting Systems to Date

We can find a wide variety of e-voting protocols in the literature, for a wide diversity of scenarios, ranging from direct-recording electronic voting machines (DRE) to remote systems, varying the use they make of cryptography, and so forth. The first relevant proposals, driven by renowned cybersecurity researchers are ThreeBallot [8], PunchCard [9], Scantegrity [10]. These three proposals use paper ballots as physical backup and optical sensors to guarantee the integrity and privacy of each user's vote. The proposals are original and creative, and are the context of our state-of-the-art. However, we try to advance through our new proposal, towards a completely virtual, remote and verifiable approach.

Regarding the use of cryptography, we can find several categories of protocols, depending on the mechanisms used [5]:

- **Mixnet** [11]: They use a layered architecture where a chain of proxy servers is created. These servers receive an input with data from different sources, and they pass the data to the next proxy after a random permutation. This makes impossible to trace the source of an input to the system (unless all nodes are corrupt). This is the underlying principle of networks like Tor [12]. Some protocols that use *mixnets* are [13,14].
- **Blind signature** [15]: With a blind signature a principal obtains a cryptographic validation for a message from another principal, without disclosing the message to the latter. Typically, a voter gets the blind signature for her vote from the administration, without the administration knowing the content of her vote. Some protocols that use blind signatures are [16,17].
- **Homomorphic encryption** [18]: They use cryptographic techniques enabling to process the information without decrypting it, therefore handling votes in an aggregate manner without revealing individual votes [7,19,20].

- **Blockchain [21]:** Blockchain is an emerging technology that jumped to the popular culture during the last years, mainly because it is the base of cryptocurrencies. The principle of blockchain is a data structure that organizes items of information in blocks, which are connected through hashes. The blockchain data is distributed, stored and validated by the nodes. This is just a very small summary of this technique, since it is the most complex point in this list, with lots of variations, different consensus algorithms, and further analysis.

We have seen an extraordinary number of e-voting proposals using Blockchain [22]. However, we consider that this technology is not suitable for a really secure e-voting system under our current threat model (at least in its current state). The main drawbacks of the blockchain technology are its poor scalability, and the needless computational and energetic waste [23]. Recently, the alternative 'Proof of Stake' consensus algorithm has been proposed to replace 'Proof of Work', as a means to solve the computational problem, however, it still presents serious security drawbacks [24]. Although we have evaluated that, in our threat model, its advantages do not overweight its problems, there are some key design decisions we would like to extract and apply to our proposal: a distributed network, and global verifiability of the messages sent through hashes.

- **Shamir's secret sharing [25]:** It is a well-known cryptographic technique, although it is used significantly less often in e-voting [26]. It has been recently used in very diverse application areas, such as randomness [27] and authentication [28]. The naive version of Shamir's secret sharing presents some known flaws, but, at the same time, there are known solutions to these problems. For example:
 - If this scheme is deployed using standard integer arithmetic, it does not fulfill all the basic properties of the secret sharing concept. This is because shares leak data about the secret. This is solved easily in our proposal using a finite field.
 - This scheme is not verifiable. While there are schemes that aim to solve this issue, like the ones based on PVSS [27], in our proposal we solve it using our own integrated mechanism, as we will see below.
 - If the shares are revealed sequentially, the last shareholder can alter the last share in order to manipulate the output of the interpolation process. The easiest way to overcome this issue is to force the shareholders to publish a hash of their shares first, so they cannot change their share. This is typically called 'commitment'.

These groups are not exclusionary. In some cases, multiple techniques are overlapping in the same e-voting system, and approaches are classified according to the main principle of the schema.

In this moment, the most accepted and relevant remote and verifiable proposals are Helios and Civitas/JCJ:

- **Helios [6]:** Helios [6] is an e-voting system designed to be used through a web browser. It takes Benaloh's *Simple Verifiable Voting* [29] as base. It is a centralized system that attempts to provide the two most basic and fundamental properties, integrity and privacy. The author prioritizes absolute integrity over the secret ballot in the worst case. Helios deliberately ignores the problem of coercion, arguing that it is an inherent problem of REV proposals. The verification processes are interactive, that is, the voter's guarantees do not depend on the amount of information sent, or the information publicly available. Instead, the verification is based on the outcome of the voter's interaction with the central Helios server and/or with the different participants of the *mixnet*.
- **Civitas/JCJ:** Civitas/JCJ is another approach to remote voting, which employs anti-coercion measures, through duplicate identities [30,31]. During the registration phase, a valid ID is obtained, in addition to other fake credentials. These fake credentials, which must be removed from the final count, are able to confuse a potential attacker as they are indistinguishable from the real one. In order to satisfy this complicated requirement, this system relies on loose assumptions. This is, voters have to place

more blind trust because its threat model is more permissive. Under a more critical analysis, these more relaxed assumptions are directly against the property discussed previously: strong and blind voter guarantees. The initialization phase is distributed, while vote counting and publication is done from the central server. Like Helios, vote privacy is based on a *mixnet* that obfuscates the origin of each virtual ballot, encrypted by the user.

Additionally, the most recent research has tried to refine both proposals. For example, we can mention:

- **Ordinos** [32]: Ordinos is an improvement to the Helios voting system. Its most prominent feature is called 'tally-hiding'. Through the use of a multi-party computation network, the system exposes a result calculated from the tally, instead of the tally itself. Like Helios, it is verifiable end-to-end, accountable, and it trusts the central Helios server.
- **Incoercible fully-remote electronic voting protocol** [33]: This system is not exactly based on JCJ/Civitas. It takes it as an inspiration, while the authors address its recognized problems and propose a new alternative that overcome said vulnerabilities. However, we can still perceive relaxed assumptions.

3. Adversarial and Threat Model

In any system, from a security point of view, it is crucial to specify the participants, their motivations, and what malicious behaviour is expected from them. Once we have established the desirable properties from our system and the threat model, we can analyze the proposed system, its strengths and vulnerabilities.

In our e-voting proposal, there are three different types of participants.

- **Administration:** It is the organization that manages the entire voting process. It would correspond, for example, to the government in the case of a national or regional election. We can expect the administration to be dependent on one or more political parties in power at the time of the election. Their main interest is that the voting process develops correctly and that the results are valid, since all users can verify their vote. Moreover, considering that this organization may not be independent, we have to consider a malicious partisan interest. It could try to manipulate the result in its favor with its special privileges.
- **Parties:** Each political party participates in the electoral process with a server that collects part of the fragments of information needed to later reconstruct the result. It is to be expected that the parties will have conflicting interests, for selfish reasons in a competitive environment, and for differences in their doctrines. Perhaps, in some cases, there may be complicity between two parties close in the ideological spectrum, although experience reveals that competition for votes is the priority during elections, unless these two formations present a joint candidacy (in which case, for technical purposes in this model, they are counted as a single party). Joint or independent corruption of the total number of parties is not considered viable. Their main motivation is to win as many votes as possible. In the balance between the struggle of interests from all parties, the security of the vote is ensured. Their possible attacks are based on discovering the vote of some users, buying their votes, or modifying the votes during the voting as much as possible so that the result is favorable to them. The rest of the parties will not accept attacks, since their interest is also to protect their votes.
- **Voters:** They are the users of the system, registered in a census, who have the right to vote (only once) in the voting system. They are the least privileged participants within the system and they do not hold responsibilities. This is the largest group, and at the same time the most varied and unpredictable. The margin of action they may have is reduced to a minimum.

Their primary legitimate interest is to vote, and then verify their participation in the final tally. However, malicious voters may try to boycott the election or vote more than once to gain an illegal advantage for their political choice.

It is important to emphasize the fact that political parties and the administration are public entities, and are subject to public scrutiny. This means that there are two levels of security measures against attacks coming from the administration or from political parties (which are the privileged agents of the system, since voters barely have attack margin). At the same time, we can actively try to block attacks and try to make them visible. The moment one of these entities attempts an attack to manipulate the vote, it will be detected by the rest of the actors. The parties and the administration must maintain a positive image in the eyes of the citizens. The empirical and demonstrated exposure of an attack coming from a public entity would terribly undermine their credibility.

4. The DiverSEC Proposal

Here we describe our remote e-voting proposal in detail, with a section for each of its stages: initialization, vote generation, authentication and casting, *mixnet*, recomposition and publication.

4.1. Initialization

In the first phase of the protocol, the administration establishes all the parameters and publishes all the information to the rest of the participants. We will list all the actors involved, the parameters that are published and their notation.

- **Parties** $\{p_1, p_2, p_3, \dots\}$: P parties participate in the voting process, noted as $p_i, 1 \leq i \leq P$. Their network addresses IP_{p_i} and all their public keys (that have been generated for this system) K_i are published. Every party has been assigned a public coordinate in the X axis. This parameter is important for the purposes of Shamir's secret sharing scheme. From now on, and without loss of generality, we will assume that for every $p_i, x = i$.
- **Options** $\{0, 1, 2, 3, \dots\}$: There are O options to vote on. This number of options is necessarily a prime number. If the number of options required by the situation is not a prime number, then the next higher prime is selected by adding blank padding votes. The options include the blank vote and all parties as a minimum. Additionally, it is possible to add other alternative options. The list, known at initialization time, includes the numbers in \mathbb{F}_O along with the option each one of them represents. For example, this is a possible list for $O = 7$:
 - 0: Blank vote
 - 1: Party A
 - 2: Party B
 - 3: Party C
 - 4: Party D
 - 5: Alternative E
 - 6: Blank vote
- **Voters** $\{v_1, v_2, v_3, \dots\}$: Citizens who have the right to cast a single vote and to verify the result later. They are identified by their public key, within the current PKI. The census is public, as it happens in a traditional election, linking voter and public key. V is the number of registered voters, and $v_j, 1 \leq j \leq V$ is the denomination of each voter. Each voter's public key is $K_{pub,j}$.
- **Mixnet sequence (M)**: The administration establishes an order for the parties, or in other words, a distribution of the set $\{1, 2, \dots, P\}$. Any permutation is equally valid, but we assume from now on that an ascending numerical order is chosen for its simplicity. M is, in fact, a cycle, since the last party of the sequence will precede the first. We denote by M_i the sequence M shifted so that the last element is i .

- **Authentication challenge (R_{au}):** Random prefix that will serve as a challenge in the authentication phase, to avoid replay attacks.
- **Recomposition challenge (R_{rc}):** Random prefix that will serve as part of the random number that will later be used to recombine the fragments of each vote.

All these parameters must be communicated to each voter, personally or through a public ‘board’. The method used is irrelevant. From now on these elements are assumed to be common knowledge. Figures and examples will be used to illustrate the procedure graphically, with four parties and five options, hence $P = 4, O = 5, M = \{1, 2, 3, 4\}$.

4.2. Vote Generation

In order to cast a vote, each voter must generate a polynomial that reflects her or her voting intention. This phase is divided in two parts, the generation of the polynomial and the packaging.

We work on the finite field defined by the elements of $\mathbb{F}_O, \text{ mod } O$. The voter generates a random polynomial of degree $P - 1, f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{(P-1)}x^{(P-1)}$, whose threshold is $k = P$. The free term a_0 represents the option chosen by the voter, according to the published list of options. The rest of the terms $\{a_1, a_2, \dots, a_{(P-1)}\}$ are random integers in the finite field \mathbb{F}_O . Fixed the polynomial $f(x)$, all the shares $f(i)$ are calculated, one for each party, $1 \leq i \leq P$. An example for the polynomial $f(x) = 2 + 3x + x^2 + 3x^3$ is illustrated in Figure 1.

$$f(0) = 2; f(1) = 2 + 3 + 1 + 3 = 4; f(2) = 2 + 1 + 4 + 4 = 1; \dots$$

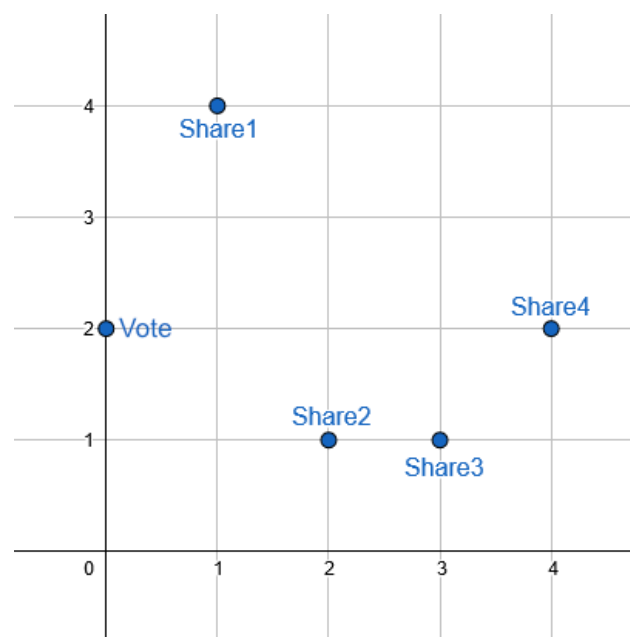


Figure 1. Obtaining shares from the generated polynomial.

With the computed shares, the voter generates a recomposition nonce using the prefix and a random private part generated by each voter $j: N_{rc,j} = R_{rc} \parallel Rand_j$. The voter must remember her $N_{rc,j}$ in order to verify her vote later. Then, the shares are encoded and encrypted as follows.

1. The share and the nonce are concatenated: “ $f(i) = \dots, N_{rc} = \dots$ ”
2. The shifted *mixnet* sequence M_i is obtained, so we can iterate from the last element i to the first.
3. For each index m , the voter takes the current state of the share and the nonce and encrypts it for the party p_m with its public key.

- During each iteration, the voter saves a hash of the current state of the packet of share and nonce, from plaintext until it has been encrypted P times.

After this process, each point $f(i)$ of the polynomial attached to N_{rc} has been encrypted using the public keys of all parties. The encryption order is determined by each shifted sequence M_i . The result is called SC_i . In our example scenario with four parties, the sequence M_2 used to encrypt the second share, would be:

$$M = \{1, 2, 3, 4\}, M_2 = \{3, 4, 1, 2\}$$

$$SC_2 = \left\{ \left\{ \left\{ \left\{ f(2) = 1, N_{rc,j} = d5a3bb \dots \right\}_{K_2} \right\}_{K_1} \right\}_{K_4} \right\}_{K_3},$$

where $N_{rc} = d5a3bb \dots$ represents the random nonce.

The voter must store an array of size $P(P + 1)$. This data will be used to verify the correct transmission and decryption of each share through the network. Figure 2 shows an example for a voter Alice.

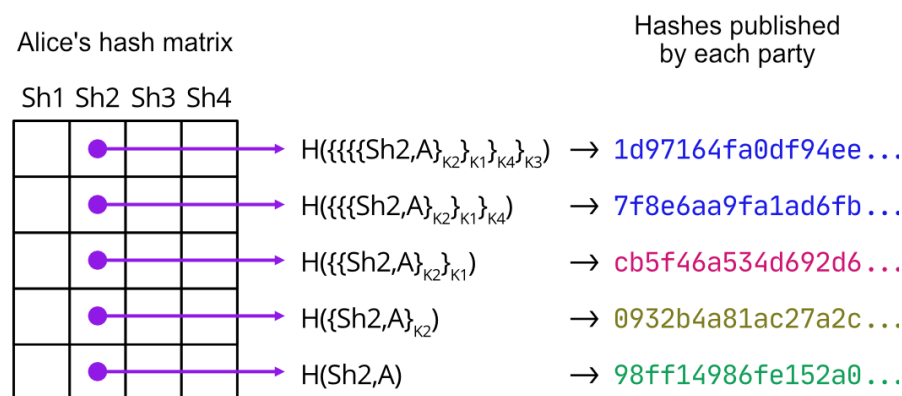


Figure 2. Hashes corresponding to voter Alice, for mixnet stage verification.

4.3. Authentication and Casting

Authentication is performed via public key, since the voter census includes their public keys and has been disclosed in the first phase. A process analogous to the public-key method of the SSHv2 protocol, documented in RFC4252 [34], is used. In this system, the voter digitally signs her intention to vote, along with her name, her public key, and the challenge R_{au} . This signature is sent to each of the parties and the administration.

Once the voter is authenticated, parties then accept only one encrypted share from each voter. The voter distributes her information as follows. Each point $f(i)$ must reach the party p_i , which is in charge of the position $x = i$. For this reason, the voter takes each encrypted share SC_i and observes its order M_i , which ends at the party p_i . The first element of M_i is the start of the chain, so the voter must send each SC_i to the first index of that list (Figure 3).

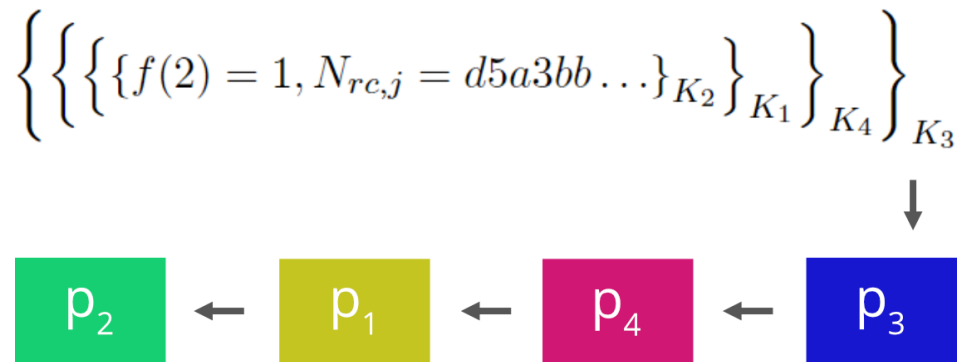


Figure 3. Each SC_i traveling through the node chain.

4.4. Mixnet

The *mixnet* phase begins when all voters have cast their shares. Of the total census, of size V , only a subset consisting of V' voters, where $V' \leq V$, participates. It is to be expected that not all voters use their right to vote. This subset is known, moreover, given the authentication signatures, which are shared by all parties and the administration.

Before starting, all parties confirm the reception of V' encrypted shares and different voter signatures. They publish a list of hash of all the SC_i they own, for voter verification. From now on we note as a hash-list the result of calculating the hash of each of the encrypted shares held by a party at a given time, in ascending order with respect to the hash, instead of the origin. This detail is critical; the order of the hash-list does not provide any information about the order of the preimages. Then, the rounds of the *mixnet* begin:

- Each party calculates the hash-list of all the encrypted shares received, and compares it with the hash-list published by the previous node, to make sure that there has been no manipulation. The first round does not rely on this check, since the previous node is not a party but the voters, and they have already checked the hash-list at the beginning.
- Each party decrypts all shares received with its private key, removing one of the encryption layers.
- Each party calculates the hash-list after decryption, and publishes it for verification by the voter the following party.
- Unless it is the last round (the last encryption layer has been removed), each party randomizes the order of its encrypted shares, and sends them to the next node, according to the cyclic order defined by M .

After P rounds, every encryption layer is removed, and each party p_i has V' shares in plaintext. Figure 4 shows this process by isolating the shares of party p_2 . Five voters cast their SC_2 to p_3 , and reach their destination after passing through all parties. Each one decrypts the layer it can, publishes the hash-list and transmits their data to the next party, randomizing its order. In each published hash-list, an entry has been highlighted. Figure 2 shows a column of the matrix that Alice has saved in a previous phase. Alice can verify that, in each of the published lists, her share is in the state it should be. As a result, her share has not been tampered with or deleted.

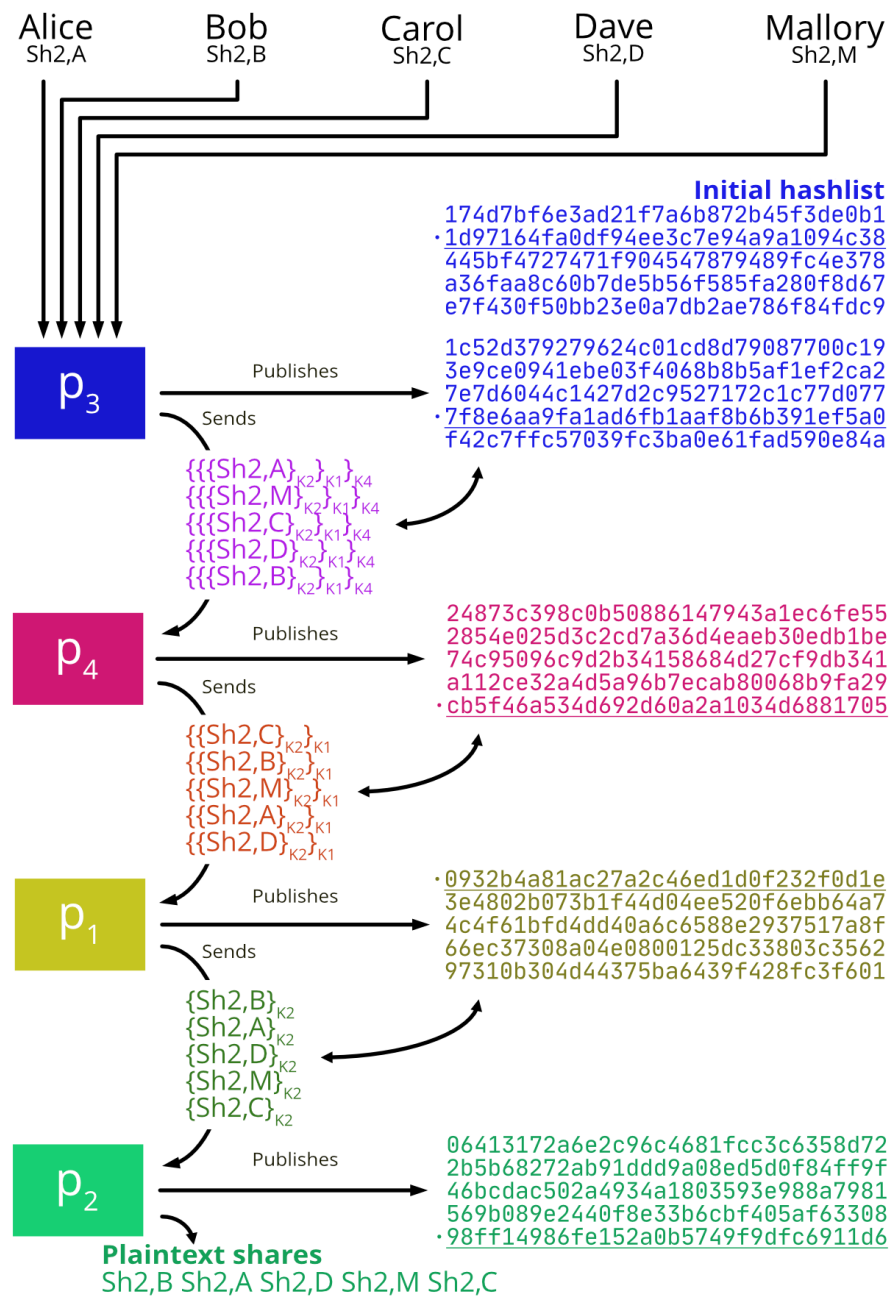


Figure 4. Mixnet example isolating shares from p_2 .

4.5. Recomposition

Each party has, at this point, a list of V' nonces $N_{rc,j}$. Voters themselves may attempt to attack the integrity of the election. Therefore, the list of nonces for each party is published. Those entries that do not include the required prefix R_{rc} are removed, and those that do not appear on all lists are removed. The rest, which are valid, are sorted according to a common rule. For simplicity, we have assumed an ascending order.

In Figure 5, we show how the nonces for our example have been ordered, and one inconsistent nonce, sent by Mallory, has been removed. Voters know their nonce, but no one else can link nonce to voter.

	p_1	p_2	p_3	p_4
0: Bob	00f6bc	00f6bc	00f6bc	00f6bc
1: Dave	8b7785	8b7785	8b7785	8b7785
2: Alice	92d58d	92d58d	92d58d	92d58d
3: Carol	cbe568	cbe568	cbe568	cbe568
Mallory	e556e7			
		fe1d89		
			dd3451	
				da4a56

Figure 5. Ordering of the nonces.

From now on, we work on the finite field $\mathbb{F}_{O^{V'}}$. Being a finite field which cardinality is the power of a prime number, the elements of the set are not numbers anymore, they are polynomials, according to the theory of finite fields [35]. In this case, the polynomials that form this field are of degree $V' - 1$ and the coefficients are integers mod O . We will call them sum polynomials $s(\alpha) = t_0 + t_1\alpha + t_2\alpha^2 + \dots + t_{(V'-1)}\alpha^{(V'-1)}$. We will denote as $s_i(\alpha)$ the sum polynomial of each party p_i . Each of them composes its sum polynomial as follows: Taking the ordered list, each party $p(i)$ determines all coefficients $(t_0, t_1, t_2, \dots, t_{(V'-1)})$ with all points $f(i)$ received, ordered according to their nonce. In other words, t_0 will correspond to the $f(i)$ whose nonce is first in the list, t_1 will correspond to the $f(i)$ whose nonce is second in the list, and so on.

4.6. Publication

After computing the polynomial sum $s_i(\alpha)$, each party publishes the hash digest of that polynomial, using random padding to avoid precalculation attacks. The space of possible polynomials is $O^{V'}$. Therefore, given a large number of choices and voters who have participated, an attack based on the precalculation of all $H(s(\alpha))$ may be unapproachable. However, adding random padding provides extra security without drawbacks.

Once every polynomial hash $H(s_i(\alpha))$ has been published, the parties publish the sum polynomial in plaintext. It is possible to check that the hash and the polynomial match, and in this way we avoid asynchronous publication attacks, this is, that the last of the parties can observe the information of the rest to elaborate an illegal $s_i(\alpha)$, which does not correspond to reality, in order to tamper the results. In Figure 6 we see an example of sum polynomials.

The voting result can be checked by all actors in the system at this point. We denote as $u(i)$ the polynomial of degree 0 whose free term is i . By interpolation, we must compute the polynomial $R(x)$, given the points $\{R(u(i)) = s_i(\alpha), 1 \leq i \leq P\}$. The result of the voting process is found in $R(0)$, which is a polynomial of degree $V' - 1$, with $V' - 1$ coefficients. Each of the coefficients, integers of the field \mathbb{F}_O , is a vote, according to the table of options defined from the initialization. According to the example in the figures, we look for a $R(x)$ such that:

$$\begin{aligned}
 R(1 + 0\alpha + 0\alpha^2 + 0\alpha^3) &= 3 + 1\alpha + 4\alpha^2 + 2\alpha^3 \\
 R(2 + 0\alpha + 0\alpha^2 + 0\alpha^3) &= 1 + 0\alpha + 1\alpha^2 + 2\alpha^3 \\
 R(3 + 0\alpha + 0\alpha^2 + 0\alpha^3) &= 4 + 1\alpha + 1\alpha^2 + 3\alpha^3 \\
 R(4 + 0\alpha + 0\alpha^2 + 0\alpha^3) &= 3 + 2\alpha + 2\alpha^2 + 2\alpha^3
 \end{aligned}$$

By interpolation, the polynomial of degree 3 we are looking for is:

$$\begin{aligned}
 &(4 + 1\alpha + 2\alpha^2 + 1\alpha^3) \\
 &+(4 + 4\alpha + 3\alpha^2 + 3\alpha^3)x \\
 &+(4 + 3\alpha + 1\alpha^2 + 1\alpha^3)x^2 \\
 &+(1 + 3\alpha + 3\alpha^2 + 2\alpha^3)x^3
 \end{aligned}$$

The result $R(0) = (4 + 1\alpha + 2\alpha^2 + 1\alpha^3)$ reveals that option 1 has obtained two votes, option 2 has obtained one vote, and option 4 has obtained one vote.

The voter can now verify her vote in $R(0)$. First, the voter locates the nonce within the ordered list. If this position is c , the coefficient of the term α^c of the polynomial $R(0)$ must match the voter’s option, that is, the a_0 of the polynomial it generated at the time. In addition, the coefficients of the term α^c of the sum polynomials $s_i(\alpha)$ must match each share sent to each party, $f(i)$. In Alice’s case, she must find the coefficients of α^2 in the sum polynomials $s_i(\alpha)$ and in the result $R(0)$.

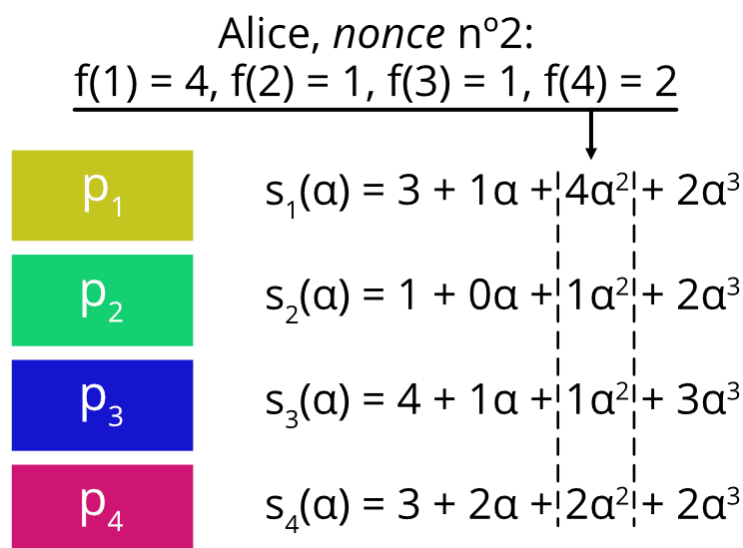


Figure 6. Sum polynomials according to nonces ordering.

5. Results and Discussion of the Contributions

In this section we describe the main results of this work and the ways in which it contributes to the existing state of the art. First, we review the properties achieved by the proposed schema. Then we outline its main strength, and also its current limitations. Finally, we perform a computational analysis of the system.

5.1. Properties Achieved

In Section 2.1, we enumerated the list of properties expected from an e-voting system according to the literature. It is appropriate to review them and check which ones this proposal satisfies and under which assumptions.

- **Ballot secrecy:** In a REV system, this property takes on a double dimension. No participant in the network, which is shared by voters, administration and parties, should know the content of a voter’s ballot. We must decouple the vote from the voter’s identity. This includes the data used to authenticates the voter, but also from any other data which could be traced to a specific voter, such as its IP address. The system must provide a solution to this problem without telling the voter to use a proxy or hidden network service.

The parties are directly involved in the process, collecting the shares, decrypting them layer by layer and sending them to the next node, and then publishing the sum. For this reason, the parties could violate the ballot secrecy, in two ways:

1. Decrypting each voter's vote as it is cast. It seems impossible, if we consider the cipher used to be secure. A single party must know or break the keys of all the other parties and collect every share. Obviously, if the encryption is broken or the keys are violated, there is no privacy.
2. Linking a clear vote to its origin. At the end of the *mixnet* phase, each party receives a point of the polynomial function along with a nonce. If a party is able to identify its origin, either by its IP address or by association with the digital signature that authenticates it, then it would violate the secret ballot. However, to identify the origin of a nonce from the beginning to the end of the *mixnet*, it is necessary that all the participating parties, without exception, are corrupt and conniving with each other. If only one of the parties randomizes the shares without revealing the order, it is impossible to trace any share to its source.

Taking this into account, we claim the system satisfies ballot secrecy under our threat model.

- **Integrity:** The integrity of each vote depends on the *mixnet* phase. Once the nonces and sum polynomials are published, the outcome of the election is determined. However, up to that point, the shares of each user are left vulnerable during the *mixnet* phase. During one of the iterations, a party could alter the shares it owns. The loss of the integrity of one or more shares of each voter results in the elimination of that vote (since the nonces will not match). The corrupt party in question does not know which vote it is manipulating, because of the properties of Shamir's secret sharing. For this reason, it would not be able to predict the outcome either. Assuming he could change one vote for another, the result would be random and there would be no incentive to perform such an attack.
In addition, there are two mechanisms to detect this phenomenon. On the one hand, users can check the hash-list that is published per party and per round during the *mixnet* phase. In this way, the voter is able to identify where an attack has occurred, in which round and which party was the culprit. On the other hand, the parties are forced to send consistent data. This is because, by randomizing and forwarding the shares from party A to party B, party B can check, before decrypting, that the hashes of what it receives correspond to those published by party A. With these two mechanisms, corruption is detected both in the transmission between parties and in the decryption, so there is no way to violate the integrity of the vote without it being detected by the voter.
- **Legitimacy and eligibility:** The total census of voters, with their identity and public key, is known from the beginning of the election process. This detail, coupled with the fact that the proposed system is distributed, prevents a successful attack in which an illegitimate voter participates, or a legitimate voter casts two votes. The length of the signature list and the hash-lists must match for all parties, and must be constant throughout the iterations.
- **Verifiability:** Voters can verify their votes end-to-end against the final tally thanks to the reconstruction challenge. The shares published must be the ones cast, and the vote must match the one cast. In addition, each voter can also verify their hashes during the *mixnet*. This verification reveals which party is the culprit of the attack.
- **Coercion resistance:** This proposal does not implement receipts issued by the administration or by any other means to verify the vote. Instead, it is the voter who generates a random nonce that identifies all the shares. Once these nonces are published and sorted, then there is no possibility of a coercion attack. Prior to this point, however, there is a vulnerability by which a voter could demonstrate her or her vote to a third party. If this voter communicates the nonce used to a third entity, this entity can check whether this nonce is in the ordered list of nonces, and the associated vote.

- **Error tolerance:** There are several expected error cases in this proposed voting system. Although not all of them have an associated troubleshooting protocol, they are detectable in real time. The following cases are considered, ordered according to the phase in which they might occur:
 - Voters using integers outside the associated finite field.
 - Voters sending authentication signatures and/or signatures only to part of the parties.
 - Parties refusing to accept authentication signatures and/or shares.
 - Parties that tamper with shares during the *mixnet* phase.
 - Parties that publish points and/or nonces that do not correspond to what they receive.

5.2. Computational Analysis

The proposed voting system has three main parameters on which the computational complexity depends: the number of parties P , the number of choices O and the number of voters V .

- P : The number of parties determines the number of iterations of the *mixnet* phase. In addition, the voters have to encrypt their P shares, P times each share. The maximum complexity observed with respect to P is quadratic, $O(P^2)$, during the second phase (vote generation), as we will observe in the following analysis.
- O : The number of options determines the finite field in which the users will operate in order to generate their vote and the finite field in which the voting result will be interpolated, \mathbb{F}_O and \mathbb{F}_{OV} , respectively. It does not affect the number of operations done by the actors in our system.
- V : Assuming that all voters participate (the most conservative case), the number of voters affects the finite field \mathbb{F}_{OV} at the time of polynomial interpolation. The voting result is a polynomial with as many terms as voters, as are all polynomials published by the parties. Thanks to the optimization presented in the following analysis, the maximum complexity for this dimension is linear, $O(V)$, found in the last three stages of the protocol.

In order to analyze the feasibility of the system, it is necessary to delimit the expected range of values for each of these parameters. The reality is that in each country, the configuration P , O , and V may change.

- P : The major parties tend to vary within the interval $[2, 10]$, but the total set of parties can be expected to be significantly larger. In any case, it is estimated that it will never exceed 50 parties, and that this is a stable and controlled number.
- O : For simplicity we can assume that the value of O equals $P + 1$. In the most conservative case, all political parties will actively participate.
- V : This is the most critical value for the computation analysis. In the worst case, we can say that there are as many voters as the population of legal age with the corresponding nationality. However, this assumption is rather extreme, since the most typical case will be to fragment the elections in different polling stations, by necessity of the corresponding electoral law or for technical convenience. In any case, it is undoubtedly the largest parameter, and can be expected to be at least 1000 individuals.

Finally, we can list the phases of the system with their computational complexity. For better scalability, we look for a low complexity class, so we can increase the number of voters without making the execution of the process unfeasible. Typically, linear complexity is usually desired.

1. **Initialization:** During the initial phase, the configuration is established along with all the necessary elements, including addresses and public keys, are made public. This phase does not require computational power.
2. **Vote generation:** Each voter generates a polynomial, computes the different shares and packs them for distribution. The generation of a polynomial involves generating

$P - 1$ random coefficients, and computing P points of the corresponding polynomial function. These two operations are independent, so a linear complexity is observed $O(P)$. The second part involves encrypting each of these points for all parties. That is, each of these P points is encrypted P times. The complexity in this case is quadratic, $O(P^2)$.

3. **Authentication and distribution:** Each voter distributes authentication signatures and her or her shares. The complexity is linear $O(P)$.
4. ***mixnet*:** The mixing network involves as many rounds as there are layers of encryption applied to the shares, therefore, the number of rounds equals P . Each round is independent. The operations that occur in each round are decryption, randomization, and computation of the hash of all the shares they manage. The complexity is linear in these two dimensions, $O(P)$ and $O(V)$.
5. **Reconstruction:** The reconstruction of the polynomial sum of each party is, in short, to order some coefficients according to the consensus list. The complexity is linear with respect to the number of voters $O(V)$.
6. **Disclosure:** This is the most computationally complex phase. Anyone can interpolate the polynomial of degree $P - 1$ into the finite field \mathbb{F}_{O^V} that fits the values provided by the parties. The complexity of the polynomial interpolation itself depends on the algorithm used, especially if we take into account that we do not need the whole polynomial, but the independent term. Shamir himself states in her proposed scheme for shared secret [25] that there are algorithms of complexity $O(P \log^2(P))$ for polynomial interpolation. On the other hand the concern about operations on the finite field \mathbb{F}_{O^V} can also be solved, by translating each operation with polynomials of degree $V - 1$ to V independent operations on the field \mathbb{F}_O , whose elements are integers mod O . In this sense, the complexity is linear $O(V)$.

The computational analysis reveals that the highest complexity observed is quadratic, regarding the number of parties in the vote generation phase, when encrypting the points obtained. This was to be expected, since the voter must obtain a matrix of hashes for subsequent verification. However, the number of parties is going to remain small, and this is not expected to be any problem for any modern CPU.

The other problematic aspect is the use of operations with very high degree polynomials, $V - 1$. The parameter that can grow the most is that of voters. In fact, a high number of these are expected. However, the complexity is linear with respect to this parameter thanks to the ability to independently interpolate each of the coefficients of the voting result.

5.3. Strengths and Limitations of the Proposal

The main strengths of the DiverSEC system can be summarized as follows:

1. The addition of secret sharing implies that the system is distributed. Distributed systems have added complexity, but that complexity creates, in return, a security tension between nodes, where parties cannot perform attacks without being detected by each other. In a centralized model, the different voting authorities could perform attacks due to their privileges, and there are only two options against this approach: prevent the attacks or trust the authorities. After distributing those privileges among independent actors, these attacks are no longer possible. In our case, each *mixnet* node knows just one part of the information. For this reason, their attack surface is greatly reduced.
2. The privileges of this system are not only distributed, they are distributed among entities with clearly competing interests. For this reason, they will make sure to report attacks from the rest of the nodes. On top of that, political parties are public entities, subject to scrutiny and judgment by the voters.
3. Attacks are prevented twofold. First, attacks are detected in real time, thanks to the verification processes. Both political parties and voters are checking the hashes in every step of the *mixnet*. It is impossible to tamper a vote without being detected as long as the chosen hash function remains secure against collisions. Second, it

is impossible to perform targeted attacks. During the *mixnet* phase, the shares are encrypted, so the nodes do not know the content nor the voter of any share. Replacing the content with a new generated share is not possible, since the attacker ignores the recomposition nonce. Even if we ignore these facts, in the Shamir's Secret Sharing Scheme, it is not possible to craft a malicious share that gives a desired arbitrary output (as long as we use a commitment scheme to publish the shares).

4. In this case, verification is based on a very simple process, based on public information. Each voter can verify the processing of her shares by comparing her stored hashes against the list published by each node during the *mixnet*, and can verify her vote end-to-end by identifying her private nonce and the matching plaintext shares and secret. This verification is not interactive and is persistent over time.

It is also important to mention that the proposed scheme is easy to implement. With the computational complexity analysis described above, we can see that a current standard server machine is able to process all the information, given the operations that we have listed until now. Any current end-user device is enough for the voter tasks, such as generation, distribution, and verification. The computational complexities are linear for the number of voters (which is the clearly the higher dimension of a voting system) and quadratic for the number of parties. However, if the amount of voters grew to be a problem, this scheme permits easily distributing the election. This approach is similar to the current election system, which distributes the complete census in different electoral districts.

One of the current concerns in designing crypto-based systems is the impact that the advance of quantum computing may have in them, so this is an issue that must be taken in consideration when proposing a new system [36]. As far as our e-voting system is concerned, the Shamir secret sharing scheme should not be impacted by the advances on quantum computing. Traditional asymmetric cryptography primitives such as RSA, however, are expected to be threatened by these advances. Fortunately, our system is flexible in the use of cryptographic primitives, so it does not need any specific primitive for encryption or signing, and thus could be easily adapted to any quantum-safe standard used at the time of implementation. Likewise, the public key management mechanisms or bulletin board implementations (i.e., [20] or [37]) are not specified either.

Regarding limitations, the main problem with this proposal is the ability of the voters to prove their vote accurately to a third party, in the lapse of time between the generation of the vote and its publication of the tally. This enables voluntary vote buying, or forced coercion.

Secondly, the proposal has no real-time troubleshooting protocols to correct integrity attacks. The current proposal can only detect them, and expose the attacker.

Finally, there are some aspects of an actual e-voting system which are out of the scope of our proposal. For asymmetric encryption to be usable in practice, a PKI infrastructure must be in place for citizens and political parties. In a similar way, actual device protection is out of the scope of our work, but there are plenty of security mechanisms that could be used to protect the computation if needed, such as security enclaves like the ones used in [36]).

6. Conclusions and Future Work

In this work, we propose a novel distributed, remote e-voting system (REV) for elections. Our proposal involves three novel elements in addition to the techniques typically used in the state of the art:

1. Shamir's secret sharing scheme. Its use has generally been limited because of its integrity vulnerabilities. Only a very recent proposal [26] uses it, but it solves this problem in a completely different way: it employs DRE machines under the assumption that those machines are secure.
2. Galois' field. Shamir's secret sharing needs finite field operations in order to eliminate one of its vulnerabilities, but it has not previously been exploited in order to limit the possible values to vote on. The size of the field was traditionally irrelevant, and a

sufficiently large prime number was chosen. In this case, in our field \mathbb{F}_{Q_V} neither the base nor the exponent are arbitrary, but correspond to voting parameters.

3. The proposal of a distributed system in which the parties themselves form the voting network. Since we are confronting the selfish interests of each political party against the rest, this creates a positive balance that empowers our security model. It is the electronic equivalent of having supervisors from every party inspecting the ballot boxes in person.

The designed system satisfies the main concerns and needs of REV systems. In an exhaustive review of the list of desirable properties, this system fulfills all but coercion resistance, which is one of the biggest problem in the current state-of-the-art.

The two most critical dimensions, ballot privacy and integrity are covered in depth. It is to be expected that vulnerabilities will arise that undermine the security offered in any protocol, system, or cryptographic primitive. For this reason, multi-layer security provides a shield against these undiscovered vulnerabilities. Encryption protects confidentiality, but at the same time, each vote is divided into different shares, which do not reveal information about the secret unless a single entity receives every single share in plaintext. Publication of the verification hashes in turn protects integrity and, at the same time, removes the incentive for parties to manipulate the shares they manage.

The non-technical dimension is also important. The vote verification process is extremely simple, accessible to any non-technical voter. This detail is critical for the population to accept the guarantees offered by this system, as opposed to others that are just as good but are extremely complex.

It should be mentioned that the threat model and the system assumption are realistic. On the contrary, other proposals are secure only under very lax criteria.

In summary, the proposal is considered to add value to the current state of electronic voting research. It is also flexible and easy to extend, so additions such as failure recovery protocols or as security layers against coercion could be easily added to this base.

Indeed, the two most immediate priorities for improving this proposal are:

- An exhaustive search for a solution to the coercion problem, to fully cover the list of desirable properties.
- Defining automatic protocols for real-time corruption detection and resolution, according to the cases we have mentioned previously in the analysis. The means for voters to report any error or manipulation are already provided, but the real-time resolution protocol is yet to be defined.

Once both requirements are met, this proposal satisfies all the current needs of e-voting systems. From this point, it is possible to think about developing an implementation, for a real future use. This would require a study on the platforms to be used, both on the server and client side, and on the scope of adoption.

Author Contributions: All authors have made relevant contributions to this work. The proposal of the problem was proposed by M.T.-R. and developed in numerous meetings involving D.O., J.J.-S., J.M.G.-G. and I.M.-M., who later led different parts of the work. Although all the authors actively participated in all tasks including writing and editing the manuscript, the theoretical part of the paper was led by M.T.-R. and D.O. All authors have read and agreed to the published version of the manuscript.

Funding: M.T.-R. is funded by a predoctoral grant from the University of Alcalá. M.T.-R., I.M.-M. and J.M.G.-G. are partially funded by Project PID2019-104855RB-I00/AEI/10.13039/501100011033 of the Spanish Ministry of Science and Innovation, by Project SBPLY/19/180501/000171 of the Junta de Comunidades de Castilla-La Mancha and FEDER and by Project UCeNet (CM/JIN/2019-031) of the Comunidad de Madrid and University of Alcalá. D.O. is partially supported by Project PID2019-104129GB-I00/AEI/10.13039/501100011033 of the Spanish Ministry of Science and Innovation, and by H2020-MSCA-RISE project 734922—CONNECT.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gibson, J.P.; Krimmer, R.; Teague, V.; Pomares, J. A Review of E-Voting: The Past, Present and Future. *Ann. Telecommun.* **2016**, *71*, 279–286. [CrossRef]
2. Madise, Ü.; Martens, T. *E-Voting in Estonia 2005. The First Practice of Country-Wide Binding Internet Voting in the World*; Gesellschaft für Informatik e.V.: Bregenz, Austria, 2006.
3. McDaniel, P.; Blaze, M.; Vigna, G. EVEREST: Evaluation and validation of election-related equipment, standards and testing. In *Ohio Secretary of State's EVEREST Project Report*; Office of the Ohio Secretary of State: Columbus, OH, USA, 2007.
4. Jones, D.; Simons, B. *Broken Ballots: Will Your Vote Count?* CSLI Publications: Stanford, CA, USA, 2012.
5. Peacock, T.; Ryan, P.Y.A.; Schneider, S.; Xia, Z. Chapter 69—Verifiable Voting Systems. In *Computer and Information Security Handbook*, 2nd ed.; Vacca, J.R., Ed.; Morgan Kaufmann: Burlington, MA, USA, 2013; pp. 1103–1125. [CrossRef]
6. Adida, B. Helios: Web-Based Open-Audit Voting. *USENIX Secur. Symp.* **2008**, *17*, 335–348.
7. Yang, X.; Yi, X.; Kelarev, A.; Han, F.; Luo, J. A distributed networked system for secure publicly verifiable self-tallying online voting. *Inf. Sci.* **2021**, *543*, 125–142. [CrossRef]
8. Rivest, R.L. The Threeballot Voting System. 2006. Available online: <https://dspace.mit.edu/handle/1721.1/96593> (accessed on 6 December 2021).
9. Popoveniuc, S.; Hosp, B. An Introduction to Punchscan. In *IAVoSS Workshop on Trustworthy Elections (WOTE 2006)*; Robinson College United Kingdom: Cambridge, UK, 2006; pp. 28–30.
10. Chaum, D.; Essex, A.; Carback, R.; Clark, J.; Popoveniuc, S.; Sherman, A.; Vora, P. Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting. *IEEE Secur. Priv.* **2008**, *6*, 40–46. [CrossRef]
11. Chaum, D.L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **1981**, *24*, 84–90. [CrossRef]
12. Reed, M.G.; Syverson, P.F.; Goldschlag, D.M. Anonymous connections and onion routing. *IEEE J. Sel. Areas Commun.* **1998**, *16*, 482–494. [CrossRef]
13. Querejeta-Azurmendi, I.; Arroyo Guardado, D.; Hernández-Ardieta, J.L.; Hernández Encinas, L. NetVote: A Strict-Coercion Resistance Re-Voting Based Internet Voting Scheme with Linear Filtering. *Mathematics* **2020**, *8*, 1618. [CrossRef]
14. Islam, N.; Alam, K.M.R.; Tamura, S.; Morimoto, Y. A new e-voting scheme based on revised simplified verifiable re-encryption mixnet. In *Proceedings of the 2017 International Conference on Networking, Systems and Security (NSysS)*, Dhaka, Bangladesh, 5–8 January 2017; pp. 12–20.
15. Chaum, D. Blind signature system. In *Advances in Cryptology*; Springer: Berlin/Heidelberg, Germany, 1984; p. 153.
16. Kumar, M.; Katti, C.P.; Saxena, P.C. A secure anonymous e-voting system using identity-based blind signature scheme. In *Proceedings of the International Conference on Information Systems Security*, Mumbai, India, 16–20 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 29–49.
17. Darwish, A.; El-Gendy, M.M. A new cryptographic voting verifiable scheme for e-voting system based on bit commitment and blind signature. *Int. J. Swarm. Intel. Evol. Comput.* **2017**, *6*, 2. [CrossRef]
18. Rivest, R.L.; Adleman, L.; Dertouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–180.
19. Yang, X.; Yi, X.; Nepal, S.; Kelarev, A.; Han, F. A secure verifiable ranked choice online voting system based on homomorphic encryption. *IEEE Access* **2018**, *6*, 20506–20519. [CrossRef]
20. Yang, X.; Yi, X.; Nepal, S.; Kelarev, A.; Han, F. Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities. *Future Gener. Comput. Syst.* **2020**, *112*, 859–874. [CrossRef]
21. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, 21260.
22. Pawlak, M.; Ponsizewska-Marañda, A. Trends in blockchain-based electronic voting systems. *Inf. Process. Manag.* **2021**, *58*, 102595. [CrossRef]
23. Jafar, U.; Aziz, M.J.A.; Shukur, Z. Blockchain for Electronic Voting System—Review and Open Research Challenges. *Sensors* **2021**, *21*, 5874. [CrossRef] [PubMed]
24. Nair, P.R.; Dorai, D.R. Evaluation of Performance and Security of Proof of Work and Proof of Stake using Blockchain. In *Proceedings of the 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Tirunelveli, India, 4–6 February 2021; pp. 279–283.
25. Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
26. Liu, Y.; Zhao, Q. E-Voting Scheme Using Secret Sharing and K-Anonymity. *World Wide Web* **2019**, *22*, 1657–1667. [CrossRef]
27. Cascudo, I.; David, B. Albatross: Publicly attestable batched randomness based on secret sharing. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, Daejeon, Korea, 7–11 December 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 311–341.
28. Gupta, K.D.; Rahman, M.L.; Dasgupta, D.; Poudyal, S. Shamir's Secret Sharing for Authentication without Reconstructing Password. In *Proceedings of the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 6–8 January 2020; pp. 0958–0963.
29. Benaloh, J. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. *EVT* **2007**, *7*, 14.
30. Neumann, S.; Feier, C.; Volkamer, M.; Koenig, R. Towards A Practical JCY/Civitas Implementation. In *INFORMATIK 2013*; Horbach, M., Ed.; Jahrestagung der Gesellschaft für Informatik: Koblenz, Germany, 2013.
31. Juels, A.; Catalano, D.; Jakobsson, M. Coercion-Resistant Electronic Elections. In *Towards Trustworthy Elections*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 37–63.

32. Küsters, R.; Liedtke, J.; Müller, J.; Rausch, D.; Vogt, A. Ordinos: A Verifiable Tally-Hiding E-Voting System. In Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS P), Genoa, Italy, 7–11 September 2020; pp. 216–235. [[CrossRef](#)]
33. Neji, W.; Blibech, K.; Rajeb, N.B. Incoercible fully-remote electronic voting protocol. In Proceedings of the International Conference on Networked Systems, Marrakech, Morocco, 17–19 May 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 355–369.
34. Ylonen, T.; Lonvick, C. *The Secure Shell (SSH) Authentication Protocol*; Technical Report; RFC 4252; IETF: Fremont, CA, USA, 2006.
35. Judson, T. *Abstract Algebra: Theory and Applications*; Virginia Commonwealth University Mathematics: Richmond, VA, USA, 2009.
36. Nguyen, V.L.; Lin, P.C.; Cheng, B.C.; Hwang, R.H.; Lin, Y.D. Security and privacy for 6G: A survey on prospective technologies and challenges. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 2384–2428. [[CrossRef](#)]
37. Culnane, C.; Schneider, S. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In Proceedings of the 2014 IEEE 27th Computer Security Foundations Symposium, Vienna, Austria, 19–22 July 2014; pp. 169–183. [[CrossRef](#)]