INFORMATICA, 2026, Vol. 37, No. 2, 349–382

© 2026 *Vilnius University*
DOI: https://doi.org/10.15388/25-INFOR616

DiReCT Voting: Dispute Resolution with Cast Timeliness

Jose Luis MARTIN-NAVARRO[1,2], Antonio M. LARRIBA[1], Damián LÓPEZ[1,*]

[1] *VRAIN – Valencian Research Institute for Artificial Intelligence, Universitat Politècnica de València, Spain*

[2] *Aalto University, Finland*

e-mail: dlopez@dsic.upv.es

Received: June 2025; accepted: December 2025

Abstract. Verification in modern e-voting protocols allows voters and the general public to independently confirm the elections results. However, verification alone is insufficient to hold entities accountable for misconduct, or to protect honest participants from false accusations. This limitation is especially critical in voting protocols with multiple authorities, where the ability to identify the specific misbehaving entity is essential. We present DiReCT, the first multiparty protocol that integrates dispute resolution with individual accountability. Our protocol addresses two previously unresolved disputes: authorities blocking access to the election; and authorities denying the casting of a ballot. In addition, DiReCT improves timeliness, allowing misconducts to be proactively detected during the elections. As a result, voters can identify and recover from attacks that prevent their ballots from being recorded. Notably, DiReCT achieves these capabilities with low trust assumptions on the authorities.

Key words: electronic vote, dispute resolution, verifiability, accountability, secret sharing.

1. Introduction

Electronic voting systems (e-voting) are modernizing electoral processes, improving voter access, and reducing expenses associated with traditional paper-based elections. However, ensuring the security, transparency, and privacy of the election remains a significant challenge. Protocols provide mechanisms and define properties to secure the election process, but the effectiveness of such measures depends directly on the trust model. For example, a protocol may ensure that each voter only votes once by using a central authority. However, this may not be enough to secure the protocol if the central authority can manipulate the results with fraudulent ballots (*ballot stuffing*) or block users from voting (*vote suppression*).

Verifiability plays a crucial role in managing trust within electronic voting systems, as it allows stakeholders to independently confirm the integrity of election results (Adida *et*

*Corresponding author.

al., 2009; Küsters *et al.*, 2020), and mitigate voter distrust in the system (Duenas-Cid, 2024). End-to-end verifiability (E2E) helps voters to independently confirm that their votes were cast, recorded, and tallied accurately, fostering transparency and strengthening confidence in the election outcome.

Although it is a standard feature in voting protocols (Bougon *et al.*, 2022; Crimmins *et al.*, 2023; Kremer *et al.*, 2010), recent studies have identified limitations in E2E-based verifiability. Crimmins *et al.* (2023) argue that eligibility verifiability should be included in E2E. Another limitation is that E2E guarantees the detection of misbehaviour but fails to provide evidence of the detection. Basin *et al.* (2020) proposed dispute resolution to solve the issue, adapting notions of accountability to e-voting. Disputes can arise in situations where voters claim a wrongdoing from an authority, who defends its honesty. Dispute resolution defines how the validity of a claim is decided, without the need for a trusted judge. Instead, the dispute needs to be solved with unambiguous evidence, and verifiable by any third party with only access to the public data.

However, little attention has been paid to dispute resolution and the types of disputes that can arise in e-voting. Furthermore, the timeliness of the verification and accountability happens, at best, during the tally, which serves as a post-mortem evaluation, rather than a proactive process.

This paper proposes DiReCT, a new protocol with low trust requirements for voting authorities, which improves the state of the art in dispute resolution. The design of DiReCT combines secret sharing and blind signatures to provide elegibility verifiability, privacy and a decentralized tally, evolving the SUVS protocol by Larriba and López (2022). We extended it to address disputes caused by the lack of accountability, preventing attacks from corrupt authorities and voters. DiReCT accounts for two disputes introduced by Basin *et al.* (2020) and considers two new disputes on vote suppression and denial of casting. DiReCT is a multiparty protocol, where casting and tallying are performed by candidates with a conflict of interest (Moran and Naor, 2010). We adapt the original definition of dispute resolution to a multiparty protocol, emphasizing individual accountability to prevent any of the authorities from acting surreptitiously. DiReCT also provides a resolution mechanism for denial of casting, allowing voters to detect and recover from the attack in due time.

The main contributions of the paper can be summarized as follows:

- The definition of two new types of disputes: DCert, a dispute related to vote suppression, where the voting authority blocks the voter during the authentication and ballot certification; and DCast, a dispute related to casting, where the voter is blocked from verifying if the casting has been successful.
- The design of DiReCT, a multi-party electronic voting protocol with improved dispute resolution, individual accountability and cast timeliness, which guarantees that voters possess the evidence to resolve disputes before the election's end.
- A detailed threat model that considers *covert adversaries*, as defined by Aumann and Lindell (2010), where entities can misbehave as long as they are not detected. This improves previous results that consider semi-trusted authorities (Basin *et al.*, 2020), assume Honest-but-curious entities (Larriba and López, 2022), or that require trusting the voting authority (Adida *et al.*, 2009; Chaum *et al.*, 2008).

- The security analysis of DiReCT according to the Universal Composability framework by Canetti (2001), proving that the protocol has recorded-as-cast, eligibility verifiability, and the resolution of both new disputes and tally related disputes such as a tally authority removing a ballot from the tally.

The paper is structured as follows: Section 2 introduces the cryptography background related to the protocol; Section 4 covers the voting model, its entities, the security assumptions and properties; Section 5 explains DiReCT and the election process; the security analysis of the protocol is proved in Section 6; the complexity and scalability of the protocol is discussed in Section 7 and, finally, Section 8 summarizes the contributions and future work.

2. Background

This section introduces the fundamental cryptographic primitives as they are used in the base protocol SUVS (2022), namely, Shamir (1979) secret-sharing scheme, and Chaum (1983) blind signatures.

2.1. *Shamir Secret-Sharing Scheme*

The ballot encoding and tally are based on Shamir (1979) (j, d)-secret sharing work. In this scheme a secret C is shared among j participants, but it can only be retrieved if $d+1$ participants collaborate. To share the secret, a polynomial $q(x)$ encodes the secret as its independent term, with randomly chosen coefficients $a_d \cdots a_1$. A set P of random points of $q(x)$ are used as pieces, distributed among the participants (Equation (1))

$$\begin{aligned} &q(x) = a_d x^d + \cdots + a_1 x + C \mod p, \\ &P = \langle (x_1, q(x_1)), \ldots, (x_{d+1}, q(x_{d+1})) \rangle. \end{aligned} \tag{1}$$

To recover C, $d+1$ points are used to interpolate the polynomial of modulo prime p, for example, with the Lagrange (1795) interpolation method, where l_i is the Lagrange Basis Polynomial:

$$q(x) = \sum_{i=0}^{d+1} q(x_i) l_i(x), \qquad l_i(x) = \prod_{k=0, k\neq i}^{d+1} \frac{x - x_k}{x_i - x_k}. \tag{2}$$

2.2. *Blind Signatures*

The protocol uses blind signatures as introduced by Chaum (1983). In the scheme, a provider applies a blind function to some data and sends it to the signer. Both the blind and the corresponding unblind functions are only known for the provider, and the blind output does not leak any information about the data contained. When the signer receives the blinded data, it signs it and sends it back to the data provider. The sign function needs

a corresponding verify function, without the need of sharing the key used for signing. The provider can then apply the unblind function and obtain the signature of the original data, thanks to the commutative property of the signing and blinding methods.

The blind signature scheme provides untraceability: it is not possible to link signed data with the blind data it comes from. The three methods that implement blind signatures are *Blind*, *Sign* and *Unblind*, with sk, pk the corresponding secret and public key of the signer. The methods, as implemented using RSA, are defined as follows:

$$\text{Blind}\left(data, mask^{pk}\right) = data \cdot mask^{pk} \mod n, \tag{3}$$

$$\begin{aligned} \text{Sign}_{sk}\left(data \cdot mask^{pk}\right) &= \text{Sign}_{sk}(data) \cdot \text{Sign}_{sk}\left(mask^{pk}\right) \mod n, \\ &= \text{Sign}_{sk}(data) \cdot mask \mod n, \end{aligned} \tag{4}$$

$$\begin{aligned} \text{Unblind}\left(\text{Sign}_{sk}\left(data \cdot mask^{pk}\right)\right) &= \text{Sign}_{sk}(data) \cdot mask \cdot mask^{-1} \mod n, \\ &= \text{Sign}_{sk}(data) \mod n. \end{aligned} \tag{5}$$

For simplicity, we explain the workflow of the protocol using RSA blind signatures, which generates blind signatures using its partial homomorphic properties under modular exponentiation (Hwang *et al.*, 2003). However, any blind signature scheme such as Pointcheval and Sanders (2016) can be used in its place without affecting the workflow of the protocol, only modifying how the Blind, Unblind and Verify methods are implemented.

3. Related work

Electronic voting protocols can be divided taking into account the cryptographic technique they use to achieve anonymity and verifiability, with the most popular being mix-nets (1981), homomorphic encryption (1978), and blind signatures (1983).

Mix-nets, introduced by Chaum (1981), involve the shuffling and re-encryption of ballots to sever the connection between voters and their votes. In 2002, Jakobsson *et al.* (2002) introduced the RPC mix-net, a method designed to enhance the robustness of voting systems without requiring full correctness proofs. However, generating and verifying these proofs is computationally intensive. Although later work by Furukawa *et al.* (2010) improved the efficiency of this approach, the time required to produce proofs remains a significant limitation.

Homomorphic encryption is a technique used to preserve anonymity in voting systems, introduced by Rivest *et al.* (1978). Existing e-voting systems primarily adopt Paillier encryption (2008) and ElGamal decryption, as defined by Kiayias and Yung (2002), Lee and Kim (2003). Cramer *et al.* (1996, 1997) proposed a new cryptographic protocol for multi-authority environments, enabling the distribution of authority among multiple entities. However, because individual ballots are not decrypted, each ballot must be verified using zero-knowledge proofs. As a result, applying this approach to multi-choice voting systems is challenging due to the significant computational overhead. Nevertheless, it can be combined with other techniques to enhance functionality. For example, Sebé *et al.* (2010)

demonstrates its integration with mix-nets to improve the efficiency of voting systems, and Yang *et al.* (2017, 2018) proposed an homomorphic based voting protocol for ranked choice.

Blind signatures (1983) are used in voting protocols during the voter registration phase, to authorize the ballot of the voter without revealing the content to the authority. The use of blind signatures was first introduced by Fujioka *et al.* (1993), and later, Okamoto (1998) proposed the first practical receipt-free voting scheme suitable for large-scale elections. However, blind signature schemes require the signer to be fully trusted. If the signer is compromised, an attacker could issue and cast arbitrary numbers of ballots, undermining the integrity of the election. Given the limitations, protocols have combined blind signatures with other techniques to limit the power of the authority: Li *et al.* (2009) proposed a verifiable process to identify different attacks on the protocol, and Nguyen Thi and Dang (2013) combined blind signatures with a dynamic ballot to let voters update their ballot without the authority intervention. However, it is the combination of blind signatures with threshold cryptographic techniques (Desmedt and Frankel, 1990) the one that more effectively limits the ballot stuffing possibility of the authority. Juang *et al.* (2002) introduced a scheme supporting distributed authorities by applying threshold methods. The work by Gong *et al.* (2019) uses threshold blind signatures to protect privacy of the voter and guarantee eligibility in a blockchain based implementation. More recently, Larriba *et al.* (2020), Larriba and López (2022) presented a lightweight threshold-based voting system that also achieves public verifiability. Other authors have proposed a similar decentralized solution using ring signatures (Chen *et al.*, 2008; Tornos *et al.*, 2014). Blind signatures have also been discussed in other voting protocols modalities, with a recent work by Willemson (2023) discussing the benefits of using blind signatures to achieve eligibility verifiability in postal voting.

Although Blockchain technology is not a cryptographic primitive, it has been proposed as a supporting technology in the implementation of voting protocols. Such protocols still combine the blockchain with other cryptographic primitives during the voting system. Yang *et al.* (2020) makes use of group signatures to protect the privacy of the voter, and homomorphic encryption to compute the final tally. Gao *et al.* (2019) makes use of ring signatures, and includes an audit mode at the end election, implemented with blockchain, to detect mismatches in the tally. Larriba *et al.* (2021) also uses ring-signatures to protect the privacy of voters, and implements a multi-party voting protocol to increase the trust in the system. For other protocols such as Adida (2008), Larriba and López (2022), although they do not mention blockchain in their design, they still require the use of a public bulletin board, which can be implemented as a blockchain.

Regarding security guarantees, few voting protocols go further than verifiability and offer dispute resolution or individual accountability, given the complexity of introducing properties while keeping the privacy and coercion resistance of the voter (Pankova and Willemson, 2022).

A principal work on dispute resolution was published by Basin *et al.* (2020). The authors introduce three types of disputes, and propose a generic protocol (*MixNet*) that addresses two of them. This is one of the few protocols that incorporate dispute resolution

in its design, although it leaves unresolved the problem of eligibility. Another recent work including dispute resolution is Themis by Bougon *et al.* (2022), which also includes a formal verification of its properties. However, since Themis is an hybrid protocol, the dispute resolution requires the collaboration of a voting officer. Themis achieved partial accountability, which means that for some of the disputes it can detect the issue but not identify the specific adversary. A detailed comparison is included in the next section.

In this work, we extend the blind signature scheme SUVS due to its low computational overhead, eligibility verifiability, and flexibility. Inspired by recent work on dispute resolution, we extend the protocol and address disputes for cases not considered, such as when voters are blocked from participating in the election. In addition, the protocol improves the timeliness of the resolution of arising disputes. The protocol makes intensive use of the public bulletin board, which can be provided by blockchain implementations such as Gong *et al.* (2019).

3.1. *Dispute Resolution*

The concept of dispute is stated by Basin in 2020 as "when a voter claims that the voting authority is dishonest and did not correctly process his ballot while the authority claims to have followed the protocol". Basin *et al.* distinguish between universal and individual verifiability to detect relevant disputes in electronic voting. *Universal verifiability* properties can be ensured only with access to public data (e.g. PBB). They claim that universal verifiable properties do not lead to disputes because any voter or third party with access to the PBB can audit the property. However, *individual verifiability* describes properties that only the voter can check, because she is the only one who knows which ballot has been cast, as well as some other private data. This characteristic leads to the occurrence of disputes. On the one hand, voters are able to detect problems with their ballots but may not be able to prove it to others. On the other hand, voters can accuse an honest authority of misbehaving, which may not be able to prove its honesty. Basin *et al.* (see Table 1) also identify three problems related to individual verifiability:

1. The voter is blocked from casting the ballot. The reasons can be technical or social. This is connected with voter suppression and disenfranchisement in traditional elections. Despite defined, this dispute was left out of the scope by the authors.

Table 1
Dispute resolution comparison in multi-party protocols. For accountability, Basin *et al.* (2020) and SUVS have partial accountability ◑, Themis by Bougon *et al.* (2022) has a mix of individual and partial accountability ◕, and this work has complete individual accountability ●.

	Properties	Basin *et al.* (2020)	Themis	SUVS	DiReCT
Disputes	Vote suppression	◑	○	○	●
	Casting	○	◑	○	●
	Recorded	●	●	●	●
	Not Recorded	●	◑	◑	◑
	Accountability	◑	◕	◑	●

2. The voter is prevented from verifying its ballot after casting.
3. The voter cannot check if the ballot was recorded or not. The authors describe two types of dispute: either the cast ballot was not recorded (D1); or an uncast ballot was recorded for the voter (D2).

In this regard, the protocol Themis by Bougon *et al.* (2022) considers a casting dispute where the voter can verify the consideration of its physical ballot as long as the server or a set of observers are trustworthy.

Although related to accountability, dispute resolution is a stronger requirement, since it requires unambiguity. In other terms, dispute resolution eliminates the need for a trusted authority acting as a judge. Any honest third party would act as such using the protocol execution and the data in the PBB.

However, different protocols consider different notions of accountability. In Basin *et al.* (2020), the definition of dispute resolution only includes partial accountability: for a protocol with multiple parties, it is enough to detect that a party misbehaved, but not which one. In the case of Themis, they improve the accountability guarantees by trying to identify which exact party misbehaved. However, for some cases Themis is only able to narrow down the blame to a set of parties, achieving partial accountability. Our proposed protocol DiReCT, is designed to provide individual accountability, as shown in the security analysis in Section 6.

4. Voting Model

DiReCT's voting model is structured around a secret sharing scheme and blind signatures (Sections 2.1 and 2.2), a design proposed and implemented in SUVS by Larriba and López (2022). In this voting model, the secret (ballot) is divided into shares and a commitment is created and blinded with a mask. The voting authority authenticates voters and signs the blinded commitments. The voters send the unblinded signed commitments to the tallying authorities. These authorities pool the shares during the tally to obtain the original votes.

4.1. *Entities and Their Roles*

The entities involved in the proposed voting protocol are: the *voters*; the *voting authority* (VA); the *k parties*; and, a *Public Bulletin Board* (PBB).

- The voters are the members of the census.
- The voting authority VA initially setups the protocol and is in charge of signing ballots of valid voters.
- Party p_j is a candidate in the election. It also plays the role of a tallying authority.
- A PBB is used to communicate public information. The PBB behaves as an append-only bulletin board (Heather and Lundin, 2009), providing a consistent view of the information posted, without the possibility of deleting any post. The PBB does not require any special assumptions compared to its use in other protocols (Doan *et al.*, 2025; Mosaheb *et al.*, 2025; Cuvelier *et al.*, 2013). For the sake of clarity, we assume

the existence of topics in the PBB, in the form of PBB:*topic*. Every entry in the PBB includes the entity and the timestamp:

PBB:topic := [entity, msg, timestamp]

4.2. *Security Assumptions*

Based on the previous definition of entities and their roles, we compile here the security assumptions of the protocol. We compiled them as a list to make it clear and cite on the protocol if needed:

SA1 Every voter v has a certificate and the corresponding signing key obtained from the census.

Sign_v denotes a signature issued by v with its certificate. The census is not an active entity in the protocol and, as other voting protocols (Adida *et al.*, 2009), its implementation is outside the scope of this paper.

SA2 The adversary $\mathcal{M}$, in addition to the (Dolev and Yao, 1983) capabilities, can corrupt voters, VA, and parties. A corrupted entity is modelled as a *covert adversary*, as defined by Aumann and Lindell (2010).

Covert adversaries are a type of semi-trusted entities that can misbehave when they do it surreptitiously. In other words, we assume that a covert adversary can perform an attack unless it can be attributed to them with public undeniable evidence (individual accountability). Covert adversaries represent a lower trust assumption compared with trusted, or *Honest but Curious* (HbC) authorities. HbC entities represent a privacy threat to voters although they are trusted to follow the protocol.

Aumann and Lindell (2010) formalized covert adversaries as a type of adversary which "faithfully models the adversarial behaviour in many commercial, political, and social settings". In terms of the adversary capability assumptions, covert adversaries are between an HbC, who follows the protocol and only targets the privacy, and a fully malicious adversary, with unlimited resources and no restraints.

SA3 There is a conflict of interest between the parties, such as being competing candidates during the election.

The concept of conflict of interest in e-voting was introduced by Moran and Naor (2010). We use the extended definition of Zou *et al.* (2017), where parties are not assumed to be trustworthy, but that they do not collude with each other. In practice, it is enough if just one party doesn't collude with the rest. In other words, the protocol works as expected as long as not all the parties in the election are colluding with each other.

SA4 During the casting phase, there is at least one party that does not block the casting and publishes the receipt.

By definition, covert adversaries are not obliged to respond or forward messages unless it leads to individual accountability. **SA4** is the only exception during the protocol. We

assume that, among all the candidates in the elections, there is at least one party interested in completing the process. Such party can still behave as a covert adversary in the rest of the protocol. This is still a lower assumption compared with other works, where authorities are trusted or always reply or forward messages even when they misbehave (Basin *et al.*, 2020).

SA5 A voter that start the process (Ballot Certification Request) will continue until her vote casting succeeds. This is an implicit requirement in other voting protocols (Clarkson *et al.*, 2008; Bougon *et al.*, 2022).

SA6 Each authority is responsible for the communication channel with the bulletin board. The authority is accountable if an expected message is missing from the PBB.

SA7 There is an anonymous channel between voters and parties which will be used by the voters to cast their ballot.

Its implementation is not part of the protocol design. We note that it can be implemented with anonymous networks such as Tor (2016) or STORK Consortium (2017). This is a lower assumption compared with mix-nets, which are needed to calculate the tally anonymously, meanwhile DiReCT only requires a channel that guarantees the anonymity of the transmission.

4.3. *Security Properties and Definitions*

This subsection describes a compact list of the desired properties of the protocol and common definitions.

Verifiability in e-voting has different approaches depending on the author. In this paper, we follow the strict definitions of verifiability from Moran and Naor (2010), Bougon *et al.* (2022), but divide it into *individual*, *universal* and *elegibility* verifiability, as defined by Kremer *et al.* (2010).

Individual verifiability is defined as the voters ability to check that their ballot is *recorded as cast* (2010) and that the ballot content did not change, known as *cast as intended* or CaI (2010).

The universal verifiability property ensures that anyone (voters and external auditors) can verify that the results of the elections align with the cast ballots (*counted-as-cast*) (Kremer *et al.*, 2010; Moran and Naor, 2010; Bougon *et al.*, 2022). Eligibility verifiability ensures that only ballots from registered voters are counted, with at most one ballot per voter (Kremer *et al.*, 2010; Bougon *et al.*, 2022).

Attacks performed by covert adversaries are also known as covert attacks. However, this term is also used to refer to attacks that cannot be detected. In our protocol, we use *surreptitious attacks* to distinguish attacks where the origin cannot be *proved*.

Vote suppression happens when the voting authority prevents some eligible voters from participating in the elections. *Denial of Casting* (DoC) is an equivalent attack where the voters ballot casting is blocked by an authority.

Timeliness, as defined by Basin *et al.* (2020), guarantees that a voter possess the evidence to resolve disputes no later than the election's end. We increase the granularity of

the timeliness definition by applying it to three specific steps in the protocol: T_{Cert} for the VA to respond to a ballot certification request; T_{Cast} for the parties to respond to a ballot casting; and, T_{Tally} for the parties to make public their shares during the tally. After performing the certification, cast, or after the tally starts, the election has set a specific window of time the process can last. Setting those timeouts allows the voter to detect the issue and resolve the dispute. Time constraints, such as the end of the voting period or how long does the voter need to wait for a response, are implicit parameters in the protocol specification. Still, in protocols such as Bougon *et al.* (2022), it is expected that the voter needs to wait for a response and can raise an issue otherwise. Note that the protocol does not rely on the synchronization of different entities to ensure its security guarantees. Timeouts serve only as upper limits on the expected processing time for various requests and can be set generously to accommodate potential timing differences. For instance, if the processing time for a request typically ranges from less than a minute to 5 minutes, setting a timeout of 15 minutes ensures that a missing response is clearly detectable while also accounting for synchronization differences of several minutes.

5. Protocol

This section describes DiReCT and the steps involved in the three phases of the elections: the preparation, the voting and the tally.

5.1. *Preparation Phase*

The preparation phase is divided into: the identity provision, where voters get credentials to participate in the election; and the system setup, where the election parameters are decided.

5.1.1. *Identity Provision*

The starting point of the identity provision in DiReCT is the existence of a honest census that can provide voters with a certificate (**SA1**).

The census owns the list of valid voters, a public PK and secret key and sends PK to PBB:prepare. For every voter v in the list of valid voters, the census sends a certificate $\textsc{Cert}_v$ signed with its secret key, and verifiable with the public key PK.

5.1.2. *System Setup*

The system setup involves the publication of keys and parameters needed for the secret sharing scheme, blind signatures and hashing. The relationship of parameters is displayed in Table 2.

The parameters associated with the secret sharing scheme (Section 2.1) are: the degree d of the polynomials; the prime p used in the modulo operations; and the maximum number of points the voter can generate l.

To setup the blind signatures (Section 2.2), the VA selects two prime numbers that are kept secret. The product of those primes, n, is published, and it is used in the modular

Table 2
Setup Parameters. Public parameters are shared in the PBB with the topic *prepare*. Private parameters are kept secret by the entities that create them.

Type	Public	Private
Census	Public key PK, used to validate certificates	Secret key, used to sign certificates
Secret sharing	Degree d, prime p and max number of points l	
Blind signature	Modulus n, public component va	Secret key s, and two prime factors of n
Hashing	*Hash* function, *fresh* value	
Waiting time	T_{Cert}, T_{Cast} and T_{Tally}	

arithmetic of the blind signature messages. The VA generates a secret key s and public component va such as $va \cdot s \equiv 1 \pmod{\phi(n)}$.

The hash function *Hash* and a freshness value used in the election *fresh* are to be decided before the election. The freshness value prevents reusing messages from other elections or precomputation attacks (Oechslin, 2003). It should be a value not used previously and unknown before the elections start. Both *fresh* and *Hash* are public.

The protocol makes use of the time variables T_{Cert}, T_{Cast} and T_{Tally}. They are timeouts for the voter to get the confirmation of the ballot certification (T_{Cert}), ballot casting (T_{Cast}), and the time the parties have to publish the shares at the beginning of the tally (T_{Tally}). As explained in Section 4.3, they are not needed in elections with honest authorities, but, in the presence of corrupt authorities, they help determine if a request has been blocked. The variables should be agreed between the VA and the parties, and be public before the election starts.

5.2. *Voting Phase*

In the voting phase voters participate in the election. It starts with the voter crafting the ballot, encoding their vote direction and producing the ballot shares. It follows the ballot certification, where the VA blindly signs the ballots and shares from valid voters in the census. The phase finishes with the casting, where the voters send their shares to the parties to be counted during the tally. Figure 1 depicts a simplified version of the process.

5.2.1. *Ballot Crafting*

The voter starts by encoding the vote direction as an integer C. The encoding is left as an implementation detail, but it can accommodate different types of voting. The encoded vote direction C is used as the independent term of a d-degree polynomial $q(x)$, as shown in Equation (1). According to the secret sharing scheme explained in Section 2.1, the voter can sample a set of points P from the polynomial (1), which can be used to form the ballot. Any single point from the set does not provide any information about the vote direction, only $d + 1$ points allow to interpolate $q(x)$ and obtain C. Each one of the k parties in the election receives a subset (SP) of P.[1]

[1]Each share can be composed by more than one point ($k \leqslant l$). Thus, the number of shares k does not necessarily correspond with the number of points l generated by the voter.

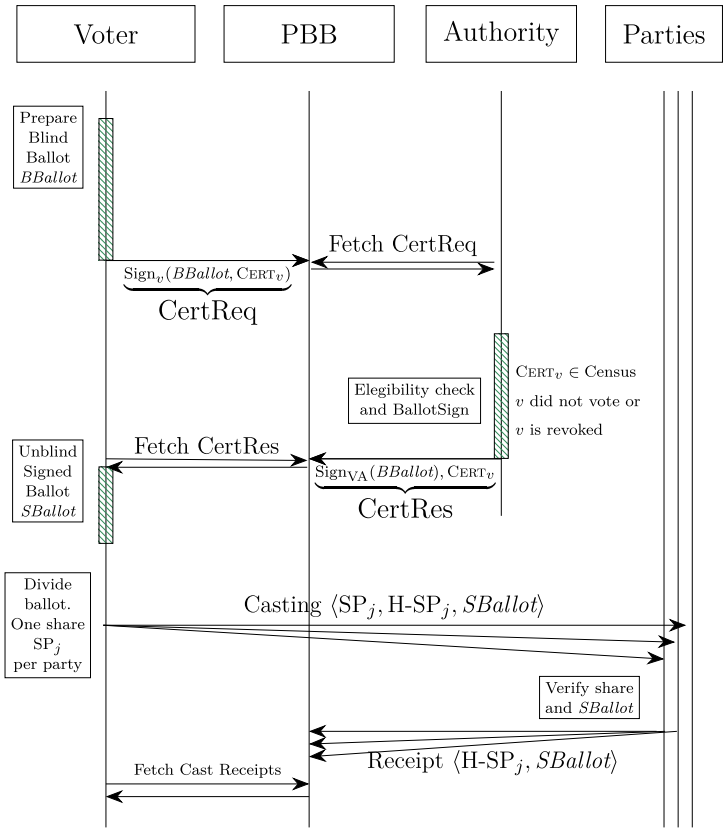

Fig. 1. Voting phase simplified diagram.

$$P = \bigcup_{i=1}^{k} \text{SP}_i \,. \tag{6}$$

Every time the shares are collected into P they are sorted to produce a consistent output. The sorting operation is omitted in the notation for the sake of clarity.

To craft the ballot commitment, the ballot is digested using the agreed hash function *Hash* and the freshness value *fresh* (Equation (7)).

$$\begin{aligned} \text{H-P} &:= \text{Hash}(P \,|\, \mathit{fresh}), \\ \text{H-SP}_i &:= \text{Hash}(\text{SP}_i \,|\, \text{H-P}). \end{aligned} \tag{7}$$

With the addition of the fresh value to the vote commitment, an adversary cannot reuse commitments from previous instances of the protocol. The commitment of the share H-SP$_i$ allows to link each share with the ballot, which will play a role in the casting verification.

5.2.2. *Ballot Certification*

During the ballot certification, the voter certifies the commitments with the VA. This guarantees that a voter can only vote once. The identity of the voter is used in this step. To prevent linking the identity with the vote direction, the certification, blind signatures are used as described in Section 2.2.

First, the voter blinds the commitments using the VA public component va (Equation (8)).

$$\begin{aligned} &\text{Blind(H-P)} := \text{Blind}\big(\text{H-P}, \mathit{mask}^{va}\big), \\ &\text{Blind(H-SP)}_i := \text{Blind}\big(\text{H-SP}_i, \mathit{mask}_i^{va}\big), \\ &\mathit{BBallot} := \big(\text{Blind(H-P)}, \big[\text{Blind(H-SP}_i)\big]_{i=1}^{k}\big). \end{aligned} \tag{8}$$

Each *mask* needs to be invertible modulo n to unblind the signature and different from each other. The voter uses the credentials obtained in the identity provision (Section 5.1.1) to sign *BBallot*. This step ensures that every voter can only obtain one ballot (unicity). Only voters with a valid certificate from the census can vote (democracy), and prevents the VA from creating extra ballots (ballot stuffing), since the VA does not possess a voter certificate.

Then, the voter v creates a secure communication with the PBB:certify to pose a request for certification (Equation (9)):

$$\text{CerRq}(v \rightarrow \text{PBB}) := v, \text{Sign}_v(\mathit{BBallot}, \text{Cert}_v), T_0. \tag{9}$$

The VA retrieves the requests from PBB:certify and processes them: verifies that the certificate Cert_v is valid and that the voter v has not requested a ballot before. If the checks are satisfied, the VA signs the concealed ballot $\text{Sign}_{\text{VA}}(\mathit{BBallot})$. The response is sent to PBB:certified:

$$\text{CertRes}(\text{VA} \rightarrow \mathit{PBB}) := \text{VA}, \langle \text{Sign}_{\text{VA}}(\mathit{BBallot}), \text{CerRq}_v \rangle, T_1. \tag{10}$$

The reason for publishing this information is three-fold: the voter can confirm that the ballot has been received as intended; it prevents the VA from covertly blocking a voter by not sending the response to the certification; and every party can check that only voters with valid credentials are certified by the VA.

The voter can easily retrieve the signed data from PBB:certified by unblinding it (Equation (11)), using the original masks. The process ends with the voter obtaining the signed ballot *SBallot* (Equation (12)). Hence, the certification of the commitments is performed without revealing the vote.

$$\text{Unblind}\big(\text{Sign}_{\text{VA}}(\mathit{BBallot})\big) = \mathit{SBallot}, \tag{11}$$

$$\mathit{SBallot} := \langle \text{Sign}_{\text{VA}}\,\text{(H-P)}\,, \big[\text{Sign}_{\text{VA}}(\text{H-SP}_i)\big]_{i=1}^{k} \rangle. \tag{12}$$

5.2.3. *Ballot Casting*

To cast her vote, the voter v sends to each party the signed ballot, a share SP and the share commitment at time T_2, as described in Equation (13).

$$\text{Casting}(v \rightarrow \text{party}_j) := \langle \text{SP}_j, \text{H-SP}_j, \mathit{SBallot} \rangle. \tag{13}$$

The party j receiving the share verifies both the signed commitment of the share and the signed ballot (Equation (14)).

$$\begin{aligned} \text{Verify}_{\text{VA}}(\text{Sign}_{\text{VA}}(\text{H-P})) &\stackrel{?}{=} \text{H-P}, \\ \left[\text{Verify}_{\text{VA}}\left(\text{Sign}_{\text{VA}}(\text{H-SP}_i)\right) \stackrel{?}{=} \text{H-SP}_i\right]_{i=1}^{k}&, \\ \text{Hash}(\text{SP}_j|\text{H-P}) &\stackrel{?}{=} \text{H-SP}_j. \end{aligned} \tag{14}$$

If the checks are successful, the party publishes a receipt in the PBB:cast. The receipt is the content of the cast message without the share. With the receipt the voter can verify that the party received the correct information.

$$\text{Casting}(\text{party}_j \rightarrow \text{PBB}) := \text{party}_j, \langle \text{H-SP}_j, \mathit{SBallot} \rangle, T_3. \tag{15}$$

During the tally, any entity can detect if a party sends a different SP_i than the one signed by checking the information in the PBB:cast.

The distribution of the shares between the candidates follows Shamir's secret sharing scheme (Section 2.1), where at least d points are needed to interpolate the d degree polynomial. These are the possible situations where the shares can be recollected:

- During the tally, when all the parties post the share and ballot in the PBB.
- If all parties collude and exchange their shares before the tally. This is not possible in our model, since the parties have conflicting interests (**SA3**).
- If the voter sends all the shares to a party. Although such vote can be detected and avoided, doing so colludes with the recovery mechanism against a denial of casting. We study the situation in Section 6.3.

5.3. *Tally Phase*

Once the voting finishes, the tally phase starts, with the parties sending the shares to the PBB:tally:

$$\text{Tally}(\text{party}_i \rightarrow \text{PBB}) := \text{party}_i, \langle \text{SP}_i, \text{H-SP}_i, \mathit{SBallot} \rangle, T_4. \tag{16}$$

To reconstruct the ballot, first the parties need to collect the shares with an identical *SBallot*. With all the shares, each party can sort them and reconstruct the original P, as seen in Equation (6). With P it is straightforward to obtain the original polynomial $q(x)$, for example using Lagrange's polynomials (see Equation (2)). The direction of vote can then be obtained from $q(x)$ by obtaining the independent term $C = q(0) \mod p$.

6. Security Analysis

The analysis of the protocol has been carried out according to the Universal Composability Framework (UC) (Canetti, 2001), which compares the real world behaviour of the protocol with an *ideal* one to prove the validity of certain statements. By simulation, the actions in the real world of an adversary are applied. The security is held if the ideal and real behaviour are indistinguishable, which implies that the adversary actions do not affect the functionality being analysed. When studying the security of a functionality, the initial state needs to be defined, which allows breaking down the protocol into components. Similarly, it is possible to use a *dummy party* that acts as a placeholder for an ideal functionality, modelling the case of parties that are not corrupted by the adversary. This case which combines the real world and the adversary with calls to an ideal functionality is known as an hybrid protocol.

Following the UC framework, this section defines the ideal functionalities that model honest entities and properties, where the covert adversary behaviour is included in the form of attacks. Theorems and proofs are included to specify how properties are achieved, how attacks are not possible in DiReCT, or how the covert adversary can be identified if it does the attack (individual accountability). The disputes are characterized as the combination of two attacks, one when the authority misbehaves, and another, where a voter makes a false claim against an honest authority. A subsection covers each of the phases in the protocol, grouping the ideal functionalities, attacks, disputes and proofs.

6.1. *Disputes in DiReCT*

Dispute resolution (Section 3.1) is a key contribution of this paper. Introduced by Basin *et al.* (2020), in their work the authors identify two disputes regarding individual verifiability: claims for unrecorded ballots (D1); and claims for uncast recorded ballots (D2).

We adapt these definitions to DiReCT without affecting their meaning. D1 is included as DTally, although in DiReCT the recorded ballots can be universally verified (recorded as cast). D2 does not apply as is to DiReCT, instead we consider a more general attack where an authority falsifies ballots (ballot stuffing), and prove that the dispute is not possible in DiReCT. Furthermore, this paper introduces two new disputes. The first one, DCert, is a dispute related to vote suppression, where the voter is blocked from the authentication and ballot certification. The second one, DCast, is a dispute related to denial of casting (DoC), where the voter is blocked from verifying the ballot casting.

6.2. *Ballot Certification*

The behaviour of a trusted VA during the ballot certification is modelled by the ideal functionality $\mathcal{F}_{Cert}$. $\mathcal{F}_{CertObs}$ ensures that any voter with valid credentials obtains a signed ballot within a given time T_{Cert}. It uses the public bulletin board PBB to log public message exchanges. The adversary can read the information but cannot block access to it, and only can post by corrupting an entity with access. The PBB provides integrity and non-repudiation: the content posted cannot be altered by the adversary and it's origin is unambiguous. Posted messages include the entity, the content and a timestamp (Section 4.1).

Ideal $\mathcal{F}_{Cert}$. Given the CerRq from a voter v:

$$\text{CerRq}(v \to \log) := v, \langle \text{Sign}_v(\mathit{BBallot}), \text{CERT}_v \rangle, T_0.$$

(a) $\mathcal{F}_{Cert}$ retrieves the request from the log. Aborts if the request does not follow the format or the signature is not from a voter in the census:

$$\text{CERT}_v \stackrel{?}{\in} \text{Census}.$$

(b) $\mathcal{F}_{Cert}$ aborts if the voter has been issued a signed ballot already according to the public log.

$$\text{Count}\big(\text{CERT}_v \in \text{CertRes}^*, \log\big) \stackrel{?}{=} 0.$$

(c) $\mathcal{F}_{Cert}$ signs the masked ballot and publishes it in the public log. The response is published in less than T_{Cert} time since the request was posted. Output CertRes.

$$\text{CertRes}(\mathcal{F}_{Cert} \to \log) := \mathcal{F}_{Cert}, \langle \text{Sign}_{\text{VA}}(\mathit{BBallot}), \text{CerRq}_v \rangle, T_1.$$

The ideal functionality or honest VA only knows that *BBallot* is a list of $k+1$ bit strings, since it has been blinded. In the case of an honest voter, *BBallot* corresponds to the ballot and shares commitments (Equation (8)).

We define a second ideal functionality $\mathcal{F}_{CertObs}$, which models the universal verification of the Ballot Certification within a time frame T_{Cert} (by any external auditor, party or voter). $\mathcal{F}_{CertObs}$ outputs a 1 if the certification is performed on time, 0 otherwise.

Ideal $\mathcal{F}_{CertObs}$. Given the CerRq from a voter v:

$$v, \langle \text{Sign}_v(\mathit{BBallot}), \text{CERT}_v \rangle, T_0.$$

(a) Perform steps (a) and (b). If the checks are not successful, abort accordingly.
(b) Check in the public log if the corresponding CertRes is published:

$$\forall\ \text{VA}, \langle \text{Sign}_{\text{VA}}(X), \text{CerRq}_v \rangle, T_1 \to \text{Verify}\big(\text{Sign}_{\text{VA}}(X)\big) \stackrel{?}{=} \mathit{BBallot}.$$

(c) If the previous step succeeds, calculate if the receipt has been published on time. Output 1 if $T_1 - T_0 \leqslant T_{\text{Cert}}$, 0 otherwise.
(d) If CertRes is not published at $T_0 + T_{\text{Cert}}$, output 0.

Vote Suppression Dispute (DCert)

The DCert vote suppression dispute is linked with two attacks: a malicious VA blocking honest voters (*vote suppression*); and a malicious voter claiming a vote suppression from an honest VA (*false suppression claim*).

In the vote suppression attack, the real world adversary $\mathcal{M}$ corrupts the VA. Since voters reveal their identity to the VA, the adversary can selectively block honest voters from ballot certification.

Hybrid protocol – Vote suppression attack. The hybrid protocol consists of an honest voter v, a simulator S of a corrupt VA.

1. v submits a valid request:

$$\text{CerRq}(v \to \text{log}) := v, \langle \text{Sign}_v(BBallot), \text{CERT}_v \rangle, T_0$$

2. S can perform the vote suppression through different actions:
   - i Omitting the answer to the request: S does not perform any action.
   - ii Signing a different content than the one in CerRq (perhaps to perform ballot stuffing) and publishing to the public log:

$$\text{VA}, \langle \text{Sign}_{\text{VA}}(X), \text{CerRq}_v \rangle, T_* \longrightarrow X \neq BBallot$$

   - iii Signing with a different key and then publishing it:

$$\text{VA}, \langle \text{Sign}_{\text{VA}'}(BBallot), \text{CerRq}_v \rangle, T_* \longrightarrow \text{VA}' \neq VA$$

3. As a result, the voter does not obtain $\text{Sign}_{\text{VA}}(BBallot)$ and cannot continue voting.

To accomplish the dispute resolution in case of vote suppression, any third party should be able to detect the attack. Framing it in terms of a covert adversary, it is not possible for the adversary to perform the vote suppression surreptitiously. This is analogous to prove that in the hybrid protocol where the vote suppression is performed, the adversary cannot modify the behaviour of the ideal functionality $\mathcal{F}_{CertObs}$, which always outputs a 0.

Theorem 1. *Vote suppression in DiReCT cannot be executed surreptitiously.*

Proof. A valid certification request is suppressed if $\mathcal{F}_{CertObs}$ outputs 0. Note that $\mathcal{F}_{CertObs}$ only aborts at step (a). However, since CerRq is valid, $\mathcal{F}_{CertObs}$ does not abort, no matter what action S performs.

To succeed the covert attack, the adversary needs to perform the vote suppression with an output from $\mathcal{F}_{CertObs}$ of 1. The options for S are:

- If S omits the answer (i), after some time T_{Cert}, $\mathcal{F}_{CertObs}$ outputs a 0 (step (d)).
- If S signs a different content (ii), in step (b) $\mathcal{F}_{CertObs}$ detects the mismatch of the signed content $\text{Verify}_{\text{VA}}(\text{Sign}_{\text{VA}}(X)) \neq BBallot$. After T_{Cert}, $\mathcal{F}_{CertObs}$ outputs a 0 (step (d)).
- If S sends a different signature (iii), similarly to the last case, $\mathcal{F}_{CertObs}$ detects the mismatch $\text{Verify}_{\text{VA}}(\text{Sign}_{\text{VA}'}(BBallot)) \neq BBallot$ and outputs a 0.

Consequently, the adversary cannot avoid $\mathcal{F}_{CertObs}$ outputting 0 at time $T = T_0 + T_{\text{Cert}}$. □

The ideal functionality $\mathcal{F}_{CertObs}$ can be translated to the real world behaviour by any third party with access to the PBB. This is what allows us to translate the hybrid protocol proof from Theorem 1 to the real world.

A voter corrupted by $\mathcal{M}$ can pose a false suppression claim at an honest VA.

Hybrid protocol – false suppression claim attack. The hybrid protocol consists of a simulator S acting as a corrupt voter, a dummy authority VA implementing $\mathcal{F}_{Cert}$, and the ideal functionality $\mathcal{F}_{CertObs}$. To proceed with the false suppression claim, the simulator proceeds as follows:

i S does not send the request CerRq.
ii S sends a malformed request.
iii S sends a request signed with a certificate not part of the census.
iv S sends a second request after having completed the ballot certification previously.

Any of these actions results on VA executing $\mathcal{F}_{Cert}$ and aborting without publishing CertRes.

In order $\mathcal{F}_{CertObs}$ to output 0, as universally verifiable evidence of the suppression of the ballot certification request. Theorem 2 proves how this false claim is not possible.

Theorem 2. *A voter cannot sustain false claims of vote suppression against the VA.*

Proof. A false suppression claim succeeds if $\mathcal{F}_{CertObs}$ outputs 0. Lets review each of S actions in the false suppression claim hybrid protocol and the output of $\mathcal{F}_{CertObs}$:

- Step i: CertReq is the input of $\mathcal{F}_{CertObs}$. Without it, $\mathcal{F}_{CertObs}$ cannot output 0.
- Steps ii and iii: Both cases are checked in step (a), which results in abort.
- Steps iv: $\mathcal{F}_{CertObs}$ verifies if the voter has received a previous CertRes, in which case it aborts (step (a)).

Thus the adversary cannot change the behaviour in the $\mathcal{F}_{CertObs}$ with its actions. □

Theorems 1 and 2 results permit successful dispute resolutions for DCert.

6.3. *Ballot Cast*

The ideal functionality $\mathcal{F}_{Cast}$ models an honest party in the casting phase. Validates a ballot share of a voter and posts a receipt in a public log within time T_{Cast}.

Ideal $\mathcal{F}_{Cast}$. Given the Casting message, sent privately from voter v to $\mathcal{F}_{Cast}$, at time T_2:

$$\text{Casting}(v \to \mathcal{F}_{Cast}) := \left\langle \text{SP}_j, \text{H-SP}_j, \underbrace{\text{Sign}_{\text{VA}}(\text{H-P}), \left[\text{Sign}_{\text{VA}}(\text{H-SP}_i)\right]_{i=1}^{k}}_{SBallot} \right\rangle$$

1. The ideal functionality verifies the message, following the protocol description Equation (14):
   a) The shares and ballot commitments have been signed by the VA, otherwise abort.

$$\text{Verify}_{\text{VA}}(\text{Sign}_{\text{VA}}(\text{H-P})) \stackrel{?}{=} \text{H-P},$$
$$\left[\text{Verify}_{\text{VA}}\left(\text{Sign}_{\text{VA}}(\text{H-SP}_i)\right) \stackrel{?}{=} \text{H-SP}_i\right]_{i=1}^{k}.$$

   b) The share commitment is included, otherwise abort.

$$\text{Hash}(\text{SP}_j|\text{H-P}) \stackrel{?}{=} \text{H-SP}_j,$$
$$\text{Sign}_{\text{VA}}(\text{H-SP}_j) \in \mathit{SBallot}.$$

2. After performing the checks, $\mathcal{F}_{Cast}$ posts to the public log a receipt. The receipt is the casting message without the secret share. Since the cast message was sent, the ideal functionality sends the receipt in less than T_{Cast} and outputs 1.

$$\mathcal{F}_{Cast}, \langle \text{H-SP}_j, \mathit{SBallot}\rangle, T_3 \longrightarrow T_3 - T_2 \leqslant T_{\text{Cast}}.$$

It is assumed that T_{Cast} is chosen based on the infrastructure and election's size. Also, in the worst case scenario, the posting will happen right before the tally starts.

We define a second ideal functionality $\mathcal{F}_{CastObs}$, which models how an honest voter can verify that the casting has been performed within a time frame T_{Cast}. $\mathcal{F}_{CastObs}$ outputs 1 if the casting is performed on time, 0 otherwise.

Ideal $\mathcal{F}_{CastObs}$. Given the Casting message of share SP_j, from a voter v to a party j:

$$\text{Casting}(v \rightarrow j) := v, \langle \text{SP}_j, \text{H-SP}_j, \mathit{SBallot}\rangle, T_2,$$
$$\mathit{SBallot} := \text{Sign}_{\text{VA}}(\text{H-P}), \left[\text{Sign}_{\text{VA}}(\text{H-SP}_i)\right]_{i=1}^{k}.$$

1. Verify the casting as in $\mathcal{F}_{Cert}$ Step 1, aborting if the checks fail.
2. Look in the public log for the corresponding cast receipt:

$$j, \langle \text{H-SP}_j, X\rangle, T_3 \longrightarrow X \stackrel{?}{=} \mathit{SBallot}.$$

3. If the previous check is successful and $_3 - T_2 \leqslant T_{\text{Cast}}$, output 1.
4. Otherwise, if the receipt is not published at $T_2 + T_{\text{Cast}}$ time, output 0.

Cast verification dispute (DCast). The second dispute we consider affects the voter's verification of her ballot casting (DCast). Given that the communication between the voter and the parties is not public, the adversary may attempt an attack without being identified (covert adversary). Two attacks can cause the dispute: a malicious party blocking some

honest voters (*Denial of Cast or DoC*); and a malicious voter claiming being denied by an honest party (*false DoC claim*).

Hybrid protocol – Denial of Casting attack (DoC). The protocol considers a dummy voter v, the ideal functionality $\mathcal{F}_{CastObs}$, and S (a corrupt party).

1. At T_2, v submits to S and $\mathcal{F}_{CastObs}$ a valid casting message:

$$\text{Casting}\big(v \to (S, \mathcal{F}_{CastObs})\big) := \langle \text{SP}_S, \text{H-SP}_S, \mathit{SBallot} \rangle.$$

2. S can perform the denial of casting through:
   (a) S does not publish the receipt of the casting.
   (b) S publishes a malformed receipt:

$$\mathit{party}_j, \langle \text{H-SP}'_S, \mathit{SBallot} \rangle, T_3 \longrightarrow \text{H-SP}'_S \neq \text{H-SP}_S.$$

3. As a result, the voter cannot confirm if the casting has been successful.

The DoC hybrid protocol summarizes how the adversary can avoid confirming the casting to the user. Theorem 3 proves that DoC is not possible in DiReCT.

Theorem 3. *A corrupt party cannot block the ballot casting of a voter without the voter noticing it.*

Proof. For any valid voter's ballot casting $\mathcal{F}_{CastObs}$ outputs 0. First, lets analyse if $\mathcal{F}_{CastObs}$ could abort. The ideal functionality only aborts in step 1. if the casting message is invalid, but the premise of the attack is that the adversary is blocking a valid message, thus $\mathcal{F}_{CastObs}$ does not abort.

The simulator can then perform two actions. If S does not publish the receipt, $\mathcal{F}_{CastObs}$ will not succeed finding it and will output a T_{Cast} waiting (step 4.). Similarly, if S publishes a receipt with the wrong share commitment, $\mathcal{F}_{CastObs}$ will ignore it and output 0.

The simulator cannot perform the denial of casting without the voter noticing after a time T_{Cast}, since it cannot modify the behaviour of $\mathcal{F}_{CastObs}$. □

We prove in Theorem 4 that every user can verify her casting before the tally.

Theorem 4. *In DiReCT, an honest voter can verify the casting of its ballot before the tally.*

Proof. In the protocol, T_{Cast} is the reasonable maximum time to receive the casting confirmation. If the voter cast her ballot to a party j within time, according to $\mathcal{F}_{CastObs}$, either j posts the receipt of the casting or the voter can detect the verification blocking waiting T_{Cast}. In both cases, the voter can verify if the casting is correct or not before the tally. □

The recovery process $\mathcal{F}_{Rec}$ is defined based on the result of Theorem 4, and proved in Theorem 5.

Ideal $\mathcal{F}_{Rec}$ – Casting with DoC recovery. The ideal functionality models an honest voter during the casting phase. Given a valid casting message of share j $\langle \mathrm{SP}_j, SBallot \rangle$ and a party:

(1) At T_2, submit the message to the party.
(2) Call $\mathcal{F}_{CastObs}$ with input the cast message.
(3) If $\mathcal{F}_{CastObs}$ outputs 1, the casting is successful and verified. Output 1.
(4) If $\mathcal{F}_{CastObs}$ outputs 0, and there are other casting authorities, pick a different $party'$ and repeat step (1).
(5) If $\mathcal{F}_{CastObs}$ outputs 0 and there are no other parties to perform step (1).

Theorem 5. ***Recovery:*** *Any voter can detect and recover from a denied ballot casting (DoC).*

Proof. Theorem 4 proves that the voter can verify the ballot casting before the tally phase. Furthermore, $\mathcal{F}_{Rec}$ models an honest voter during the casting phase. We define an hybrid model where $\mathcal{F}_{Rec}$ interacts with parties corrupted by the adversary. According to $\mathcal{F}_{Rec}$, every time the voter tries to cast with a corrupted party, $\mathcal{F}_{CastObs}$ outputs a 0 and $\mathcal{F}_{Rec}$ tries with a different authority. In a proof by contradiction, let us assume that the adversary has prevented the voter from performing the recovery. This means that the adversary has been able to modify the behaviour of the ideal functionality $\mathcal{F}_{Rec}$ to output a 0. By its definition, this only happens if all the casting authorities are corrupted and performing the DoC. However, this contradicts assumption **SA4** where at least one party participates in the casting. □

Recovery depends on at least one party participating in the casting **SA4**. This is included as a security assumption for the sake of completeness. However, it is reasonable to assume that among all the candidates of the elections, there is at least a party that is willing to complete the process. Note that the party can still behave as a covert adversary. In addition, DoC is performed without knowing the identity of the voter or the content of the ballot. Thus there is little incentive for all the parties to perform the DoC other than to disrupt the elections.

Note that the recovery mechanism, where voters may send their shares again to a different party, does not imply any linkability risk. The transmission is private (**SA7**), and the content received by the party cannot be linked to the voter, independently of the amount of shares a party receives.

The dispute resolution of DCast relies on honest voters performing the recovery. In other words, in case of blocking, the voter does not have evidence to prove the block to a third party. This is the reason why DiReCT dismisses all claims in the denial of casting. The dispute resolution is achieved by combining this dismissal with the fact that voters can always detect and recover from the blocking (Theorem 5).

6.4. *Tally*

We define two ideal functionalities related to the tally. $\mathcal{F}_{TallyObs}$ represents the record-as-cast universal verifiability. Given a ballot that has been correctly cast with the corresponding public receipt ($\mathcal{F}_{CastObs}(\text{Casting}_j)$ with output 1), any entity with only access to the log can verify the tally. The function aborts if not enough receipts were issued to reconstruct the vote. It outputs (0, NotSharing, [party$_x$]) with the parties that did not publish a share that had a cast receipt, or (0, ShareNotFromReceipt, [party$_x$]) if the share was not correct. (0, ShareNotFromP) if the shares pass all the checks but they do not account for the ballot, and (1, $SBallot_j$, $vote_j$) if the tally is successful and it is possible to retrieve the vote.

Ideal $\mathcal{F}_{TallyObs}$. This functionality takes as an input a casting receipt from the public log:

$$\text{Receipt}_j := \mathcal{F}_{Cast}, \langle \text{H-SP}_j, SBallot_j \rangle, \_ \tag{17}$$

1. Look in the public log for receipts of the remaining shares.

$$\text{Receipt}_m := \text{party}_*, \langle \text{H-SP}_m, SBallot_m \rangle, \_$$
$$\longrightarrow SBallot_j \stackrel{?}{=} SBallot_m$$

   a) If the amount of receipts l is not enough to build the vote ($l < k$), abort.
2. Gather the l casting messages corresponding to the receipts from the public log. If all the valid casting messages are not present in the log at T_{Tally} since the star of the tally, output (0, NotSharing, [party$_x$]) for all the parties that did not share the message. Otherwise, for each casting message:
   a) Check that the share matches the receipt published during the casting, otherwise output (0, ShareNotFromReceipt, [party$_x$]) with the parties that sent a different share.

$$\text{Hash}(\text{SP}_m|\text{H-P}) \stackrel{?}{=} \text{H-SP}_m$$

3. Collect the shares into P and verify that it matches the ballot, if not, output (0, ShareNotFromP).

$$\text{H-P} \stackrel{?}{=} \text{Hash}(P \mid fresh)$$

4. If it matches, interpolate the vote, publish in the public log the vote and ballot, and output (1, $SBallot_j$, $vote_j$).

$\mathcal{F}_{Tally}$ models the behaviour of an honest party during the tally. Only shares with the corresponding receipts ($\mathcal{F}_{Tally}$ output 1) are considered. It considers $\mathcal{F}_{TallyObs}$ and outputs 1 when the tally is correct, 0 if there is an intentional problem, or abort if not enough shares have been cast.

Ideal $\mathcal{F}_{Tally}$. This ideal functionality takes as an input a Casting_j message and the corresponding receipt (output 1 with $\mathcal{F}_{Cast}$):

$$\text{Casting}_j := \langle \text{SP}_j, \text{H-SP}_j, SBallot \rangle,$$
$$\text{Receipt}_j := \mathcal{F}_{Cast}, \langle \text{H-SP}_j, SBallot_j \rangle, T_3.$$

1. Look in the public log for receipts of the remaining shares.

$$\text{Receipt}_m := \text{party}_*, \langle \text{H-SP}_m, SBallot_m \rangle, \_ \longrightarrow SBallot_j \stackrel{?}{=} SBallot_m$$

   a) If the amount of receipts l is not enough to build the vote ($l < k$), abort.

   b) Otherwise, publish the casting message in the public log.

1. Call $\mathcal{F}_{TallyObs}$ with Receipt_j and output accordingly.

Recorded as Cast (DTally)

The DTally dispute arises when a ballot is not recorded. Theorem 5 proved that the user can recover from cast-blocking. This subsection is built on top of that result, leaving to the adversary the only option to block the recording of the ballot during the tally. The dispute resolution is composed by two attacks: the interception of the ballot by the adversary (*blind and selective tally interception*); and a corrupt voter false claim (*false tally interception claim*).

In the random tally interception the adversary's goal is to *randomly* block some ballots from being counted with plausible deniability.

Hybrid protocol – Blind Tally interception attack. The protocol simulates a corrupt party_j. This attack takes place during the tally phase. The input is a successfully cast message Casting_j with output 1.

S has two possible actions to intercept the vote:

- S does not send the share.
- Instead of sending the received SP_j, S sends a different one $\text{SP}_j^{\mathfrak{M}}$. The reconstruction of the shares would lead to an incorrect vote P':

$$P = \left\{ \bigcup_{i=1}^{k} \text{SP}_i \right\}, \qquad P' = \left\{ \bigcup_{i=1, i \neq j}^{k} \text{SP}_i \right\} \cup \text{SP}_j^{\mathfrak{M}}. \tag{18}$$

In the selective tally interception, the adversary's goal is to learn the vote direction of a share in order to block it, instead of blindly blocking some ballots from the tally.

Hybrid protocol – Selective Tally interception attack. The hybrid protocol consists of a simulator S of the corrupt party$_j$. This attack takes place during the tally phase. The input is a successfully cast message Casting$_j$:= $\langle \text{SP}_j, SBallot_j \rangle$

1. S waits until the rest of the shares corresponding to $SBallot_j$ are published by the rest of the parties:

$$\left[\text{party}_i, \langle \text{SP}_i, SBallot_i \rangle, T_i\right]_{i=1}^{k-1}.$$

2. S reconstructs P and obtains the vote direction $vote_j$.
3. If S decides to block the share based on the vote direction, follow the Blind Tally interception.

We prove that DiReCT is robust against blind (Theorem 6) and selective (Theorem 7) tally interception.

Theorem 6. *DiReCT ensures individual accountability against blind tally interception.*

Proof. The adversary $\mathcal{M}$ can perform the blind tally interception through two actions:

- $\mathcal{M}$ does not send its share: any party or external entity can execute $\mathcal{F}_{TallyObs}$, which only requires access to the public log and the receipt from the casting. The receipt is in the public log as defined in the attack ($\mathcal{F}_{CastObs}(\text{Casting}_j) = 1$). The result of $\mathcal{F}_{TallyObs}$ is (0, NotSharing, [party$_x$]), with $\mathcal{M} \in$ [party$_x$].
- $\mathcal{M}$ sends a different share: similarly, any entity can call $\mathcal{F}_{TallyObs}$, which outputs (0, ShareNotFromReceipt, [party$_x$]).

In both cases, $\mathcal{F}_{TallyObs}$ can detect the attack and which party performed the interception (individual accountability). □

Theorem 7. *DiReCT ensures individual accountability against selective tally interception.*

Proof. The selective tally interception is an extended version of the blind tally interception. Due to UC, if DiReCT offers individual accountability for blind tally interception, it also offers it to any attack that uses it. □

By combining the previous results, Corollary 8 demonstrates the record-as-cast property in the protocol.

Corollary 8. *DiReCT possess the record-as-cast property.*

Proof. Record-as-cast states that any ballot correctly cast ends up in the final tally. The definition of $\mathcal{F}_{Tally}$ summarizes the process, if a ballot has enough receipts, either the parties exchange the shares and perform the tally or the party not collaborating is identified

(Theorems 6 and 7). Under a covert adversaries threat model (**SA2**), all cast ballots are then recorded successfully. No other attack is possible since all that is needed for the final tally is access to the PBB. □

The following ideal functionality $\mathcal{F}_{Vote}$ models an honest voter performing all the steps to vote in DiReCT. It interacts with the public log, k casting authorities $[A_i]_{i=1}^{k}$, $\mathcal{F}_{CertObs}$ and $\mathcal{F}_{Rec}$. The definition $\mathcal{F}_{Vote}$ attacks on the results of the elections.is needed to define in simpler terms

Ideal functionality – $\mathcal{F}_{Vote}$.

The ideal functionality takes as an input a list of shares $[\mathrm{SP}_i]_{i=1}^{k}$, a ballot commitment *Ballot* and a certificate part of the census CERT:

1. Blind the ballot and publish a certification request (CerRq):

$$BBallot := \mathrm{Blind}(Ballot),$$
$$\mathrm{CerRq}(\mathcal{F}_{Vote} \rightarrow \log) := \mathcal{F}_{Vote}, \langle \mathrm{Sign}_{\mathcal{F}_{Vote}}(BBallot), \mathrm{CERT} \rangle, T_0.$$

2. Call $\mathcal{F}_{CertObs}$ with the above input. Abort if $\mathcal{F}_{CertObs}$ aborts, output 0 if $\mathcal{F}_{CastObs}$ outputs 0.
3. If $\mathcal{F}_{CertObs}$ outputs 1, retrieve CertRes from the public log and unblind the signed ballot:

$$\mathrm{CertRes} := \mathrm{VA}, \langle \mathrm{Sign}_{\mathrm{VA}}(BBallot), \mathrm{CerRq}_v \rangle, T_1,$$
$$SBallot := \mathrm{Unblind}\big(\mathrm{Sign}_{\mathrm{VA}}(BBallot)\big).$$

4. Call $\mathcal{F}_{Rec}$ with a share and the casting authority. Output 0 if $\mathcal{F}_{Rec}$ does it.
5. If $\mathcal{F}_{Rec}$ outputs 1, the voting is successful, output 1 as well.

To perform a false claim, the adversary still needs to cast a message that will pass $\mathcal{F}_{CastObs}$. For example, if the adversary tries to send a fake share that does not match the commitment, the parties will not publish the receipt and the share will be discarded from the tally.

Hybrid protocol – False Tally interception claim attack. The hybrid protocol consists of a simulator S of v. The protocol includes calls to $\mathcal{F}_{Cert}$ and $\mathcal{F}_{Cast}$.

1. S first selects a vote direction, encodes it as a set of points P and creates the corresponding k shares SP_i.
2. S exchanges one of the shares SP_j by a fake one $\mathrm{SP}_j^{\mathcal{M}}$ and calculates a commitment with the fake share and P commitment H-P: $\mathrm{H\text{-}SP}_j^{\mathcal{M}}$.

$$Ballot^{\mathcal{M}} := \mathrm{H\text{-}P}, \mathrm{H\text{-}SP}_j^{\mathcal{M}} \cup [\mathrm{H\text{-}SP}_i]_{i=1 \neq j}^{k}.$$

3. The new share leads to a different set of points P', in a way that does not allow reconstructing the vote:

$$P = \left\{ \bigcup_{i=1}^{k} \mathrm{SP}_i \right\}, \qquad P' = \left\{ \bigcup_{i=1, i\neq j}^{k} \mathrm{SP}_i \right\} \cup \mathrm{SP}_j^{\mathcal{M}}, \tag{19}$$

$$\text{H-P} \neq \mathrm{Hash}\big(P' \mid \mathit{fresh}\big).$$

4. S then performs the voting, calling:

$$\mathcal{F}_{\mathit{Vote}}\big(\mathrm{SP}_j^{\mathcal{M}} \cup [\mathrm{SP}_i]_{i=1\neq j}^{k}, \mathit{Ballot}^{\mathcal{M}}, \textsc{Cert}_v\big)$$

5. $\mathcal{F}_{\mathit{Vote}}$ will output 1, but the ballot cannot be reconstructed in the tally. Then S can try to make a false claim against party$_j$.

To complete the resolution of the DTally dispute, Theorem 9 demonstrates that the false interception claim cannot be performed:

Theorem 9. *A party$_j$ being subject to a false tally interception claim has universally verifiable evidence of its honesty.*

Proof. In the attack, the adversary calls $\mathcal{F}_{\mathit{Vote}}$ with output 1. Thus, party$_j$ can call $\mathcal{F}_{\mathit{Tally}}$ with the adversary fake material. The output will be (0, ShareNotFromP), which means that all the material submitted matches the signed commitments, but the shares do not match P. After all parties have shared their votes, any third party can perform the same steps as $\mathcal{F}_{\mathit{Tally}}$ to arrive to the same conclusion, which is universal verifiable evidence of its honesty. □

6.5. *Election Security*

$\mathcal{F}_{\mathit{Eleg}}$ models the universal verification of the eligibility in the elections, which can be performed after the tally by any external auditor, party or voter. $\mathcal{F}_{\mathit{Eleg}}$ outputs 1 when only members of the census voted, 0 otherwise.

Ideal functionality – $\mathcal{F}_{\mathit{Eleg}}$. Given access to the log after the tally (all casting authorities performed $\mathcal{F}_{\mathit{Tally}}$):

1. Retrieve all the CertRes from the public log:

$$\mathrm{CertRes}_x := \mathrm{VA}, \big\langle \mathrm{Sign}_{\mathrm{VA}}(\mathit{BBallot}), \mathrm{CerRq}_x \big\rangle, T_{1,x}$$

2. For each $\mathrm{CertRes}_x$, take CerRq_x and verify it using $\mathcal{F}_{\mathit{CertObs}}$. This verifies that the request is from a voter in the census that has not voted before.

3. If the output of $\mathcal{F}_{CertObs}$ is $(1, \text{CertRes}_y)$, ensure that they are the same:

$$\text{CertRes}_x \stackrel{?}{=} \text{CertRes}_y$$

4. Output 0 if the check fails or if $\mathcal{F}_{CertObs}$ outputs 0. Otherwise, continue and retrieve all the different votes in the public log from the tally:

$$\text{party}_x, \langle SBallot_y, \text{vote}_y \rangle$$

5. If the amount of CertRes equals the amount of unique votes, output 1.
6. Otherwise output 0.

The ballot stuffing hybrid protocol describes how an adversary may attempt to generate ballots to alter the elections, and Theorem 10 proves how the covert adversary is identified in DiReCT.

Hybrid protocol – Ballot Stuffing attack. The hybrid protocol consists of a simulator S, acting as a corrupt VA, and some dummy parties calling $\mathcal{F}_{Cast}$ and $\mathcal{F}_{Tally}$.

1. S obtains the shares and ballot from the desired vote direction.
2. S signs the ballot with VA key and casts the ballot using $\mathcal{F}_{Rec}$.

Theorem 10. *An adversary cannot perform ballot stuffing surreptitiously. If the adversary performs the attack, it is detected and attributed correctly.*

Proof. $\mathcal{F}_{Eleg}$ detects a mismatch between the number of ballot certification responses and the number of votes. To perform the attack without being detected, the adversary needs to modify the behaviour of $\mathcal{F}_{Eleg}$ and obtain an output of 1.

$\mathcal{F}_{Eleg}$ compares the number of CertRes with votes in the tally. By the assumption **SA5**, all the voters that start the certification complete the cast process. In addition, Corollary 8 proved that all cast ballots are recorded (record-as-cast), thus $\mathcal{F}_{Eleg}$ outputs 1 under normal conditions. The adversary cannot corrupt the census **SA2** or obtain extra certificates from the census. The adversary cannot fake CertRes.

If the adversary performs ballot stuffing , then there will be more ballots counted than ballot certified: $[\text{CertRes}_i]_{i=1}^{x}, [\text{vote}_j]_{j=1}^{y} \longrightarrow x \neq y$.

The adversary can only perform the ballot stuffing unnoticed if it blocks ballots from honest voters. However, as it has been proved, the adversary cannot block ballots during the casting (Theorem 5), during the tally (Corollary 8), neither can manipulate CertRes (Theorem 3).

Since the only entity able to sign the ballot is the VA, any mismatch on the tally can be individually accounted to a corrupt VA. □

Corollary 11. *Eligibility verifiability is guaranteed in DiReCT.*

Proof. $\mathcal{F}_{Eleg}$ explains how to universally verify that only voters in the census voted, that they only voted once, and that the amount of ballot certification responses matches the amount of votes. Theorem 10 proves that ballot stuffing cannot be performed surreptitiously. With the assumption that $\mathcal{M}$ is a covert adversary (**SA2**) the attack cannot be performed. This implies that eligibility verifiability is achieved in DiReCT. □

7. Complexity and Scalability Analysis

We analyse the scalability of the protocol in two parts. First, we conduct a complexity analysis, measuring the number of messages sent by each voter as well as the total number of messages exchanged, and compare these figures with other voting protocols. Second, we evaluate the storage requirements in the PBB, using some example values for reference.

The voter performs carries out the following steps: she sends a certification request to the PBB (Equation (9)); she fetches the certification result from the PBB (Equation (10)); she casts (distributes) the shares to k tally authorities (Equation (13)); and retrieves the casting receipts from the PBB (Equation (15)). These steps result in a total of $3 + k$ messages per voter.

The overall number of messages in the election consists of the messages exchanged by voters and the voting authority during ballot casting, as well as those exchanged among the tally authorities. During ballot casting, the system must handle $v(k + 8)$ messages, where v is the number of voters. During the tally, $3k$ additional messages are required: first to send the collected shares; and, then, to post the final count results. In total, the election involves $v(k + 8) + 3k$ messages.

As a comparison, Table 3 presents a summary of voter's messages and total amount of messages of different voting protocols. Note that differences in the amount of messages account for the different entities and primitives involved in each protocol. For example, protocols such as Cramer *et al.* (1997), Yang *et al.* (2018), Chen *et al.* (2008), Yang *et al.* (2020), Gao *et al.* (2019), Larriba *et al.* (2021), Larriba and López (2022) and our solution involve multiple parties in the voting process, which improves the security guarantees against an adversary colluding with a voting authority. As discussed in the related work, depending on the underlying technology, each multi-party protocol may have additional tradeoffs apart from the total amount of messages: a homomorphic solution may require additional zero knowledge proofs to guarantee the verifiability of the results, and it can be difficult to protect the privacy of the voters in a system implemented on a blockchain.

However, when comparing our solution with the rest of the electronic voting protocols, we want to emphasize that only a subset of the existing protocols address the resolution of disputes during the elections (Basin *et al.*, 2020; Bougon *et al.*, 2022; Larriba and López, 2022) as covered by Table 1, and that our protocol is the only one that considers the resolution of disputes related to vote suppression, while providing individual accountability through the whole election process. Therefore, our protocol effectively improves the security of elections against covert adversaries colliding with voters or authorities.

Regarding storage, DiReCT relies on the PBB to maintain the election state. Each entry in the PBB corresponds to a distinct phase of the protocol: certification request,

Table 3
Summary of messages sent by the voter and the total of messages for voting protocols and the primitives used. k represents the number of authorities in multi-party protocols, and v the number of processed votes. The * symbol represents systems deployed using blockchain. The symbol † represents systems that combine e-voting and physical booths. The system Themis does not define the details of the encryption used or the tally method, represented by ‡.

Protocol	Voter's messages	Total messages	Cryptographic primitives
Chaum (1981)	2	2v	Blind signatures
Li *et al.* (2009)	5	3v	Blind signatures
Nguyen Thi and Dang (2013)	4	7v	Blind signatures
Larriba *et al.* (2020)	3	2v+1	Blind signatures
Cramer *et al.* (1997)	1	v+k	Homomorphic prop.
Yang *et al.* (2017, 2018)	2	2(v+k)	Homomorphic prop.
Tornos *et al.* (2014)	3	3v+1	Ring signatures
Chen *et al.* (2008)	2	2+v+2k	Ring signatures
Yang *et al.* (2020)*	2	2k	Homomorphic prop.
Gao *et al.* (2019)*	2	v	Ring signatures
Larriba *et al.* (2021)*	2	2+v	Ring signatures
Basin *et al.* (2020)	2	$4v + 1$	Homomorphic prop.
Bougon *et al.* (2022)†	16	$26v$ + Tally‡	–
Larriba and López (2022)	$1 + k$	$2k \cdot v$	Blind signatures
This work	$3 + k$	$v(k + 8) + 3k$	Blind Signatures

Table 4
Example storage requirements for parameters of DiReCT.

Name	Instance	Example value
Tally Authorities	k	10
Hash	SHA-256	256B
Signature	RSA-2048	256B
Certificate	.x509	3KB
Timestamp	64 bits	8B
ID voter	$\log(v)$	20B
ID PBB	$2 * \log(v)$	21B
SP_j	(x, y)	8B
H-P	Hash	256B
H-SP_j	Hash	256B
BBallot	$(k + 1)$Hash	2816B
SBallot	$(k + 1) \cdot$ (Signature + Hash)	5632B

certification response, casting receipt, tally shares, and final count. The estimation of the storage is collected in Table 5. The storage requirements of each entry are determined by the cryptographic primitives and identifiers involved, as detailed in Table 4.

For instance, a certification request (CertRq) contains the voter identity, a blind ballot signature, the voter's certificate, and a timestamp. Using RSA-2048 signatures (256B), SHA-256 hashes (256B), and X.509 certificates (∼3KB), it results in an entry of approximately 3.7KB. Certification responses (CertRes) are much smaller (around 305B each), since they only include the signed blind ballot, the original request reference, and a timestamp. Casting receipts (Casting), which aggregate multiple signed shares, imply around 3.2KB each, while tally shares (Tally) and final count entries are compact (at 57B and 220B respectively).

Table 5
PBB storage evaluation, per type of entry and in total. Three size values are shown depending on the number of voters in the election. The size is calculated with the example values displayed in Table 4, with ten tally authorities ($k = 10$).

PBB Entry	Content	PBB size per n. of voters		
		10^3	10^6	$5 * 10^6$
CerRq	v, $\langle \text{Sign}_v(\textit{BBallot}), \text{CERT}_v \rangle$, T	3.7KB	3.7KB	3.7KB
CerRes	VA, $\langle \text{Sign}_{\text{VA}}(\textit{BBallot}), \text{CerRq}_v \rangle$, T	285B	305B	311B
Casting Receipt	party_i, $\langle \text{H-SP}_j, \textit{SBallot}_j \rangle$, T	3.2KB	3.2KB	3.2KB
Tally	party_i, $\langle \text{SP}_i, \text{H-SP}_i, \textit{SBallot} \rangle$, T_4	37B	57B	63B
Count	$[\text{party}_i, \text{count}]$	110B	220B	253B
Total	$v * (\text{CerRq} + \text{CerRes} + k * \text{Casting}) + k \cdot v \cdot \text{Tally} + \text{Count}$	34.6MB	24.5 GB	170.7 GB

Using these figures, we evaluate the storage requirements of a full election with $v = 10^6$ voters and $k = 10$ tally authorities. The total storage is given by

$$v \cdot (\text{CertRq} + \text{CertRes} + k \cdot \text{Casting}) + k \cdot v \cdot \text{Tally} + \text{Count},$$

which corresponds to approximately 24.5 GB of PBB data, a size well within the capacity of modern servers and, with hash-based storage, compatible with distributed ledger architectures.

8. Conclusions

This work presented DiReCT, an electronic voting protocol that improves the current state of the art in terms of accountability. In previous definitions of dispute resolution, for elections conducted by multiple parties the requirement was to detect when *any* of the parties misbehaves (Basin *et al.*, 2020), or to partially account them (Bougon *et al.*, 2022). We extended this requirement by providing individual accountability, which allows the identification of the misbehaving party. Without individual accountability, a malicious party could abuse the dispute resolution with impunity, diluting the blame into the rest of honest parties in the election.

In our protocol, the timeliness guarantee described by Basin *et al.* (2020) is effective during the casting, not at the end of the election. It is our intention to emphasize the importance of the verification timeliness in electronic voting. Ensuring that certain attacks can be detected during the elections is crucial for voters, who may be able to recover and continue voting.

Furthermore, DiReCT is the first protocol that addresses dispute resolution in the case of vote suppression. Although this attack is often overlooked in the literature, it's consequences are as effective as other election manipulation attacks.

With this work we highlight the benefits of including dispute resolution as part of voting protocols and showcase that is possible to achieve it in a multi party protocol. The threat model includes covert adversaries, which we believe is a realistic assumption

for voting protocols that combines nicely with the verification and accountability focus of DiReCT. Regarding the complexity and scalability analysis of DiReCT, although the number of messages needed by DiReCT increments with the number of tally authorities, it is still linear with the number of processed votes. Our analysis shows a reasonable estimated storage needed in the PBB (Table 5), only needing 25.5 GB for an election with one million voters. As a future work, we will address the formal verification and resolution of our results. We will also explore the usage of zero knowledge proofs as in Iovino *et al.* (2020), in particular regarding the benefits and trade-offs in terms of accountability.

Acknowledgements

Supported by INCIBE's Chair funded by the EU-NextGenerationEU through the Spanish government's Plan de Recuperacion, Transformacion y Resiliencia.

References

Adida, B. (2008). Helios: Web-based open-audit voting. In: *USENIX Security Symposium*, Vol. 17, pp. 335–348. https://dl.acm.org/doi/10.5555/1496711.1496734.

Adida, B., de Marneffe, O., Pereira, O., Quisquater, J.-J. (2009). Electing a University President using Open-Audit Voting: analysis of real-world use of Helios. In: *Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*. https://dl.acm.org/doi/10.5555/1855491.1855501.

Aumann, Y., Lindell, Y. (2010). Security against covert adversaries: efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2), 281–343. https://doi.org/10.1007/s00145-009-9040-7.

Basin, D., Radomirović, S., Schmid, L. (2020). Dispute resolution in voting. In: *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pp. 1–16, 2374-8303. https://doi.org/10.1109/CSF49147.2020.00009.

Bougon, M., Chabanne, H., Cortier, V., Debant, A., Dottax, E., Dreier, J., Gaudry, P., Turuani, M. (2022). Themis: an on-site voting system with systematic cast-as-intended verification and partial accountability. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Los Angeles CA USA, pp. 397–410. 978-1-4503-9450-5. https://doi.org/10.1145/3548606.3560563.

Canetti, R. (2001). Universally composable security: a new paradigm for cryptographic protocols. In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pp. 136–145. 1552-5244. https://doi.org/10.1109/SFCS.2001.959888.

Chaum, D. (1983). Blind Signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (Eds.), *Advances in Cryptology*. Springer US, Boston, MA, pp. 199–203. 978-1-4757-0602-4. https://doi.org/10.1007/978-1-4757-0602-4_18.

Chaum, D.L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), 84–90. https://dl.acm.org/doi/10.1145/358549.358563.

Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., Vora, P. (2008). Scantegrity: end-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy*, 6(3), 40–46. https://doi.org/10.1109/MSP.2008.70.

Chen, G., Wu, C., Han, W., Chen, X., Lee, H., Kim, K. (2008). A new receipt-free voting scheme based on linkable ring signature for designated verifiers. In: *2008 International Conference on Embedded Software and Systems Symposia*, pp. 18–23. https://doi.org/10.1109/ICESS.Symposia.2008.54.

Clarkson, M.R., Chong, S., Myers, A.C. (2008). Civitas: toward a secure voting system. In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 354–368. 2375-1207. https://doi.org/10.1109/SP.2008.32.

Consortium, E. (2017). STORK 2.0 Secure idenTity acrOss boRders linKed 2.0 | Interoperable Europe Portal.

Cramer, R., Gennaro, R., Schoenmakers, B. (1997). A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5), 481–490. https://doi.org/10.1002/ett.4460080506.

Cramer, R., Franklin, M., Schoenmakers, B., Yung, M. (1996). Multi-authority secret-ballot elections with linear work. In: Maurer, U. (Ed.), *Advances in Cryptology — EUROCRYPT '96*. Springer, Berlin, Heidelberg, pp. 72–83. 978-3-540-68339-1. https://doi.org/10.1007/3-540-68339-9_7.

Crimmins, B.L., Rhea, M., Halderman, J.A. (2023). RemoteVote and SAFE Vote: towards usable End-to-End verification for Vote-by-Mail. In: *Financial Cryptography and Data Security. FC 2022 International Workshops: CoDecFin, DeFi, Voting, WTSC, Grenada, May 6, 2022, Revised Selected Papers*. Springer-Verlag, Berlin, Heidelberg, pp. 391–406. 978-3-031-32414-7. https://doi.org/10.1007/978-3-031-32415-4_27.

Cuvelier, E., Pereira, O., Peters, T. (2013). Election verifiability or ballot privacy: do we need to choose? In: Crampton, J., Jajodia, S., Mayes, K. (Eds.), *Computer Security – ESORICS 2013*. Springer, Berlin, Heidelberg, pp. 481–498. 978-3-642-40203-6. https://doi.org/10.1007/978-3-642-40203-6_27.

Desmedt, Y., Frankel, Y. (1990). Threshold cryptosystems. In: Brassard, G. (Ed.), *Advances in Cryptology – CRYPTO' 89 Proceedings*. Springer, New York, NY, pp. 307–315. 978-0-387-34805-6. https://doi.org/10.1007/0-387-34805-0_28.

Doan, T.V.T., Pereira, O., Peters, T. (2025). Threshold receipt-free single-pass eVoting. In: Duenas-Cid, D., Roenne, P., Volkamer, M., Budurushi, J., Blom, M., Rodríguez-Pérez, A., Spycher-Krivonosova, I., Castellà Roca, J., Barrat Esteve, J. (Eds.), *Electronic Voting*. Springer Nature Switzerland, Cham, pp. 20–36. 978-3-031-72244-8. https://doi.org/10.1007/978-3-031-72244-8_2.

Dolev, D., Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), 198–208. Conference Name: IEEE Transactions on Information Theory. https://doi.org/10.1109/TIT.1983.1056650.

Duenas-Cid, D. (2024). Trust and distrust in electoral technologies: what can we learn from the failure of electronic voting in the Netherlands (2006/07). In: *Proceedings of the 25th Annual International Conference on Digital Government Research*, dg.o '24. Association for Computing Machinery, New York, NY, USA, pp. 669–677. 9798400709883. https://doi.org/10.1145/3657054.3657262.

Fujioka, A., Okamoto, T., Ohta, K. (1993). A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (Eds.), *Advances in Cryptology — AUSCRYPT '92*. Springer, Berlin, Heidelberg, pp. 244–251. 978-3-540-47976-5. https://doi.org/10.1007/3-540-57220-1_66.

Furukawa, J., Mori, K., Sako, K. (2010). An implementation of a mix-net based network voting scheme and its use in a private organization. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (Eds.), *Towards Trustworthy Elections: New Directions in Electronic Voting*. Springer, Berlin, Heidelberg, pp. 141–154. 978-3-642-12980-3. https://doi.org/10.1007/978-3-642-12980-3_8.

Gao, S., Zheng, D., Guo, R., Jing, C., Hu, C. (2019). An anti-quantum E-voting protocol in blockchain with audit function. *IEEE Access*, 7, 115304–115316. https://doi.org/10.1109/ACCESS.2019.2935895.

Gong, B., Lu, X., Fat, L.W., Au, M.H. (2019). Blockchain-based threshold electronic voting system. In: Meng, W., Furnell, S. (Eds.), *Security and Privacy in Social Networks and Big Data*. Springer, Singapore, pp. 238–250. 9789811507588. https://doi.org/10.1007/978-981-15-0758-8_18.

Haines, T., Boyen, X. (2016). VOTOR: Australasian Computer Science Week Multiconference, ACSW 2016. In: *Proceedings of the Australasian Computer Science Week Multiconference, ACSW 2016*. Association for Computing Machinery (ACM). https://doi.org/10.1145/2843043.2843362.

Heather, J., Lundin, D. (2009). The append-only web bulletin board. In: Degano, P., Guttman, J., Martinelli, F. (Eds.), *Formal Aspects in Security and Trust*, Vol. 5491. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 242–256. 978-3-642-01464-2 978-3-642-01465-9. https://doi.org/10.1007/978-3-642-01465-9_16.

Hwang, M.-S., Lee, C.-C., Lai, Y.-C. (2003). An untraceable blind signature scheme. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 86(7), 1902–1906. https://doi.org/10.1109/ICEBE.2006.25.

Iovino, V., Rial, A., Rønne, P.B., Ryan, P.Y.A. (2020). Universal unconditional verifiability in E-voting without trusted parties. In: *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pp. 33–48. 2374-8303. https://doi.org/10.1109/CSF49147.2020.00011.

Jakobsson, M., Juels, A., Rivest, R.L. (2002). Making mix nets robust for electronic voting by randomized partial checking. In: *11th USENIX Security Symposium (USENIX Security 02)*. https://dl.acm.org/doi/10.5555/647253.720294.

Juang, W.-S., Lei, C.-L., Liaw, H.-T. (2002). A verifiable multi-authority secret election allowing abstention from voting. *The Computer Journal*, 45(6), 672–682. https://doi.org/10.1093/comjnl/45.6.672.

Kiayias, A., Yung, M. (2002). Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (Eds.), *Public Key Cryptography*. Springer, Berlin, Heidelberg, pp. 141–158. 978-3-540-45664-3. https://doi.org/10.1007/3-540-45664-3_10.

Kremer, S., Ryan, M., Smyth, B. (2010). Election verifiability in electronic voting protocols. In: Gritzalis, D., Preneel, B., Theoharidou, M. (Eds.), *Computer Security – ESORICS 2010*. Springer, Berlin, Heidelberg, pp. 389–404. 978-3-642-15497-3. https://doi.org/10.1007/978-3-642-15497-3_24.

Küsters, R., Liedtke, J., Müller, J., Rausch, D., Vogt, A. (2020). Ordinos: a verifiable tally-hiding E-voting System. In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 216–235. https://doi.org/10.1109/EuroSP48549.2020.00022.

Lagrange, J.L. (1795). Leçon cinquieme: sur l'usage des courbes dans la solution des problemes. *Séances des Écoles Normales recueillies par les sténographes et revues par les professeurs, Reynier, Paris*.

Larriba, A.M., López, D. (2022). SUVS: secure unencrypted voting scheme. *Informatica*, 33(4), 749–769. https://doi.org/10.15388/22-INFOR503.

Larriba, A.M., Sempere, J.M., López, D. (2020). A two authorities electronic vote scheme. *Computers & Security*, 97, 101940. https://doi.org/10.1016/j.cose.2020.101940.

Larriba, A.M., Cerdà i Cucó, A., Sempere, J.M., López, D. (2021). Distributed trust, a blockchain election scheme. *Informatica*, 32(2), 321–355. https://doi.org/10.15388/20-INFOR440.

Lee, B., Kim, K. (2003). Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Goos, G., Hartmanis, J., Van Leeuwen, J., Lee, P.J., Lim, C.H. (Eds.), *Information Security and Cryptology — ICISC 2002*, Vol. 2587. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 389–406. 978-3-540-00716-6 978-3-540-36552-5. https://doi.org/10.1007/3-540-36552-4_27.

Li, C.-T., Hwang, M.-S., Lai, Y.-C. (2009). A verifiable electronic voting scheme over the Internet. In: *2009 Sixth International Conference on Information Technology: New Generations*, pp. 449–454. https://doi.org/10.1109/ITNG.2009.93.

Moran, T., Naor, M. (2010). Split-ballot voting: everlasting privacy with distributed trust. *ACM Transactions on Information and System Security*, 13(2), 1–43. https://doi.org/10.1145/1698750.1698756.

Mosaheb, R., Roenne, P., Ryan, P.Y.A., Sarfaraz, S. (2025). Direct and transparent voter verification with everlasting receipt-freeness. In: *Electronic Voting, E-Vote-ID 2024*. Springer Science and Business Media Deutschland GmbH. 978-3-031-72243-1. https://doi.org/10.1007/978-3-031-72244-8_8.

Nguyen Thi, A.T., Dang, T.K. (2013). Enhanced security in internet voting protocol using blind signatures and dynamic ballots. In: *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*, IIWAS '12. Association for Computing Machinery, New York, NY, USA, pp. 278–281. 978-1-4503-1306-3. https://doi.org/10.1145/2428736.2428781.

Oechslin, P. (2003). Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (Ed.), *Advances in Cryptology – CRYPTO 2003*. Springer, Berlin, Heidelberg, pp. 617–630. 978-3-540-45146-4. https://doi.org/10.1007/978-3-540-45146-4_36.

Okamoto, T. (1998). Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (Eds.), *Security Protocols*. Springer, Berlin, Heidelberg, pp. 25–35. 978-3-540-69688-9. https://doi.org/10.1007/BFb0028157.

Pankova, A., Willemson, J. (2022). Relations between privacy, verifiability, accountability and coercion-resistance in voting protocols. In: *Applied Cryptography and Network Security*, Vol. 13269. Springer International Publishing, Cham, pp. 313–333. 978-3-031-09233-6 978-3-031-09234-3. https://doi.org/10.1007/978-3-031-09234-3_16.

Pointcheval, D., Sanders, O. (2016). Short randomizable signatures. In: Sako, K. (Ed.), *Topics in Cryptology – CT-RSA 2016*. Springer International Publishing, Cham, pp. 111–126. 978-3-319-29485-8. https://doi.org/10.1007/978-3-319-29485-8_7.

Rivest, R.L., Adleman, L., Dertouzos, M.L. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11), 169–180.

Ryan, P.Y.A. (2008). Prêt à voter with Paillier encryption. *Mathematical and Computer Modelling*, 48(9), 1646–1662. https://doi.org/10.1016/j.mcm.2008.05.015.

Sebé, F., Miret, J.M., Pujolàs, J., Puiggalí, J. (2010). Simple and efficient hash-based verifiable mixing for remote electronic voting. *Computer Communications*, 33(6), 667–675. https://doi.org/10.1016/j.comcom.2009.11.013.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11), 612–613. https://doi.org/10.1145/359168.359176.

Tornos, J.L., Salazar, J.L., Piles, J.J., Saldana, J., Casadesus, L., Ruiz-Mas, J., Fernandez-Navajas, J. (2014). An eVoting system based on ring signatures. *Network Protocols and Algorithms*, 6(2), 38–54. https://doi.org/10.5296/npa.v6i2.5390.

Willemson, J. (2023). Analyzing and improving eligibility verifiability of the proposed Belgian remote voting system. In: Rios, R., Posegga, J. (Eds.), *Security and Trust Management*. Springer Nature Switzerland, Cham, pp. 126–135. 978-3-031-47198-8. https://doi.org/10.1007/978-3-031-47198-8_8.

Yang, X., Yi, X., Ryan, C., van Schyndel, R., Han, F., Nepal, S., Song, A. (2017). A verifiable ranked choice Internet voting system. In: Bouguettaya, A., Gao, Y., Klimenko, A., Chen, L., Zhang, X., Dzerzhinskiy, F., Jia, W., Klimenko, S.V., Li, Q. (Eds.), *Web Information Systems Engineering – WISE 2017*. Springer International Publishing, Cham, pp. 490–501. 978-3-319-68786-5. https://doi.org/10.1007/978-3-319-68786-5_39.

Yang, X., Yi, X., Nepal, S., Kelarev, A., Han, F. (2018). A secure verifiable ranked choice online voting system based on homomorphic encryption. *IEEE Access*, 6, 20506–20519. https://doi.org/10.1109/ACCESS.2018.2817518.

Yang, X., Yi, X., Nepal, S., Kelarev, A., Han, F. (2020). Blockchain voting: publicly verifiable online voting protocol without trusted tallying authorities. *Future Generation Computer Systems*, 112, 859–874. https://doi.org/10.1016/j.future.2020.06.051.

Zou, X., Li, H., Li, F., Peng, W., Sui, Y. (2017). Transparent, auditable, and stepwise verifiable online E-voting enabling an open and fair election. *Cryptography*, 1(2), 13. https://doi.org/10.3390/cryptography1020013.

J.L. Martin-Navarro obtained his computer science degree in 2019 at Polytechnic University of Valencia (Spain), a master's degree in security and cloud computing in 2022 at Aalto University (Finland), a master of science in electrical engineering and computer science (communication system) in 2022 at KTH Royal Institute of Technology (Sweden), and he is currently working in his PhD thesis at Aalto University and Polytecnic University of Valencia. In 2018 he was awarded with a National mobility program SICUE at Universidad Carlos III de Madrid (Spain) and he was with Spanish National Research Council (Madrid, Spain) as a Junior Researcher in 2020. His topics of interest include cryptography and information security, secure development and deployment, and cloud computing.

A.M. Larriba obtained his computer science degree in 2016, a master's degree in artificial intelligence in 2017, and defended his PhD Thesis in 2023, all at Polytechnic University of Valencia (Spain). He was awarded with a governmental grant for this purpose. His topics of interest include cryptography, artificial intelligence, and blockchain.

D. López is an associate professor at the Universitat Politècnica de València (Spain), where he obtained his PhD in computer science in 2003. Currently, he is an accademic coordinator of the Computation Section at the Departamento de Sistemas Informáticos y Computación, and a member of the Valencian Research Institute for Artificial Intelligence (VRAIN). His research interests include cryptography, formal languages, and grammatical inference.