VoteSentinel: Post-quantum Verifiable Coercion-resistant Voting with Dispute Resolution*

Anonymous

Abstract. Electronic voting systems have to satisfy challenging requirements: balancing the need for transparency for verifiability, while pleasing voters with secrecy and protection. These systems need to resist both classical and quantum adversaries. Specifically, no work to date attests to a scheme with provably strong privacy guarantees and post-quantum security simultaneously. Classical systems, such as Helios and Microsoft ElectionGuard, become vulnerable to clash attacks, ballot stuffing, and coercion through confirmation codes. All in all, electronic voting is widely used, and designing post-quantum systems is needed, but post-quantum voting schemes not only suffer from a lack of formal ballot/participation privacy proofs from strong definitions, but also have no guarantee for coercion resistance. In this work, we introduce VoteSentinel, the first voting scheme that achieves a wide range of security properties, including coercion resistance without using expensive zero-knowledge proofs or signature schemes under a single framework. Our security analysis shows that VoteSentinel closes a large array of vulnerabilities previously discovered in other schemes, while supporting ranked-choice and re-voting. Moreover, VoteSentinel is key-loss resilient through threshold cryptography: in scenarios like the IACR 2025 incident, the election can complete even when some trustees lose their keys.

Keywords: Applied Cryptography, Electronic Voting, Ballot Privacy, Coercion Resistance, Post-quantum Cryptography

1 Introduction

The idea of electronic voting is to modernize democratic processes through more accessibility and efficiency; however, designing schemes that can meet a wide range of security and privacy requirements remains one of the challenging problems in cryptography [1]. The paradox lies between transparency toward public verifiability and a high level of privacy for individual votes [2]. This challenge gets more complicated by the advent of quantum computers, which will be in a position to retroactively break today's classically encrypted ballots [3]. In spite of the fact that classical schemes have made significant progress in verifiability, they remain vulnerable to sophisticated attacks. Some Helios variants suffer from clash

* This document contains the full version of VoteSentinel scheme, along with proofs.

attacks, where adversaries can force ballot collisions [4]. ElectionGuard's confirmation codes [5] enable coercion through forced disclosure.

Moving towards post-quantum cryptography introduces additional complexity through larger key sizes and more intricate algebraic structures [6]. Transition to post-quantum e-voting needs to happen as soon as possible, because of the Store Now, Decrypt Later attack [**?**], in which encrypted data is harvested today with the intention of decrypting it in the future, posing a big risk to information that must remain secure long term. Existing lattice-based voting schemes, such as NTRU Mix-Net [7], provide quantum resistance but lack formal ballot/participation privacy proofs from strong definitions and fail to address other essential properties, e.g., dispute resolution, accountability, etc.

Our observations show that a flexible voting scheme should be designed such that no entity can learn the content of ballots (*ballot privacy* [8]), participation patterns remain hidden from observers (*participation privacy* [9]), voters can verify their ballots were recorded and anyone can verify correct tallying (*verifiability* [10]), the coercer cannot know whether voter followed his instruction or not (*coercion resistance* [11]), ballot secrecy gets preserved against computationally unbounded future adversaries (*everlasting privacy* [12]), only the most recent vote from each voter influences the final tally (*strong non-reusability* [13]), voters can update their choices multiple times (*re-voting* [14]), conflicts between voters and authorities are resolvable through cryptographic evidence (*dispute resolution* [15]), misbehaving parties can be identified and held responsible (*accountability* [16,17]), the protocol execution is indistinguishable from interaction with a trusted party (*ideal functionality* [18]), voters cannot construct proofs of their choices to third parties (*receipt-freeness* [19]), the election can complete even when some trustees lose their secret keys (*key-loss resilience* [20]), and voters can express preferences by ranking candidates in some cases (*ranked-choice voting* [21]). However, currently, there is no voting scheme that can simultaneously provide all of these properties; there are many schemes that each target one or a very limited number of these properties, not all of them in one place, as it is difficult to provide more than 2 of these properties at the same time [6], and it is challenging.

1.1 Contributions

These gaps motivated us to design a voting system that integrates a wide range of security properties, as Table 1 shows, instead of designing a core scheme and then attempting to achieve different separated properties by repeatedly designing some variants of it, as most classical approaches do. First, we recall and identify some attack vectors and conceptual weaknesses of the existing classical and post-quantum voting schemes, such as clash attacks, as well as a lack of strong formal privacy proofs. Next, we present VoteSentinel, a verifiable voting scheme that not only achieves all of the aforementioned properties together at the same time through a unified threshold-homomorphic framework with nullifier-based ballot resolution, but also brings post-quantum-level security, combining ML-KEM for secure channels, BFV for homomorphic tallying, and VSS for distributed trust, our scheme addresses vulnerabilities in existing schemes. Then, based upon this, we

prove the privacy of VoteSentinel for every claimed property under the standard lattice assumptions.

1.2 Organization

Section 2 surveys classical and post-quantum voting schemes, identifying vulnerabilities that motivate our design. Section 3 establishes cryptographic foundations, defining ML-KEM, BFV, and VSS. Section 4 presents the VoteSentinel construction with its encoding methods, system model, and protocol specification. Section 5 states that VS satisfies privacy and verifiability properties. Section 6 concludes with future directions.

2 Related Works

We check some of the existing electronic voting schemes, analyzing classical systems (Helios [22], Belenios [23], ElectionGuard [5]) and post-quantum approaches (Epoque [24], NTRU Mix-Net [7], FOO-based [3], Code-based [6]) to identify fundamental vulnerabilities and limitations that VS addresses through its lattice-based dual-chain architecture.

2.1 Classical Voting Approaches

Helios The voting scheme Helios[1] is a web-based voting system that prioritizes verifiability over coercion resistance, making it suitable for low-coercive environments. It employs exponential ElGamal encryption with homomorphic tallying: voters encrypt ballots using public key pk generated via distributed key generation among trustees, encrypted votes are homomorphically aggregated as $\prod_{i=1}^{n} E_{pk}(v_i) = E_{pk}(\sum_{i=1}^{n} v_i)$, and threshold decryption reveals only the final tally [22]. However, several variants of Helios are vulnerable to Clash Attacks [4], and upon compromise of the server, they can be attacked by Ballot Stuffing, and Individual Verifiability violations [25] and other privacy attacks [26] as well.

Belenios Another voting system known as Belenios[2] is a web-based verifiable voting system led by Stéphane Glondu that improves Helios through formal verification. It employs exponential ElGamal encryption with homomorphic tallying and threshold decryption, but enhances Helios via ballot re-randomization $E_{pk}(v;r) \to E_{pk}(v;r')$ for preventing malleability [23]. Clash Attacks, Ballot Reordering Attacks, and Strong Ballot Stuffing [25] under the corrupted registrar are some of the Belenios' vulnerabilities. Developers have published further issues with Belenios regarding fragile vote privacy and the mixnet tally [27], as well.

[1] https://vote.heliosvoting.org
[2] https://www.belenios.org

Microsoft ElectionGuard This is an open-source framework developed by Microsoft Research that enables verifiable electronic voting systems. It allows third parties to independently verify the integrity of election results without compromising voter privacy [5]. ElectionGuard[3] gives the ability for voters, candidates, observers, and independent organizations to verify election results. Despite all the new components, ElectionGuard can be considered, in essence, a modification of Cramer et al. [28]. Although ElectionGuard is a software development kit, it is not compatible with all existing modes of casting votes.

Helios has been well-studied during the past decades, and Belenios is ultimately a variant of it; the weaknesses of these two schemes have already been observed. In the following, we provide our findings of ElectionGuard's downsides.

ElectionGuard does not support ranked-choice voting. Additionally, in ElectionGuard, voters receive confirmation codes $c_i = H(E_{pk}(v_i))$ for their encrypted ballots, which enables coercion: adversaries with decryption capabilities can demand these codes to verify compliance. Public disclosure of confirmation codes compromises ballot privacy, as c_i uniquely links voters to their encrypted ballots $E_{pk}(v_i)$, enabling unauthorized decryption and vote recovery [29]. In voting systems that use this framework with cumulative voting, the discrete logarithm sizes grow beyond small values when voters cast multiple votes for a candidate, and ElectionGuard's precomputed DLog tables are susceptible to attacks, expose keys, lack generality, and suffer from efficiency trade-offs [29]. The secrecy of ballots in ElectionGuard is tied to a single value known as the ballot nonce [5]. These values must be carefully protected because exposure of this random value can directly lead to the adversary decrypting votes[4]. Up to today, ElectionGuard has not been formalized yet; also, no proof for ballot privacy, or receipt-freeness, or participation privacy has been provided for it.

None of the Helios, Belenios, and ElectionGuard provides solutions for dispute resolution. ElectionGuard addresses minor complaints during the election initialization process, but does not provide formal and robust answers. Furthermore, to our knowledge, none of the classical and post-quantum schemes give any robust result recoverability for scenarios like IACR 2025 incident [20,31,32], and existing schemes have weak resilience to key-loss.

The next section provides an overview of the state-of-the-art in electronic voting schemes that are resistant to attacks by quantum computers, similar to VoteSentinel.

2.2 Post-quantum Voting Schemes

NTRU Mix-Net [7] NTRU Mix-Net employs NTRUEncrypt with single-element ciphertexts and verifiable mixing. The NTRU parameters must be carefully chosen to avoid the overstretched regime where $q > 0.0058 \cdot \sigma^2 \cdot d^{2.484}$ enables subfield lattice attacks, and the scheme's privacy fundamentally relies on the assumption that at least one mix server remains honest without providing detection mechanisms for malicious mixing beyond proof verification.

[3] https://www.electionguard.vote

[4] `EG v2.1` [30] somehow addresses this issue, but at the cost of encrypting the nonce itself.

FOO-based Voting [3] FOO-based Voting combines threshold lattice-based blind signatures with Dual-Regev encryption for everlasting privacy via perfectly hiding commitments. Authorities generate threshold shares for signature and encryption keys using Shamir secret sharing over polynomial rings. The one-more unforgeability of the underlying lattice-based blind signature scheme remains conjectural, as noted by Hauck et al. [33], potentially allowing adversaries to obtain additional valid signatures beyond those authorized; furthermore, the scheme requires a perfectly anonymous channel for ballot submission, which, if compromised, breaks the everlasting privacy guarantee despite the perfectly hiding commitments.

Code-based E-voting [6] Code-based E-voting [6] employs LPN assumption for commitments and ZK proofs, constituting the first code-based verifiable e-voting system. The scheme relies on the relatively less-studied dual-LPN assumption for the commitment binding property, which, unlike standard LPN, lacks an extensive cryptanalytic background; additionally, the concrete parameter selection for achieving 128-bit security against Information Set Decoding attacks remains uncertain, given recent advances in code-based cryptanalysis and the lack of standardized parameter recommendations.

Epoque [24] Epoque employs the ABB lattice-based IBE for verifiability without mix-nets. Unlike other similar schemes, Epoque shows accountability and privacy proof, but even assuming their proof is valid, developers use a very limited definition of ballot privacy known as *"Vote Privacy"* [34]. It has been shown that this limited definition restricts the class of protocols or privacy breaches [18]. Specifically, this definition may lead to inconsistent results when applied to protocols that use natural result functions, such as the majority function [35]. Additionally, the definition may not apply to protocols that output auxiliary data beyond just the result, such as proofs of correct tallying that most protocols produce, e.g., ElectionGuard [5]. The definition also appears to only detect specific types of privacy breaches, such as determining whether two voters cast the same vote, while missing other potential privacy violations [18]. Epoque developers explicitly state that it lacks coercion resistance; they also refer to the wrong work while discussing this by pointing to [36], but BeleniosRF is not coercion resistant, and receipt-freeness does not imply coercion resistance[5].

Apart from the aforementioned disadvantages, none of these four post-quantum schemes [3, 6, 7, 24] provides wide-range proofs for their the cryptographic privacy properties that an e-voting system should have[6], as mentioned in Section 1.

As Figure 1 shows, in Section 4 we present a voting scheme VoteSentinel that satisfies a broad range of privacy properties simultaneously with many additional

[5] Even the definition that has been used to show Epoque is verifiable [37] is not strong as shown in [10] and does not distinguish between attack scenarios where honest voters' ballots are dropped (deleted) versus those where ballots are actually modified to contain different choices, even though ballot modification has a significantly stronger negative impact on election outcomes than ballot deletion.

[6] Code-based Voting [6] shares some claims but no proof of ballot/participation privacy.

capabilities (re-voting, ranked voting, plausible deniable, etc.) designed using only three primitives as core (ML-KEM)+BFV+VSS.

Property/PQE-voting Scheme	Epoque [24]	NTRU Mix-Net [7]	FOO-based [3]	Code-based [6]	VoteSentinel (Ours)
Computational Assumption	RLWE,RSIS	RLWE,MSIS,RSIS	RLWE,OM-RSIS,MSIS	Dual-LPN,SDP,MDP	RLWE,MLWE
Core Building Blocks	IBE [38]+Com [39]+ZKP [40]	NTRU [41]+Shuffle [42]+ZKP [43]	BS [44]+PKE [45]	Com [6, 46]+ZKP [6, 46]+Shuffle [6]	ML-KEM [47]+BFV [48]+VSS [49]
Ballot Privacy Proof [18]	✓	×	×	×	✓
Participation Privacy Proof [9]	×	×	×	×	✓
Individual Verifiability [10]	✓	✓	Partial	✓	✓
Universal Verifiability [10]	✓	✓	✓	✓	✓
Everlasting Privacy Proof [50]	×	×	×	×	✓
Receipt-Freeness [19]	Partial	✓	×	Partial	✓
Accountability [17]	✓	×	×	×	✓
Ideal Functionality Proof [18]	×	×	×	×	✓
Coercion Resistance [19]	×	×	×	×	✓
Dispute Resolution [15]	×	×	×	×	✓
Vote Updating/Revoting [14]	×	×	×	×	✓
Homomorphic Tallying [35]	✓	×	×	×	✓
Trusted Setup	Required	Partial	Required	Not Needed	Required
Honest Majority Talliers	Required	Required	Required	Required	Required
Trusted Bulletin Board	Required	Required	Required	Required	Required

Table 1: Comparison of post-quantum electronic voting schemes [3, 6, 7, 24] by structural components, privacy properties, trust assumptions, and features. ✓indicates satisfying an specific definition, and × shows not providing a desirable feature.

There are a couple of other post-quantum schemes [51–56], but either their structure is different than the aforementioned systems and not similar to ours, or these are more strict for specific scenarios. For example, despite the scheme of Abapour et al. [51] having ballot privacy proof, it is exclusively for economic structures and, due to the utilization of blockchain, does not have enough flexibility to be used for other purposes [57] like Helios.

3 Preliminaries

Throughout this work, we define several mini-voting schemes separately to demonstrate the property definitions independently, thereby avoiding overlap. Let λ denote the security parameter. We write $x \leftarrow\!\!\$\, S$ for uniform sampling from set S, and $x \leftarrow \mathcal{D}$ for sampling from distribution $\mathcal{D}$. We denote by $\mathsf{negl}(\lambda)$ a negligible function in λ. Let $R_q = \mathbb{Z}_q[X]/(X^n+1)$ be the polynomial ring where n is a power of 2.

Definition 1 (Hash Function [58]). *A collision-resistant deterministic map $H : \{0,1\}^* \to \{0,1\}^\lambda$, from arbitrary-length inputs to fixed-length λ bits outputs.*

Definition 2 (Commitment Scheme Com [59]). *A commitment scheme* $\mathsf{Com} = (\mathsf{Cmt},\mathsf{Opn})$ *consists of two sub-procedures for obligation of parties:*

- $\gamma \leftarrow \mathsf{Cmt}(m;r)$*: On input message m and randomness $r \in \{0,1\}^\lambda$, outputs commitment $\gamma = H(m\|r)$.*
- $\{0,1\} \leftarrow \mathsf{Opn}(\gamma,m,r)$*: Outputs 1 iff $\gamma = H(m\|r)$.*

Definition 3 (Key Encapsulation [47]). *The key encapsulation mechanism* $\mathsf{ML\text{-}KEM} = (\mathsf{KG},\mathsf{Enc},\mathsf{Dec})$ *based on Module Learning With Errors* MLWE *[60] consists of:*

- $(\mathsf{ek},\mathsf{dk}) \leftarrow \mathsf{KG}(1^\lambda)$: *Outputs encapsulation key* ek *and decapsulation key* dk.
- $(K,\psi) \leftarrow \mathsf{Enc}(\mathsf{ek})$: *Outputs shared key* $K \in \{0,1\}^\lambda$ *and ciphertext* ψ.
- $K \leftarrow \mathsf{Dec}(\mathsf{dk},\psi)$: *Outputs shared key* K *or* $\perp$.

Definition 4 (Homomorphic Encryption BFV [48]). *A homomorphic encryption scheme* $\mathsf{BFV} = (\mathsf{KG},\mathsf{Enc},\mathsf{Dec},\mathsf{Add})$ *based on Ring Learning With Errors* RLWE *[60] with plaintext space* R_p *and ciphertext space* R_q^2 *consists of:*

- $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$: *Outputs public key* $\mathsf{pk} = (a,b) \in R_q^2$ *and secret key* $\mathsf{sk} \in R_q$.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk},m;\rho)$: *On input* $m \in R_p$ *and randomness* ρ, *outputs ciphertext* $c = (c_0,c_1) \in R_q^2$.
- $m \leftarrow \mathsf{Dec}(\mathsf{sk},c)$: *Outputs plaintext* $m \in R_p$.
- $c' \leftarrow \mathsf{Add}(c_1,c_2)$: *Outputs* c' *satisfying* $\mathsf{Dec}(\mathsf{sk},c') = \mathsf{Dec}(\mathsf{sk},c_1) + \mathsf{Dec}(\mathsf{sk},c_2)$.

Definition 5 (Verifiable Secret Sharing VSS [49]). *A* (t,n)*-threshold verifiable secret sharing scheme* $\mathsf{VSS} = (\mathsf{Shr},\mathsf{Vf},\mathsf{Rec},\mathsf{PDec},\mathsf{Comb})$ *over* R_q *based on post-quantum hash-based commitment [61] consists of:*

- $(\{u_i\}_{i\in[n]},\{W_j\}_{j\in[0,t-1]}) \leftarrow \mathsf{Shr}(s,t,n)$: *Samples* $\{a_j\}_{j=1}^{t-1} \leftarrow\!\!\$\, R_q^{t-1}$, *defines polynomial* $f(X) = s + \sum_{j=1}^{t-1} a_j X^j$, *outputs shares* $u_i = f(i)$ *and commitments* $W_j = H(a_j)$ *where* $a_0 = s$.
- $\{0,1\} \leftarrow \mathsf{Vf}(i,u_i,\{W_j\}_{j\in[0,t-1]})$: *Outputs 1 iff share* u_i *is consistent with commitments.*
- $s \leftarrow \mathsf{Rec}(\{(i,u_i)\}_{i\in S})$: *For* $|S| \geq t$, *computes* $\mu_i = \prod_{j\in S\setminus\{i\}} \frac{j}{j-i}$ *and outputs* $s = \sum_{i\in S} \mu_i \cdot u_i$.
- $\delta_i \leftarrow \mathsf{PDec}(u_i,c)$: *Outputs partial decryption* $\delta_i = c_1 \cdot u_i$.
- $m \leftarrow \mathsf{Comb}(\{(i,\delta_i)\}_{i\in S},c,t)$: *Computes* μ_i, *then* $\tilde{c}_1 = \sum_{i\in S} \mu_i \cdot \delta_i$, *outputs* $m = \lfloor p(c_0+\tilde{c}_1)/q \rceil \bmod p$.

4 VoteSentinel

4.1 Vote Encoding Methods

We use One-hot [62], Coefficient Encoding [35], and bit-vector [63]/bit-packing [64] techniques for encoding single-choice, ranked-choice, and identifier, respectively.

Definition 6 (Single-Choice Encoding [62]). *For* k *candidates, define* $\mathsf{EncS}: [k] \to R_p$ *by* $\mathsf{EncS}(v) = X^{v-1}$ *and* $\mathsf{DecS}: R_p \to \mathbb{Z}^k$ *extracting coefficients. To validate, all we need to do is to calculate* $\sum_{i=0}^{k-1} a_i = 1$ *per ballot.*

Definition 7 (Ranked-Choice Encoding [35]). *For* k *candidates with ranking* $\pi = (\pi_1,\ldots,\pi_k)$ *where* $\pi_i \in [0,k-1]$ *is points for candidate* i, *define* $\mathsf{EncR}: [0,k-1]^k \to R_p$ *by* $\mathsf{EncR}(\pi) = \sum_{i=1}^{k} \pi_i \cdot X^{i-1}$. *Validity:* $\sum_{i=0}^{k-1} a_i = k(k-1)/2$ *per ballot.*

Definition 8 (Identifier Encoding [64]). *For nullifier base* $\nu \in \{0,1\}^\lambda$ *and counter* $\ell \in [0,L)$, *define* $\mathsf{EncID}: \{0,1\}^\lambda \times [0,L) \to R_p$ *by* $\mathsf{EncID}(\nu,\ell) = \sum_{i=0}^{\lambda-1} \nu[i] \cdot X^i + \ell \cdot X^\lambda$ *where* $\nu[i]$ *is the* i*-th bit of* ν.

Setup(1^λ,n,t,k,N,L,mode)

$(d,q,p) \leftarrow \mathsf{Param}(1^\lambda)$ s.t. $d > \lambda + \lceil \log_2 L \rceil \wedge p > N \cdot k^2$
$H \leftarrow \mathsf{HashGen}(1^\lambda)$; $a \leftarrow\!\!\$\, R_q$; $\mathcal{E} \leftarrow \mathtt{eligibleVoters}$
for $i \in [n]$ **do** $(\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{ML\text{-}KEM.KG}(1^\lambda)$
for $i \in [n]$ **do**
  $\alpha_i \leftarrow\!\!\$\, R_q$
  $(\{u_{i,j}\}_{j\in[n]}, \{W_{i,\ell}\}_{\ell\in[0,t-1]}) \leftarrow \mathsf{VSS.Shr}(\alpha_i, t, n)$
  $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{com\text{-}vss}, i, \{W_{i,\ell}\}_\ell)\}$
for $i \in [n]$ **do**
  for $j \in [n] \setminus \{i\}$ **do**
    $(K_{i,j}, \psi_{i,j}) \leftarrow \mathsf{ML\text{-}KEM.Enc}(\mathsf{ek}_j)$
    $\sigma_{i,j} \leftarrow u_{i,j} \oplus K_{i,j}$
    $\mathsf{Send}(T_i \to T_j : (\psi_{i,j}, \sigma_{i,j}))$
$\mathcal{Q} \leftarrow [n]$
for $j \in [n]$ **do**
  for $i \in [n] \setminus \{j\}$ **do**
    $K_{i,j} \leftarrow \mathsf{ML\text{-}KEM.Dec}(\mathsf{dk}_j, \psi_{i,j})$
    $\hat{u}_{i,j} \leftarrow \sigma_{i,j} \oplus K_{i,j}$
    if $\mathsf{VSS.Vf}(j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell) = 0$ **then**
      $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{complain}, j, i, \hat{u}_{i,j})\}$
      $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{i\}$
  $s_j \leftarrow u_{j,j} + \sum_{i \in \mathcal{Q} \setminus \{j\}} \hat{u}_{i,j}$
if $|\mathcal{Q}| < t$ **then return** $\perp$
for $i \in \mathcal{Q}$ **do**
  $\varepsilon_i \leftarrow\!\!\$\, \chi_{\mathsf{err}}$; $b_i \leftarrow -a \cdot s_i + \varepsilon_i$
  $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{pk\text{-}share}, i, b_i)\}$
$b \leftarrow \sum_{i \in \mathcal{Q}} b_i$; $\mathsf{pk} \leftarrow (a, b)$
for $i \in \mathcal{Q}$ **do**
  $\omega_i \leftarrow\!\!\$\, \{0,1\}^\lambda$; $\Gamma_i \leftarrow H(s_i \| \omega_i)$
  $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{com\text{-}key}, i, \Gamma_i)\}$
$\mathsf{pp} \leftarrow (d, q, p, t, k, N, L, \mathsf{mode}, H, \mathsf{pk}, \{\Gamma_i\}_{i\in\mathcal{Q}}, \mathcal{Q}, \mathcal{E})$
return $(\mathsf{pp}, \{(s_i, \omega_i)\}_{i\in\mathcal{Q}})$

Reg(pp,j)

if $j \notin \mathcal{E} \vee j \in \mathcal{R}$ **then return** $\perp$
$\xi_j \leftarrow\!\!\$\, \{0,1\}^{2\lambda}$
$\nu_j \leftarrow H(\xi_j \| j \| \mathtt{0x00})$
$\mathcal{R} \leftarrow \mathcal{R} \cup \{j\}$; $\mathsf{cred}_j \leftarrow (\xi_j, \nu_j)$
return cred_j

Vote(pp,cred_j,v,ℓ)

$(\xi_j, \nu_j) \leftarrow \mathsf{cred}_j$
if mode = single $\wedge\, v \notin [k]$ **then return** $\perp$
if mode = ranked $\wedge\, v \notin [0,k-1]^k$ **then return** $\perp$
if $\ell \notin [0,L)$ **then return** $\perp$
$\rho_1 \leftarrow H(\xi_j \| \ell \| \mathtt{0x01})$
$\rho_2 \leftarrow H(\xi_j \| \ell \| \mathtt{0x02})$
$r \leftarrow H(\xi_j \| \ell \| \mathtt{0x03})$
if mode = single **then** $m \leftarrow \mathsf{EncS}(v)$
else $m \leftarrow \mathsf{EncR}(v)$
$c \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, m; \rho_1)$
$m' \leftarrow \mathsf{EncID}(\nu_j, \ell)$
$c' \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, m'; \rho_2)$
$\gamma \leftarrow \mathsf{Com.Cmt}(c \| c' \| \nu_j \| \ell; r)$
$\beta \leftarrow (c, c', \gamma)$
$\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{ballot}, \beta)\}$
return (β, r)

Tally(pp,BB,$\{(s_i,\omega_i)\}_{i\in S}$)

if $|S| < t$ **then return** $\perp$
$\mathcal{B} \leftarrow \{(c,c',\gamma) : (\mathsf{ballot}, (c,c',\gamma)) \in \mathsf{BB}\}$
for $(c,c',\gamma) \in \mathcal{B}$ **do**
  for $i \in S$ **do**
    $\delta'_i \leftarrow \mathsf{VSS.PDec}(s_i, c')$
  $m'_{c'} \leftarrow \mathsf{VSS.Comb}(\{(i, \delta'_i)\}_{i\in S}, c', t)$
  $(\nu_{c'}, \ell_{c'}) \leftarrow \mathsf{DecID}(m'_{c'})$
$\mathcal{M} \leftarrow \emptyset$
for $(c,c',\gamma) \in \mathcal{B}$ **do**
  if $\nu_{c'} \notin \mathsf{dom}(\mathcal{M}) \vee \mathcal{M}[\nu_{c'}].\ell < \ell_{c'}$ **then**
    $\mathcal{M}[\nu_{c'}] \leftarrow (c, \ell_{c'})$
$\mathcal{F} \leftarrow \{c : \exists \nu, (c, \cdot) = \mathcal{M}[\nu]\}$
$\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{final\text{-}set}, \mathcal{F})\}$
if $|\mathcal{F}| > N$ **then return** $(\perp, \mathsf{stuffing})$
$C \leftarrow (0,0)$
for $c \in \mathcal{F}$ **do** $C \leftarrow \mathsf{BFV.Add}(C, c)$
$\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{aggregate}, C)\}$
for $i \in S$ **do**
  $\delta_i \leftarrow \mathsf{VSS.PDec}(s_i, C)$
  $\pi_i \leftarrow H(\delta_i \| \Gamma_i)$
  $\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{pdec}, i, \delta_i, \pi_i)\}$
$M \leftarrow \mathsf{VSS.Comb}(\{(i, \delta_i)\}_{i\in S}, C, t)$
if mode = single **then**
  $(r_0, \ldots, r_{k-1}) \leftarrow \mathsf{DecS}(M)$
  if $\sum_{i=0}^{k-1} r_i \neq |\mathcal{F}|$ **then return** $(\perp, \mathsf{invalid})$
else
  $(r_0, \ldots, r_{k-1}) \leftarrow \mathsf{DecR}(M)$
  if $\sum_{i=0}^{k-1} r_i \neq |\mathcal{F}| \cdot k(k-1)/2$ **then return** $(\perp, \mathsf{invalid})$
$\mathsf{res} \leftarrow (r_0, \ldots, r_{k-1})$
$\mathsf{BB} \leftarrow \mathsf{BB} \cup \{(\mathsf{result}, \mathsf{res})\}$
return $(\mathsf{res}, \mathcal{F}, \{\delta_i\}_{i\in S})$

Vf(pp,BB,mode,aux)

if mode = ind **then**
  $(\mathsf{cred}_j, v, \ell, r) \leftarrow \mathsf{aux}$
  $(\xi_j, \nu_j) \leftarrow \mathsf{cred}_j$
  $\rho_1 \leftarrow H(\xi_j \| \ell \| \mathtt{0x01})$
  $\rho_2 \leftarrow H(\xi_j \| \ell \| \mathtt{0x02})$
  if pp.mode = single **then** $m \leftarrow \mathsf{EncS}(v)$
  else $m \leftarrow \mathsf{EncR}(v)$
  $c \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, m; \rho_1)$
  $m' \leftarrow \mathsf{EncID}(\nu_j, \ell)$
  $c' \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, m'; \rho_2)$
  $\gamma \leftarrow \mathsf{Com.Cmt}(c \| c' \| \nu_j \| \ell; r)$
  return $(\mathsf{ballot}, (c,c',\gamma)) \in \mathsf{BB} \wedge c \in \mathcal{F}$
if mode = univ **then**
  $(\mathsf{res}, C, \mathcal{F}, \{\delta_i\}_{i\in S}) \leftarrow \mathsf{aux}$
  if $|\mathcal{F}| > N$ **then return** 0
  $C' \leftarrow (0,0)$
  for $c \in \mathcal{F}$ **do** $C' \leftarrow \mathsf{BFV.Add}(C', c)$
  if $C' \neq C$ **then return** 0
  for $i \in S$ **do**
    if $(\mathsf{pdec}, i, \delta_i, \pi_i) \in \mathsf{BB} \wedge \pi_i \neq H(\delta_i \| \Gamma_i)$ **then**
      return 0
  $(r_0, \ldots, r_{k-1}) \leftarrow \mathsf{res}$
  if pp.mode = single $\wedge \sum_i r_i \neq |\mathcal{F}|$ **then return** 0
  if pp.mode = ranked $\wedge \sum_i r_i \neq |\mathcal{F}| \cdot k(k-1)/2$ **then return** 0
  return 1

Jdg(pp,BB,E)

if $\mathsf{pk} \notin R_q^2$ **then return** $(\mathsf{EA}, \mathsf{err}_1)$
$\mathcal{B} \leftarrow \{(\mathsf{ballot}, \cdot) \in \mathsf{BB}\}$
if $|\mathcal{B}| > N \cdot L$ **then return** $(\mathsf{EA}, \mathsf{err}_2)$
$(\mathsf{final\text{-}set}, \mathcal{F}) \leftarrow \mathsf{BB}$
if $|\mathcal{F}| > N$ **then return** $(\mathsf{Trustees}, \mathsf{err}_3)$
for $(\mathsf{type}, \mathsf{data}) \in E$ **do**
  if type = drop **then**
    $(j, c, c', \gamma, \nu_j, \ell, r) \leftarrow \mathsf{data}$
    if $\mathsf{Com.Opn}(\gamma, c \| c' \| \nu_j \| \ell, r) = 1$ **then**
      if $(\mathsf{ballot}, (c,c',\gamma)) \notin \mathsf{BB}$ **then**
        return $(\mathsf{EA}, \mathsf{err}_4, j)$
  if type = not-counted **then**
    $(j, c, c', \nu_j, \ell) \leftarrow \mathsf{data}$
    if $(\mathsf{ballot}, (c,c',\cdot)) \in \mathsf{BB} \wedge c \notin \mathcal{F}$ **then**
      $m' \leftarrow \mathsf{EncID}(\nu_j, \ell)$
      if $\nexists \ell' > \ell : \mathsf{EncID}(\nu_j, \ell')$ in any $c'' \in \mathsf{BB}$ **then**
        return $(\mathsf{Trustees}, \mathsf{err}_5, j)$
  if type = bad-pdec **then**
    $(i, \delta_i, s_i, \omega_i) \leftarrow \mathsf{data}$
    if $H(s_i \| \omega_i) = \Gamma_i$ **then**
      $(\mathsf{aggregate}, C) \leftarrow \mathsf{BB}$
      $\delta'_i \leftarrow \mathsf{VSS.PDec}(s_i, C)$
      if $(\mathsf{pdec}, i, \delta_i, \cdot) \in \mathsf{BB} \wedge \delta'_i \neq \delta_i$ **then**
        return (T_i, err_6)
  if type = bad-share **then**
    $(i, j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell) \leftarrow \mathsf{data}$
    if $(\mathsf{com\text{-}vss}, i, \{W_{i,\ell}\}_\ell) \in \mathsf{BB} \wedge \mathsf{VSS.Vf}(j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell) = 0$ **then**
      return (T_i, err_7)
return $\perp$

Fig. 1: Voting System $\mathsf{VS} = (\mathsf{Setup}, \mathsf{Reg}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Vf}, \mathsf{Jdg})$ based on Key Encapsulation $\mathsf{ML\text{-}KEM} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$, Fully Homomorphic Encryption Scheme $\mathsf{BFV} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add})$, and Verifiable Secret Sharing $\mathsf{VSS} = (\mathsf{Shr}, \mathsf{Vf}, \mathsf{Rec}, \mathsf{PDec}, \mathsf{Comb})$

4.2 System Model and Parties

VoteSentinel involves five categories of participants, each with distinct responsibilities and trust assumptions.

Election Authority (EA). The election authority is a central coordinating entity responsible for initializing the election, managing voter registration, and maintaining the bulletin board. The EA determines the eligibility set $\mathcal{E}$ containing identifiers of all voters permitted to participate. During registration, the EA generates credentials for eligible voters and tracks who has registered via the set $\mathcal{R}$. The EA also receives ballots from voters and appends them to the bulletin board. While the EA has significant operational responsibilities, the protocol is designed such that EA misbehavior (dropping ballots, accepting excess ballots) is detectable and attributable through the dispute resolution mechanism.

Trustees T_1,...,T_n. The trustees collectively manage the election's cryptographic keys through threshold cryptography. No single trustee possesses the complete decryption capability; instead, decryption requires cooperation of at least t trustees, where $t = \lceil (n+1)/2 \rceil$ ensures an honest majority suffices. During setup, each trustee contributes to the distributed key generation protocol, receiving a share s_i of the implicit secret key. During tallying, trustees perform partial decryptions that are combined to recover the election result. The threshold structure provides key-loss resilience: if up to $n-t$ trustees lose their keys, the remaining t trustees can still complete the election.

Voters V_1,...,V_N. Voters are individuals from the eligibility set $\mathcal{E}$ who wish to cast ballots. Each voter V_j receives a credential cred_j during registration and uses it to construct ballots during voting. Voters may submit up to L ballots, with only the most recent counting toward the final tally. This revoting capability enables coercion resistance: a coerced voter can comply with the coercer's demands, then later submit their true preference.

Bulletin Board (BB). The bulletin board is a public, append-only data structure that records all election-related information. Every party can read BB, and authorized parties can append entries. The append-only property ensures that once data is recorded, it cannot be modified or deleted. Entries on BB are tagged with identifiers indicating their type: com-vss for VSS commitments, pk-share for public key shares, com-key for key commitments, ballot for submitted ballots, final-set for the filtered ballot set, aggregate for the homomorphic sum, pdec for partial decryptions, and result for the final tally.

Judge (J). The judge is an abstract algorithm that evaluates evidence to determine accountability when disputes arise. The judge examines publicly available data on BB together with evidence provided by complainants, and outputs either the identity of a misbehaving party or $\perp$ if no misbehavior is detected. The judge's decisions are publicly verifiable since all inputs are public or can be verified against public commitments.

4.3 Parameters and Notation

The protocol operates with the following parameters, each serving a specific purpose in the system's security and functionality.

Security Parameter λ. The security parameter λ determines the cryptographic strength of all primitives. Typical values are $\lambda = 128$ or $\lambda = 256$. Hash outputs, random seeds, and cryptographic keys are all λ bits or derived from λ-bit entropy sources.

Ring Parameters (d,q,p). The polynomial ring $R_q = \mathbb{Z}_q[X]/(X^d+1)$ provides the algebraic structure for BFV encryption. The degree d is a power of 2 (e.g., $d = 2048$ or $d = 4096$) and must satisfy $d > \lambda + \lceil \log_2 L \rceil$ to accommodate the identifier encoding which stores λ bits of the nullifier base plus $\lceil \log_2 L \rceil$ bits for the counter. The ciphertext modulus q is a large prime determining the size of ciphertext coefficients. The plaintext modulus p defines the plaintext space $R_p = \mathbb{Z}_p[X]/(X^d+1)$ and must satisfy $p > N \cdot k^2$ to ensure that homomorphic aggregation of up to N votes, each with coefficients up to $k-1$, does not cause overflow.

Threshold Parameters (t,n). The system employs n trustees with reconstruction threshold $t = \lceil (n+1)/2 \rceil$. This choice ensures that an honest majority (at least t honest trustees) guarantees security, while tolerating the failure or corruption of up to $n-t$ trustees. For example, with $n = 3$ trustees, $t = 2$ allows the election to proceed even if one trustee loses their key.

Election Parameters (k,N,L,mode). The number of candidates k determines the vote encoding dimension. The voter count N bounds the maximum number of distinct voters. The revote limit L specifies how many ballots each voter may submit. The mode $\mathsf{mode} \in \{\mathsf{single}, \mathsf{ranked}\}$ determines whether the election uses single-choice voting (one candidate selected) or ranked-choice voting (points assigned to each candidate).

Hash Function H. The collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ is used throughout the protocol for commitments, key derivation, and integrity verification. Collision resistance ensures that no efficient adversary can find distinct inputs $x \neq x'$ with $H(x) = H(x')$.

Public Key pk. The BFV public key $\mathsf{pk} = (a,b) \in R_q^2$ enables encryption of votes. The element a is a common random ring element, while $b = \sum_{i \in \mathcal{Q}} b_i$ is the sum of individual trustee contributions $b_i = -a \cdot s_i + \varepsilon_i$. The implicit secret key is $\mathsf{sk} = \sum_{i \in \mathcal{Q}} s_i$, which no single party knows.

Trustee Commitments $\{\Gamma_i\}_{i \in \mathcal{Q}}$. Each qualified trustee T_i publishes commitment $\Gamma_i = H(s_i \| \omega_i)$ binding them to their secret share s_i using random salt ω_i. These commitments enable accountability: if a trustee provides an incorrect partial decryption, another party with knowledge of (s_i,ω_i) can prove the commitment opens to a share that would yield a different partial decryption.

Qualified Set $\mathcal{Q}$. The qualified set $\mathcal{Q} \subseteq [n]$ contains trustees who successfully completed the setup protocol without valid complaints against them. Only qualified trustees contribute to the public key and participate in decryption.

Eligibility and Registration Sets $(\mathcal{E},\mathcal{R})$. The eligibility set $\mathcal{E}$ is established during setup and contains identifiers of all voters permitted to participate. The registration set $\mathcal{R}$, initially empty, tracks which eligible voters have registered. A voter j can register only if $j \in \mathcal{E}$ and $j \notin \mathcal{R}$.

4.4 Setup Phase

The setup phase establishes all cryptographic infrastructure required for the election. It involves the election authority and all n trustees executing a distributed protocol.

Parameter Generation. The election authority begins by generating the BFV parameters (d,q,p) satisfying the constraints $d > \lambda + \lceil \log_2 L \rceil$ and $p > N \cdot k^2$. The first constraint ensures sufficient polynomial degree for identifier encoding; the second prevents overflow during vote aggregation. The EA also instantiates the hash function H and samples a common public ring element $a \in R_q$ uniformly at random. This element a will be shared across all trustees' public key computations, ensuring they contribute to a common key structure. Finally, the EA establishes the eligibility set $\mathcal{E}$ containing identifiers of all eligible voters.

ML-KEM Key Generation. Each trustee T_i independently generates an ML-KEM key pair by invoking $(\mathsf{ek}_i,\mathsf{dk}_i) \leftarrow \mathsf{ML\text{-}KEM.KG}(1^\lambda)$. The encapsulation key ek_i is a public key that any party can use to establish a shared secret with T_i. The decapsulation key dk_i is kept secret by T_i and enables recovery of shared secrets from ciphertexts. Trustee T_i publishes ek_i so that other trustees can securely communicate with them.

Secret Contribution and Sharing. Each trustee T_i samples a secret contribution $\alpha_i \in R_q$ uniformly at random. This value represents T_i's contribution to the distributed secret key. Trustee T_i then invokes $\mathsf{VSS.Shr}(\alpha_i, t, n)$ to create a (t,n)-threshold sharing of α_i. Internally, this algorithm samples $t-1$ random coefficients $a_1,\ldots,a_{t-1} \in R_q$ and constructs the polynomial $f_i(X) = \alpha_i + a_1 X + \cdots + a_{t-1} X^{t-1}$. The algorithm outputs shares $u_{i,j} = f_i(j)$ for each $j \in [n]$ and commitments $W_{i,\ell} = H(a_\ell)$ for $\ell \in [0, t-1]$, where $a_0 = \alpha_i$. The commitments enable verification that shares lie on a polynomial of degree at most $t-1$ without revealing the polynomial coefficients. Trustee T_i publishes the commitments on BB as the entry $(\mathsf{com\text{-}vss}, i, \{W_{i,\ell}\}_{\ell \in [0,t-1]})$.

Secure Share Distribution. Each trustee T_i must transmit share $u_{i,j}$ to every other trustee T_j confidentially. To achieve this without a trusted channel, T_i uses the ML-KEM key encapsulation mechanism. For each $j \in [n] \setminus \{i\}$, trustee T_i invokes $(K_{i,j}, \psi_{i,j}) \leftarrow \mathsf{ML\text{-}KEM.Enc}(\mathsf{ek}_j)$, obtaining a shared key $K_{i,j} \in \{0,1\}^\lambda$

and ciphertext $\psi_{i,j}$. The share is then masked using one-time pad encryption: $\sigma_{i,j} = u_{i,j} \oplus K_{i,j}$. Trustee T_i sends the pair $(\psi_{i,j},\sigma_{i,j})$ to T_j. The security of ML-KEM ensures that only T_j, possessing dk_j, can recover $K_{i,j}$ and hence $u_{i,j}$.

Share Recovery and Verification. The qualified set is initialized as $\mathcal{Q} = [n]$, containing all trustees. Each trustee T_j processes incoming messages from all other trustees. For each $i \in [n] \setminus \{j\}$, trustee T_j receives $(\psi_{i,j}, \sigma_{i,j})$ and recovers the shared key by invoking $K_{i,j} \leftarrow \mathsf{ML\text{-}KEM.Dec}(\mathsf{dk}_j, \psi_{i,j})$. The share is then unmasked as $\hat{u}_{i,j} = \sigma_{i,j} \oplus K_{i,j}$. Trustee T_j verifies the share's validity by invoking $\mathsf{VSS.Vf}(j,\hat{u}_{i,j},\{W_{i,\ell}\}_{\ell\in[0,t-1]})$, which checks that $\hat{u}_{i,j}$ is consistent with T_i's published commitments. If verification fails, T_j posts a complaint $(\mathsf{complain},j,i,\hat{u}_{i,j})$ to BB as evidence of T_i's misbehavior, and T_i is removed from $\mathcal{Q}$. After processing all shares, T_j computes their aggregate secret share as $s_j = u_{j,j} + \sum_{i\in\mathcal{Q}\setminus\{j\}} \hat{u}_{i,j}$. By construction, if $f = \sum_{i\in\mathcal{Q}} f_i$ is the combined polynomial, then $s_j = f(j)$ for all $j \in \mathcal{Q}$.

Threshold Verification. If $|\mathcal{Q}| < t$, too many trustees have been disqualified, and the setup protocol fails. In this case, the election cannot proceed and must be restarted, potentially with different trustees. Otherwise, the protocol continues with the qualified set $\mathcal{Q}$.

Public Key Construction. Each qualified trustee T_i (for $i \in \mathcal{Q}$) contributes to the public key construction. Trustee T_i samples an error term ε_i from the error distribution χ_{err} (a discrete Gaussian over R_q with small standard deviation) and computes $b_i = -a \cdot s_i + \varepsilon_i$. This computation follows the BFV key generation structure: for a single party with secret s and error ε, the public key would be $(a,-a\cdot s+\varepsilon)$. In the distributed setting, each trustee contributes b_i corresponding to their share s_i. Trustee T_i publishes b_i on BB as $(\mathsf{pk\text{-}share},i,b_i)$. The combined public key is $\mathsf{pk} = (a,b)$ where $b = \sum_{i\in\mathcal{Q}} b_i = -a\cdot(\sum_{i\in\mathcal{Q}} s_i) + \sum_{i\in\mathcal{Q}} \varepsilon_i$. The implicit secret key is $\mathsf{sk} = \sum_{i\in\mathcal{Q}} s_i = f(0)$ by Lagrange interpolation, since $f(0) = \sum_{i\in\mathcal{Q}} \alpha_i$. No single trustee knows sk; it exists only implicitly as the sum of the individual shares.

Key Commitment. Each qualified trustee T_i samples a random salt $\omega_i \in \{0,1\}^\lambda$ and computes commitment $\Gamma_i = H(s_i \| \omega_i)$. This commitment binds T_i to their share s_i without revealing it. Trustee T_i publishes Γ_i on BB as $(\mathsf{com\text{-}key},i,\Gamma_i)$ and retains (s_i,ω_i) secretly. The commitment serves accountability: if T_i later provides an incorrect partial decryption, any party who learns (s_i,ω_i) (e.g., from another trustee who detected misbehavior) can prove that the commitment opens to a share yielding a different partial decryption.

Setup Output. The public parameters $\mathsf{pp} = (d,q,p,t,k,N,L,\mathsf{mode},H,\mathsf{pk},\{\Gamma_i\}_{i\in\mathcal{Q}},\mathcal{Q},\mathcal{E})$ are published on BB and available to all parties. The secret state $\{(s_i,\omega_i)\}_{i\in\mathcal{Q}}$ is distributed among the qualified trustees, with each T_i holding only their own (s_i,ω_i).

4.5 Registration Phase

The registration phase establishes credentials for eligible voters. It is managed by the election authority and occurs after setup is complete.

Registration Request. A voter identified by index j initiates registration by contacting the election authority through an authenticated channel. The authentication mechanism (outside the scope of this protocol) ensures the EA can verify the voter's identity corresponds to index j.

Eligibility and Uniqueness Checks. The EA verifies two conditions: first, $j \in \mathcal{E}$, confirming the voter is in the eligibility set established during setup; second, $j \notin \mathcal{R}$, confirming the voter has not already registered. If either check fails, registration is rejected and the EA returns $\perp$.

Credential Generation. Upon successful verification, the EA generates a credential for voter V_j. The EA samples a master seed $\xi_j \in \{0,1\}^{2\lambda}$ uniformly at random. The seed length 2λ provides sufficient entropy to derive multiple cryptographically independent values: the protocol derives three hash outputs (nullifier base and two domain-separated randomness values per vote), and 2λ bits ensures these derivations remain statistically independent even under multiple uses. From the master seed, the EA computes the nullifier base $\nu_j = H(\xi_j \| j \| \texttt{0x00})$, where the byte `0x00` serves as a domain separator. The nullifier base is a λ-bit value unique to voter j that will be used to link multiple ballots from the same voter during revoting while keeping the voter's identity hidden from public observers.

State Update. The EA adds j to the registration set: $\mathcal{R} \leftarrow \mathcal{R} \cup \{j\}$. This prevents the voter from registering again.

Credential Delivery. The EA constructs the credential $\mathsf{cred}_j = (\xi_j, \nu_j)$ and transmits it to voter V_j through a secure channel. The credential consists of the master seed ξ_j (from which all voting randomness will be derived) and the nullifier base ν_j (precomputed for efficiency). The voter must keep this credential secret; possession of ξ_j enables creating ballots that will be counted as V_j's votes.

Privacy Properties. Unlike some voting systems that publish voter identifiers during registration, VoteSentinel publishes no voter-identifying information on BB at this stage. The registration set $\mathcal{R}$ is maintained privately by the EA. This design supports participation privacy: external observers cannot determine which eligible voters have registered. The trustees will learn participation information during tallying (since they decrypt identifier ciphertexts), but this information is not publicly revealed.

4.6 Voting Phase

The voting phase allows registered voters to cast ballots. Each ballot contains an encrypted vote and an encrypted identifier, ensuring ballots on BB are unlinkable to voters.

Input Validation. Voter V_j prepares to cast vote v with revote counter ℓ. The protocol first validates these inputs against the election mode. For single-choice mode ($\mathsf{mode} = \mathsf{single}$), the vote v must be an integer in $[k] = \{1,...,k\}$ representing the selected candidate. For ranked-choice mode ($\mathsf{mode} = \mathsf{ranked}$), the vote v must be a vector $(v_1,...,v_k) \in [0,k-1]^k$ where v_i represents the points assigned to candidate i. The counter ℓ must satisfy $\ell \in [0,L) = \{0,1,...,L-1\}$, where $\ell = 0$ for the first ballot, $\ell = 1$ for the second, and so forth. If validation fails, the protocol returns $\perp$.

Deterministic Randomness Derivation. All randomness used in ballot construction is derived deterministically from the credential $\mathsf{cred}_j = (\xi_j, \nu_j)$ and the counter ℓ. This deterministic derivation serves two purposes: it enables the voter to later verify their ballot by recomputation, and it ensures that different votes or different revote attempts produce independent randomness. Three values are derived using domain-separated hashing: $\rho_1 = H(\xi_j \| \ell \| \mathtt{0x01})$ provides randomness for encrypting the vote, $\rho_2 = H(\xi_j \| \ell \| \mathtt{0x02})$ provides randomness for encrypting the identifier, and $r = H(\xi_j \| \ell \| \mathtt{0x03})$ provides randomness for the commitment. The domain separators $\mathtt{0x01}$, $\mathtt{0x02}$, $\mathtt{0x03}$ ensure these outputs are cryptographically independent even though they derive from the same seed.

Vote Encoding. The vote v is encoded as a polynomial $m \in R_p$ using an encoding scheme that supports homomorphic aggregation. For single-choice voting, $m = \mathsf{EncS}(v) = X^{v-1}$: a polynomial with coefficient 1 at position $v-1$ and 0 elsewhere. This one-hot encoding ensures that summing N such polynomials yields a polynomial whose coefficient at position i counts how many voters selected candidate $i+1$. The validity property is that each valid single-choice encoding contributes exactly 1 to the sum of coefficients: $\sum_{i=0}^{k-1} a_i = 1$. For ranked-choice voting, $m = \mathsf{EncR}(v) = \sum_{i=1}^{k} v_i \cdot X^{i-1}$: a polynomial whose coefficient at position $i-1$ equals the points v_i assigned to candidate i. Summing N such polynomials yields aggregate point totals per candidate. The validity property assumes each voter assigns points $0,1,...,k-1$ (a permutation), so the coefficient sum is $\sum_{i=0}^{k-1} a_i = 0+1+\cdots+(k-1) = k(k-1)/2$ per ballot.

Vote Encryption. The encoded vote m is encrypted using BFV encryption: $c \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, m; \rho_1)$. The ciphertext $c = (c_0, c_1) \in R_q^2$ hides the vote m under the semantic security (IND-CPA) of BFV. The homomorphic property ensures that $\mathsf{BFV.Dec}(\mathsf{sk}, \mathsf{BFV.Add}(c, c')) = \mathsf{BFV.Dec}(\mathsf{sk}, c) + \mathsf{BFV.Dec}(\mathsf{sk}, c')$, enabling aggregation of encrypted votes without decrypting them individually.

Identifier Encoding and Encryption. The identifier encoding $m' = \mathsf{EncID}(\nu_j, \ell)$ embeds the nullifier base ν_j and counter ℓ into a polynomial. Specifically, $m' = \sum_{i=0}^{\lambda-1} \nu_j[i] \cdot X^i + \ell \cdot X^\lambda$, where $\nu_j[i]$ denotes the i-th bit of ν_j. The first λ coefficients store the bits of the nullifier base, and coefficient λ stores the counter. The constraint $d > \lambda + \lceil \log_2 L \rceil$ ensures sufficient polynomial degree. This identifier is encrypted as $c' \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, m'; \rho_2)$. The ciphertext c' hides the voter's identity from public observers; only the trustees, who can decrypt c' during tallying, learn the identifier.

Commitment Construction. The voter computes a commitment as the value of $\gamma = \mathsf{Com.Cmt}(c\|c'\|\nu_j\|\ell;r) = H(c\|c'\|\nu_j\|\ell\|r)$. This commitment binds the ballot components (c,c') to the voter's identifier (ν_j,ℓ) using randomness r. The commitment serves dispute resolution: if the voter's ballot is dropped from BB, the voter can reveal (c,c',ν_j,ℓ,r) to prove they constructed a valid ballot. The binding property of the commitment (from collision resistance of H) ensures the voter cannot later claim a different ballot was dropped.

Ballot Structure and Submission. The complete ballot is $\beta = (c,c',\gamma)$, consisting of the vote ciphertext, identifier ciphertext, and commitment. The voter submits β to the EA, who appends (ballot,β) to BB. The voter retains the opening value r (which can be recomputed from ξ_j and ℓ) for later verification.

Coercion Resistance Analysis. The ballot $\beta = (c,c',\gamma)$ on BB reveals nothing about the voter's identity to public observers. The vote ciphertext c is semantically secure under BFV. The identifier ciphertext c' encrypts the nullifier base and counter, but decryption requires the secret key shares held by trustees. The commitment γ is a hash that reveals nothing without the opening. Consequently, a coercer observing BB cannot link ballots to voters, cannot determine if a particular voter has voted, and cannot detect if a voter has revoted. The coercer can demand the voter reveal their credential cred_j, but the voter can comply with coercion initially (casting the demanded vote with counter ℓ) and later cast their true preference (with counter $\ell+1$). Since ballots are unlinkable on BB, the coercer cannot verify whether the voter revoted.

4.7 Tallying Phase

The tallying phase computes the election result from the ballots on BB. It requires cooperation of at least t trustees from $\mathcal{Q}$ and proceeds in five stages: identifier decryption, nullifier resolution, homomorphic aggregation, threshold decryption, and result extraction.

Participation Requirement. Tallying requires a set $S \subseteq \mathcal{Q}$ of participating trustees with $|S| \geq t$. Each participating trustee T_i contributes their secret share s_i and salt ω_i. The threshold property ensures that if $n-t$ trustees are unavailable (due to key loss, corruption, or non-participation), the remaining t trustees can complete tallying. This provides the key-loss resilience property: for $n = 3$ and $t = 2$, one trustee can lose their key without preventing tally completion.

Ballot Collection. All ballots are extracted from BB into the set $\mathcal{B} = \{(c,c',\gamma) : (\mathsf{ballot},(c,c',\gamma)) \in \mathsf{BB}\}$. This set contains every ballot submitted during the voting phase, including multiple ballots from voters who revoted.

Identifier Decryption. For each ballot $(c,c',\gamma) \in \mathcal{B}$, the trustees jointly decrypt the identifier ciphertext c' to recover the encoded identifier. Each participating trustee T_i computes their partial decryption $\delta'_i = \mathsf{VSS.PDec}(s_i,c') = c'_1 \cdot s_i$, where $c' =$

(c'_0,c'_1). The partial decryptions are combined using Lagrange interpolation: first, the Lagrange coefficients $\mu_i = \prod_{j\in S\setminus\{i\}} \frac{j}{j-i}$ are computed; then, $\tilde{c}'_1 = \sum_{i\in S} \mu_i \cdot \delta'_i$; finally, the plaintext is recovered as $m'_{c'} = \lfloor p\cdot(c'_0+\tilde{c}'_1)/q \rceil \bmod p$. The identifier is decoded as $(\nu_{c'},\ell_{c'}) = \mathsf{DecID}(m'_{c'})$, extracting the λ-bit nullifier base from the low coefficients and the counter from coefficient λ. This decryption happens within the trustee collective and is not published on BB, preserving voter identity privacy.

Nullifier Resolution. The nullifier resolution stage implements the last-vote-counts policy using a map $\mathcal{M}$ from nullifier bases to (vote ciphertext, counter) pairs. For each ballot $(c,c',\gamma) \in \mathcal{B}$ with decoded identifier $(\nu_{c'},\ell_{c'})$: if $\nu_{c'}$ is not in the domain of $\mathcal{M}$ (this voter hasn't been seen) or $\mathcal{M}[\nu_{c'}].\ell < \ell_{c'}$ (this ballot has a higher counter), then $\mathcal{M}[\nu_{c'}]$ is updated to $(c,\ell_{c'})$. After processing all ballots, $\mathcal{M}$ maps each unique nullifier base to the vote ciphertext with the highest counter. This ensures that if voter V_j submitted ballots with counters 0,1,2, only the ballot with counter 2 (the most recent) is retained.

Final Set Construction. The final set $\mathcal{F} = \{c : \exists \nu, (c,\cdot) = \mathcal{M}[\nu]\}$ contains exactly one vote ciphertext per participating voter. The trustees publish $(\mathsf{final\text{-}set},\mathcal{F})$ on BB. If $|\mathcal{F}| > N$, more ballots exist in the final set than eligible voters, indicating ballot stuffing. In this case, tallying aborts with error $\mathsf{stuffing}$. The publication of $\mathcal{F}$ (containing only vote ciphertexts c, not identifier ciphertexts c') enables public verification while preserving voter anonymity.

Homomorphic Aggregation. The aggregate ciphertext C is computed by summing all vote ciphertexts in $\mathcal{F}$. The computation initializes $C = (0,0) \in R_q^2$ and iteratively updates $C \leftarrow \mathsf{BFV.Add}(C,c)$ for each $c \in \mathcal{F}$. By the additive homomorphic property of BFV, $\mathsf{BFV.Dec}(\mathsf{sk},C) = \sum_{c\in\mathcal{F}} \mathsf{BFV.Dec}(\mathsf{sk},c) = \sum_{c\in\mathcal{F}} m_c$, where m_c is the encoded vote in ciphertext c. The aggregate ciphertext is published on BB as $(\mathsf{aggregate},C)$.

Threshold Decryption. Each participating trustee T_i computes their partial decryption of the aggregate ciphertext: $\delta_i = \mathsf{VSS.PDec}(s_i,C) = C_1 \cdot s_i$, where $C = (C_0,C_1)$. Trustee T_i also computes a proof $\pi_i = H(\delta_i \| \Gamma_i)$ linking the partial decryption to their key commitment. The pair (δ_i,π_i) is published on BB as $(\mathsf{pdec},i,\delta_i,\pi_i)$. The proof π_i enables accountability: anyone can verify that the published δ_i is consistent with Γ_i (given the opening), and an incorrect δ_i constitutes evidence of misbehavior.

Combination. The partial decryptions are combined to recover the plaintext aggregate: $M = \mathsf{VSS.Comb}(\{(i,\delta_i)\}_{i\in S},C,t)$. This computes Lagrange coefficients μ_i, combines $\tilde{C}_1 = \sum_{i\in S} \mu_i \cdot \delta_i$, and recovers $M = \lfloor p\cdot(C_0+\tilde{C}_1)/q \rceil \bmod p$.

Decoding and Validity Check. The result is decoded according to the voting mode. For single-choice mode, $(r_0,\ldots,r_{k-1}) = \mathsf{DecS}(M)$ extracts the coefficients of M, where r_i is the vote count for candidate $i+1$. The validity check verifies $\sum_{i=0}^{k-1} r_i = |\mathcal{F}|$: since each valid single-choice ballot contributes exactly 1 to the

coefficient sum, the total must equal the number of ballots. For ranked-choice mode, $(r_0,...,r_{k-1}) = \mathsf{DecR}(M)$ extracts aggregate points per candidate. The validity check verifies $\sum_{i=0}^{k-1} r_i = |\mathcal{F}| \cdot k(k-1)/2$: since each valid ranking contributes $0+1+\cdots+(k-1) = k(k-1)/2$ to the total, the aggregate sum must equal this value times the ballot count. If the validity check fails, at least one ballot contains a malformed encoding, and tallying returns error $\mathsf{invalid}$.

Result Publication. The final result $\mathsf{res} = (r_0, \ldots, r_{k-1})$ is published on BB as $(\mathsf{result}, \mathsf{res})$. The tally returns $(\mathsf{res}, \mathcal{F}, \{\delta_i\}_{i \in S})$, providing all information needed for verification.

4.8 Verification Phase

The verification phase enables both individual voters and public observers to confirm election integrity. Two modes are supported: individual verification (for voters checking their own ballot) and universal verification (for anyone checking the overall tally).

Individual Verification. A voter V_j who cast vote v with counter ℓ verifies their ballot was recorded and counted. The voter provides auxiliary input as $\mathsf{aux} = (\mathsf{cred}_j, v, \ell, r)$ containing their credential, vote, counter, and commitment opening. The verification algorithm recomputes all ballot components deterministically as: $\rho_1 = H(\xi_j \| \ell \| \mathtt{0x01})$ and $\rho_2 = H(\xi_j \| \ell \| \mathtt{0x02})$ for encryption randomness; $m = \mathsf{EncS}(v)$ or $m = \mathsf{EncR}(v)$ depending on mode; $c = \mathsf{BFV.Enc}(\mathsf{pk}, m; \rho_1)$ for the vote ciphertext; $m' = \mathsf{EncID}(\nu_j, \ell)$ for the identifier encoding; $c' = \mathsf{BFV.Enc}(\mathsf{pk}, m'; \rho_2)$ for the identifier ciphertext; and $\gamma = H(c \| c' \| \nu_j \| \ell \| r)$ for the commitment. Verification succeeds if and only if $(\mathsf{ballot}, (c, c', \gamma)) \in \mathsf{BB}$ (the ballot was recorded) AND $c \in \mathcal{F}$ (the ballot was included in the final tally). The second condition confirms that this was the voter's most recent ballot and was not superseded by a revote.

Universal Verification. Any observer can verify the tally is correct given the ballots in $\mathcal{F}$. The observer provides auxiliary input $\mathsf{aux} = (\mathsf{res}, C, \mathcal{F}, \{\delta_i\}_{i \in S})$ from BB. Verification proceeds through several checks. First, the ballot count check verifies $|\mathcal{F}| \leq N$, ensuring no more ballots are counted than eligible voters. Second, the aggregation check recomputes the aggregate C' by initializing $C' = (0,0)$ and updating $C' \leftarrow \mathsf{BFV.Add}(C', c)$ for each $c \in \mathcal{F}$, then verifies $C' = C$. Third, the partial decryption check verifies each proof $\pi_i = H(\delta_i \| \Gamma_i)$ matches the published values. Fourth, the result consistency check verifies the validity condition: $\sum_i r_i = |\mathcal{F}|$ for single-choice or $\sum_i r_i = |\mathcal{F}| \cdot k(k-1)/2$ for ranked-choice. If all checks pass, the verifier is assured that the published result correctly aggregates the ballots in $\mathcal{F}$.

4.9 Dispute Resolution Phase

The dispute resolution phase handles accusations of misbehavior. The judge algorithm Jdg processes evidence to identify culpable parties.

Evidence Types. Evidence E is a list of (type, data) pairs representing different accusations. The supported types are drop (ballot was dropped), $\mathsf{not\text{-}counted}$ (ballot not in final set), $\mathsf{bad\text{-}pdec}$ (incorrect partial decryption), and $\mathsf{bad\text{-}share}$ (invalid share distribution).

Error 1: Invalid Setup (err_1). The judge first checks whether $\mathsf{pk} \in R_q^2$. If the public key is malformed (not a valid pair of ring elements), the election authority is blamed for allowing an invalid setup to proceed.

Error 2: Ballot Stuffing at Submission (err_2). The judge counts ballots on BB: $|\mathcal{B}| = |\{(\mathsf{ballot}, \cdot) \in \mathsf{BB}\}|$. If $|\mathcal{B}| > N \cdot L$, more ballots exist than possible from N voters each submitting at most L ballots. The EA is blamed for accepting excess ballot submissions.

Error 3: Ballot Stuffing at Tally (err_3). The judge retrieves $(\mathsf{final\text{-}set}, \mathcal{F})$ from BB and checks $|\mathcal{F}| \leq N$. If $|\mathcal{F}| > N$, the final set contains more ballots than voters, and the trustees collectively are blamed for incorrect nullifier resolution.

Error 4: Dropped Ballot (err_4). Evidence of type drop contains $(j, c, c', \gamma, \nu_j, \ell, r)$: the voter index, ballot components, identifier values, and commitment opening. The judge verifies the commitment opens correctly: $\mathsf{Com.Opn}(\gamma, c \| c' \| \nu_j \| \ell, r) = 1$, which checks $\gamma = H(c \| c' \| \nu_j \| \ell \| r)$. If the commitment is valid but $(\mathsf{ballot}, (c, c', \gamma)) \notin \mathsf{BB}$, the EA is blamed for dropping voter j's ballot. The valid commitment proves the voter constructed a legitimate ballot that was not recorded.

Error 5: Ballot Not Counted (err_5). Evidence of type $\mathsf{not\text{-}counted}$ contains the values (j, c, c', ν_j, ℓ). The judge checks whether $(\mathsf{ballot}, (c, c', \cdot)) \in \mathsf{BB}$ but $c \notin \mathcal{F}$. If so, the ballot was recorded but not counted. The judge then checks whether a later revote exists: if there is no $\ell' > \ell$ such that $\mathsf{EncID}(\nu_j, \ell')$ appears encrypted in any ballot on BB, then the ballot should have been counted but was improperly excluded. The trustees are blamed for incorrect nullifier resolution.

Error 6: Incorrect Partial Decryption (err_6). Evidence of type $\mathsf{bad\text{-}pdec}$ contains $(i, \delta_i, s_i, \omega_i)$: the trustee index, published partial decryption, and claimed share with salt. The judge first verifies the share commitment: if $H(s_i \| \omega_i) = \Gamma_i$, the claimed share s_i is indeed trustee T_i's committed share. The judge retrieves $(\mathsf{aggregate}, C)$ from BB and computes the correct partial decryption $\delta_i' = \mathsf{VSS.PDec}(s_i, C)$. If $(\mathsf{pdec}, i, \delta_i, \cdot) \in \mathsf{BB}$ and $\delta_i' \neq \delta_i$, trustee T_i published an incorrect partial decryption and is blamed.

Error 7: Invalid Share Distribution (err_7). Evidence of type $\mathsf{bad\text{-}share}$ contains $(i, j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell)$: the sender trustee, receiver trustee, received share, and published commitments. The judge verifies the commitments match those on BB: $(\mathsf{com\text{-}vss}, i, \{W_{i,\ell}\}_\ell) \in \mathsf{BB}$. If $\mathsf{VSS.Vf}(j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell) = 0$, the share $\hat{u}_{i,j}$ is inconsistent with T_i's commitments, and T_i is blamed for distributing an invalid share.

No Misbehavior Detected. If none of the evidence entries result in blame assignment, the judge returns $\perp$, indicating no detectable misbehavior.

5 Privacy and Verifiability Analysis of VoteSentinel

This section formally establishes the security properties of VS under standard assumptions on ML-KEM, BFV, and VSS, all of which are post-quantum.

To address the key-loss resiliency, in the IACR 2025 election failure [20,31,32], Helios's (3,3)-threshold caused irrecoverable failure when one trustee lost their key. Under VoteSentinel with $(t=2,n=3)$-threshold, when trustee T_3 loses s_3, the remaining trustees $S=\{1,2\}$ satisfy $|S|\geq t$, enabling VSS.Comb to recover sk via Lagrange interpolation: $\tilde{s}=\mu_1 s_1+\mu_2 s_2$ where $\mu_1=2,\mu_2=-1$, yielding $\tilde{s}=2s_1-s_2=$ sk, thereby completing threshold decryption through $\delta_i = \mathsf{VSS.PDec}(s_i, C)$ for $i\in S$ (cf. Tally, Section 4), guaranteeing election completion despite loss of up to $n-t=1$ trustee keys under (t,n)-VSS correctness (Definition 5).

5.1 Resistance to Coercion

The Coercion Resistance property implies many other privacy guarantees; i.e., Coercion Resistance [11] $\overset{[19]}{\Rightarrow}$ Receipt-freeness [9] $\overset{[19]}{\Rightarrow}$ Privacy with Malicious Board [8] $\overset{[8]}{\Rightarrow}$ Ballot Privacy [18] $\overset{[65]}{\Rightarrow}$ Individual Verifiability [10] $\overset{[17]}{\Leftarrow}$ Accountability [16]. Among all definitions of coercion resistance, Juels, Catalano & Jakobsson (JCJ) has been well-studied, and we consider the patched version of it [11] as follows.

Definition 9 (Coercion Resistance [11]). *Let $\Gamma=$(Setup,Register,Vote,Tally) be an election scheme, n_a, n_c and n_v be integers, and $\mathcal{D}$ be a distribution over $\{1,\ldots,n_c,\phi,\lambda\}^{n_v}$. We say Γ satisfies* JCJ *with respect to n_a, n_c, n_v, $\mathcal{D}$, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists probabilistic polynomial-time algorithm $\mathcal{B}$, algorithm* fakecred*, and a negligible function* negl*, such that for all security parameters κ, we have:*

$$|\Pr[\mathsf{JCJ}(\Gamma,\mathcal{A},n_a,n_c,n_v,\mathsf{fakecred},\mathcal{D},\kappa)=1]-\Pr[\mathsf{JCJ\text{-}\$}(\Gamma,\mathcal{B},n_a,n_c,n_v,\mathcal{D},\kappa)=1]|\leq \mathsf{negl}(\kappa)$$

where games JCJ *and* JCJ-$ *are defined as follows in Figure 2b, and algorithm* fakecred *takes a public key, a public credential, and a private credential as input, and outputs a (fake) private credential.*

The JCJ-$($\Gamma,\mathcal{B},n_a,n_c,n_v,\mathcal{D},\kappa$) *game is identical to* JCJ *[11] except:*

- *Line 13: $bb_2 \leftarrow \mathcal{B}(sd',M,|bb_1|)$ (receives bulletin board length)*
- *Line 14: $v \leftarrow$ Ideal-Tally$(sk,bb_1\cup bb_2,L,n_c,\kappa)$*
- *Line 15: $g \leftarrow \mathcal{B}(v)$ (no proof)*

where Ideal-Tally *tallies bb_1 normally and specially handles $bb_2 \setminus bb_1$ by counting ballots from corrupt voters without double-voting.*

Theorem 1. *Let H be modeled as a quantum random oracle. Assuming* $\mathsf{MLWE}_{n,q,\chi}$ *and* $\mathsf{RLWE}_{n,q,\chi}$ *problems are hard for QPT adversaries,* VS *satisfies coercion resistance (Definition 9). Specifically, there exists a QPT simulator* $\mathsf{fakecred}$ *such that for all QPT adversaries $\mathcal{A}$, there exists QPT $\mathcal{B}$ with:*

$$|\Pr[\mathsf{JCJ}(\mathsf{VS},\mathcal{A},n_a,n_c,n_v,\mathsf{fakecred},\mathcal{D},\lambda)=1]-\Pr[\mathsf{JCJ\text{-}\$}(\mathsf{VS},\mathcal{B},n_a,n_c,n_v,\mathcal{D},\lambda)=1]|\leq \mathsf{negl}(\lambda)$$

The $\mathsf{fakecred}$ algorithm generates indistinguishable fake credentials by exploiting the re-voting mechanism: a coerced voter submits a compliant ballot with counter ℓ, then later submits their true vote with counter $\ell' > \ell$. The nullifier resolution in Tally ensures only the highest-counter ballot is counted, while IND-CPA security of BFV prevents the coercer from distinguishing real from fake credentials.

5.2 Participation Privacy

Definition 10 ((δ,k_{priv})-Participation Privacy [9]). *A voting scheme $\mathcal{S}$ achieves (δ,k)-participation privacy if for any PPT adversary $\mathcal{A}$, any two honest voters id_0,id_1, and any $k' \leq k$, then we say that the following relation holds* $\left|\Pr[\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathcal{S},k}=0]-\Pr[\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathcal{S},k}=0]\right| \leq \delta$*, where the experiment* $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},k}$ *is defined as Setup phase:*

1. *Run setup to obtain public parameters and keys*
2. *Initialize two bulletin boards* $\mathsf{BB}_0,\mathsf{BB}_1 \leftarrow \emptyset$
3. *Register eligible voters including id_0,id_1*
4. *Choose* $\beta \leftarrow_R \{0,1\}$

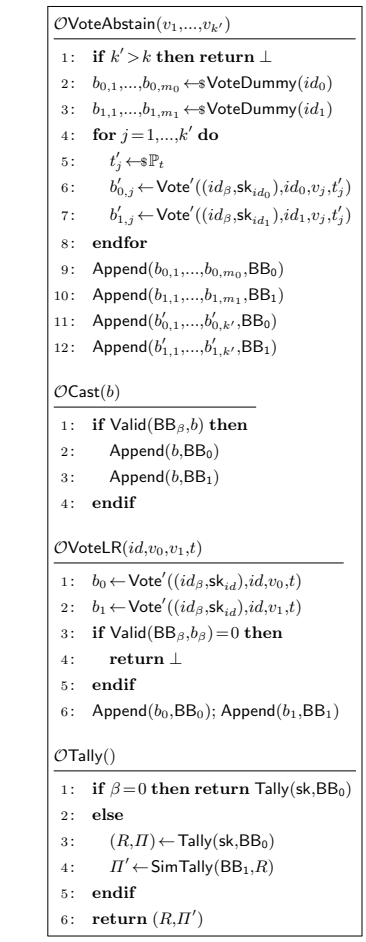
In the query phase, the adversary $\mathcal{A}$ has access to a set of the oracles $\mathcal{Q}_S = \{\mathsf{OCast},\mathsf{OVoteAbstain},\mathsf{OVoteLR},\mathsf{OTally}\}$ *and sees* BB_β*. Eventually, $\mathcal{A}$ outputs guess β'. The experiment returns $(\beta'=\beta)$. The scheme achieves (δ,k)-participation privacy if no adversary can distinguish whether voter id_0 abstained while id_1 voted up to k times, or vice versa, with advantage greater than δ.*

Theorem 2. *Let H be modeled as QROM. Assuming* $\mathsf{RLWE}_{n,q,\chi}$ *problem is hard for QPT adversaries, the voting scheme* VS *achieves (δ,L)-participation privacy (Definition 10) with $\delta \leq \mathsf{negl}(\lambda)$. For all QPT adversaries $\mathcal{A}$, any two honest voters $id_0,id_1 \in \mathcal{E}$, and $k' \leq L$:* $\left|\Pr[\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathsf{VS},L}(\lambda)=1]-\Pr[\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathsf{VS},L}(\lambda)=1]\right| \leq \mathsf{negl}(\lambda)$.

Voter identities in VS are encoded in encrypted identifier ciphertexts $c' = \mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncID}(\nu_j,\ell))$, which are IND-CPA secure under RLWE. The nullifiers $\nu_j = H(\xi_j \| j \| \mathsf{0x00})$ are computationally hidden, and the public BB reveals only ciphertexts, not participation patterns.

5.3 Verifiability

It has been shown that privacy implies individual verifiability [65]. But we do further check on VS by Theorem 3 that states VS satisfies both qualitative and quantitative verifiability under the framework of Küsters et al., demonstrating that honest voters who perform verification checks can detect any manipulation of their ballots with overwhelming probability, while the bulletin board integrity and threshold decryption mechanism ensure universal verifiability of the tallying process.



$\mathcal{O}$VoteAbstain$(v_1,\ldots,v_{k'})$

1: **if** $k' > k$ **then return** $\perp$
2: $b_{0,1},\ldots,b_{0,m_0} \leftarrow\$$ VoteDummy(id_0)
3: $b_{1,1},\ldots,b_{1,m_1} \leftarrow\$$ VoteDummy(id_1)
4: **for** $j=1,\ldots,k'$ **do**
5: $\quad t'_j \leftarrow\$ \mathbb{P}_t$
6: $\quad b'_{0,j} \leftarrow$ Vote$'((id_\beta,\mathsf{sk}_{id_0}),id_0,v_j,t'_j)$
7: $\quad b'_{1,j} \leftarrow$ Vote$'((id_\beta,\mathsf{sk}_{id_1}),id_1,v_j,t'_j)$
8: **endfor**
9: Append$(b_{0,1},\ldots,b_{0,m_0},\mathsf{BB}_0)$
10: Append$(b_{1,1},\ldots,b_{1,m_1},\mathsf{BB}_1)$
11: Append$(b'_{0,1},\ldots,b'_{0,k'},\mathsf{BB}_0)$
12: Append$(b'_{1,1},\ldots,b'_{1,k'},\mathsf{BB}_1)$

$\mathcal{O}$Cast(b)

1: **if** Valid(BB_β,b) **then**
2: $\quad$ Append(b,BB_0)
3: $\quad$ Append(b,BB_1)
4: **endif**

$\mathcal{O}$VoteLR(id,v_0,v_1,t)

1: $b_0 \leftarrow$ Vote$'((id_\beta,\mathsf{sk}_{id}),id,v_0,t)$
2: $b_1 \leftarrow$ Vote$'((id_\beta,\mathsf{sk}_{id}),id,v_1,t)$
3: **if** Valid$(\mathsf{BB}_\beta,b_\beta)=0$ **then**
4: $\quad$ **return** $\perp$
5: **endif**
6: Append(b_0,BB_0); Append(b_1,BB_1)

$\mathcal{O}$Tally$()$

1: **if** $\beta=0$ **then return** Tally$(\mathsf{sk},\mathsf{BB}_0)$
2: **else**
3: $\quad (R,\Pi) \leftarrow$ Tally$(\mathsf{sk},\mathsf{BB}_0)$
4: $\quad \Pi' \leftarrow$ SimTally(BB_1,R)
5: **endif**
6: **return** (R,Π')

(a) The Participation Privacy Oracles

Game JCJ$(\Gamma,\mathcal{A},n_a,n_c,n_v,\mathsf{fakecred},\mathcal{D},\kappa)$

1: $(pk,sk,mb,mc) \leftarrow$ Setup(κ)
2: $V \leftarrow \mathcal{A}(pk,\kappa)$
3: **for** $i \in [1,n_v]$ **do**
$\quad (pd_i,sd_i) \leftarrow$ Register(pk,κ)
4: $L \leftarrow \{pd_1,\ldots,pd_{n_v}\}$
5: $M \leftarrow \{(i,sd_i) : i \in V \wedge i \in [1,n_v]\}$
6: $(j,v) \leftarrow \mathcal{A}(M,L)$
7: **if** $|V| \neq n_a \vee j \notin [1,n_v] \setminus V \vee v \notin [1,n_c] \cup \{\phi\}$ **then**
$\quad$ **return** 0
8: $bb_1 \leftarrow \emptyset$
9: $\beta \leftarrow\$ \{0,1\}$
10: **if** $\beta=0$ **then**
$\quad$ **if** $v \neq \phi$ **then**
$\quad\quad b \leftarrow$ Vote(sd_j,pk,v,n_c,κ)
$\quad\quad bb_1 \leftarrow bb_1 \cup \{b\}$
$\quad sd' \leftarrow \mathsf{fakecred}(pk,pd_j,sd_j)$
11: **else**
$\quad sd' \leftarrow sd_j$
12: **for** $i \in [1,n_v] \setminus (\{j\} \cup V)$ **do**
$\quad v \leftarrow\$ \mathcal{D}$
$\quad$ **if** $v \neq \phi$ **then**
$\quad\quad$ **if** $v = \lambda$ **then**
$\quad\quad\quad v \leftarrow\$ [1,n_c]$
$\quad\quad\quad sd_i \leftarrow \mathsf{fakecred}(pk,pd_i,sd_i)$
$\quad\quad b \leftarrow$ Vote(sd_i,pk,v,n_c,κ)
$\quad\quad bb_1 \leftarrow bb_1 \cup \{b\}$
13: $bb_2 \leftarrow \mathcal{A}(sd',bb_1)$
14: $(v,pf) \leftarrow$ Tally$(sk,bb_1 \cup bb_2,L,n_c,\kappa)$
15: $g \leftarrow \mathcal{A}(v,pf)$
16: **return** $(\beta=g)$

(b) The JCJ Game for Coercion Resistance

Fig. 2: Security Games for Coercion Resistance and Participation Privacy

Definition 11 (Verifiability Framework [10]). *Let $\mathcal{P}$ be a protocol with the set of agents Σ. Let $\delta \in [0,1]$ be the tolerance, $J \in \Sigma$ be the judge and γ be a goal. Then, we say that the protocol $\mathcal{P}$ is (γ,δ)-verifiable by the judge J if for all adversaries π_A and $\pi = (\pi_{\mathcal{P}} \| \pi_A)$, the probability $\Pr[\pi(\ell) \rightarrow \neg\gamma,(J : accept)]$ is δ-bounded as a function of ℓ.*

The following Definition 12 describes the qualitative verifiability goal, and Definition 13 recalls the quantitative verifiability goal from [10].

Definition 12. *The qualitative goal $\gamma_{ql}(\varphi)$ is satisfied in a run r if either (a) the trust assumption φ does not hold true in r, or if (b) φ holds true in r and there exist valid choices $\tilde{c}_1,...,\tilde{c}_n$ for which the following conditions are satisfied:*

1. *An election result is published in r and it is equal to $\rho(\tilde{c}_1,...,\tilde{c}_n)$.*
2. *The multiset $\{\tilde{c}_1,...,\tilde{c}_n\}$ consists of:*
   - *all the actual choices of honest voters who successfully performed their check,*
   - *plus a subset of actual choices of honest voters who did not perform their check (successfully),*
   - *and plus at most n_d additional choices (where n_d is the number of dishonest voters).*

Definition 13. *Let d be distance function defined by $d(\mathbf{c},\mathbf{c}') = \sum_{c\in\mathcal{C}} |f_{count}(\mathbf{c})[c] - f_{count}(\mathbf{c}')[c]|$ where f_{count} counts occurrences of each choice. The quantitative goal $\gamma_{qn}(k,\varphi)$ is satisfied in a run r if either (a) the trust assumption φ does not hold true in r, or if (b) φ holds true in r and there exist valid choices $c'_1,...,c'_{n_d}$ (representing possible choices of dishonest voters) and $\tilde{c}_1,...,\tilde{c}_n$, such that:*

1. *An election result is published and it is equal to $\rho(\tilde{c}_1,...,\tilde{c}_n)$, and*
2. *$d((c_1,...,c_{n_h},c'_1,...,c'_{n_d}),(\tilde{c}_1,...,\tilde{c}_n)) \leq k$.*

where $c_1,...,c_{n_h}$ are the actual choices of honest voters.

Theorem 3. *Let φ_t denote the trust assumption that at least t trustees are honest and* BB *is append-only. The voting scheme* VS *is $(\gamma_{ql}(\varphi_t),0)$-verifiable and $(\gamma_{qn}(0,\varphi_t),\delta)$-verifiable with $\delta \leq \mathsf{negl}(\lambda)$ (Definitions 11–13). Formally, for judge $J = \mathsf{Vf}$ and all adversaries π_A:*

$$\Pr[\pi \to \neg\gamma_{ql}(\varphi_t),(J:\mathsf{accept})] = 0 \quad \wedge \quad \Pr[\pi \to \neg\gamma_{qn}(0,\varphi_t),(J:\mathsf{accept})] \leq \mathsf{negl}(\lambda)$$

Individual verifiability follows from deterministic ballot construction: voters recompute (c,c',γ) from (cred_j,v,ℓ) and verify presence on BB. Universal verifiability holds because the homomorphic aggregation $C = \sum_{c\in\mathcal{F}} c$ is publicly recomputable, partial decryptions δ_i are verified against commitments $\Gamma_i = H(s_i \| \omega_i)$, and the sum constraint $\sum_i r_i = |\mathcal{F}|$ detects any manipulation.

5.4 Everlasting Privacy

This concept is based on the principle that a voter's individual choice remains secret forever, even against an adversary with unlimited computational power in the future who could potentially break today's cryptographic assumptions. We use the definition [50], which is recommended by [12] as well.

Definition 14 (Practical Everlasting Privacy [50]). *Let the voting scheme $\Pi = (\mathsf{Setup},\mathsf{Register},\mathsf{Vote},\mathsf{Tally},\mathsf{Verify})$ with n voters $\mathcal{V} = \{v_1,...,v_n\}$, m trustees $\mathcal{T} = \{T_1,...,T_m\}$, and public bulletin board* BB.

Consider the experiment $\mathsf{Exp}^{\mathsf{priv}}_{\Pi,\mathcal{A}}(\lambda,b)$:

1. $(\mathsf{pp},\mathsf{sk}_{\mathsf{sys}},\mathsf{state}_0) \leftarrow \mathsf{Setup}(1^\lambda)$ *where* $\mathsf{sk}_{\mathsf{sys}}$ *denotes system secrets (trustee shares, decryption keys)*
2. $(\mathsf{state}_1,i_0,i_1,\mathsf{vote}_0,\mathsf{vote}_1) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{reg}},\mathcal{O}_{\mathsf{vote}}}(\mathsf{pp},\mathsf{state}_0)$ *where:*
   - $\mathcal{O}_{\mathsf{reg}}$*: Registration oracle allowing* $\mathcal{A}$ *to register honest voters*
   - $\mathcal{O}_{\mathsf{vote}}$*: Voting oracle allowing* $\mathcal{A}$ *to cast votes for corrupted voters*
   - $i_0,i_1 \in \mathcal{V}$ *are two uncorrupted voters*
   - $\mathsf{vote}_0,\mathsf{vote}_1$ *are valid votes with* $\rho(\ldots,\mathsf{vote}_0,\ldots) = \rho(\ldots,\mathsf{vote}_1,\ldots)$ *for result function* ρ
3. *Execute* Vote *for voter* i_b *with vote* vote_b*, adding the resulting ballot to* BB
4. $(\mathsf{result},\pi_{\mathsf{tally}}) \leftarrow \mathsf{Tally}(\mathsf{pp},\mathsf{BB},\mathsf{sk}_{\mathsf{sys}})$
5. $\mathsf{view}_{\mathcal{A}} \leftarrow (\mathsf{pp},\mathsf{BB},\mathsf{result},\pi_{\mathsf{tally}},\mathsf{state}_1)$
6. $b' \leftarrow \mathcal{A}_2^\infty(\mathsf{view}_{\mathcal{A}})$ *where* $\mathcal{A}_2^\infty$ *is computationally unbounded and output* b'*.*

Protocol Π *satisfies* practical everlasting privacy *if for all probabilistic polynomial-time adversaries* $\mathcal{A}_1$ *(modeling adversarial behavior during election) and all computationally unbounded adversaries* $\mathcal{A}_2$ *(modeling future adversary):*

$$|\Pr[\mathsf{Exp}^{\mathsf{priv}}_{\Pi,\mathcal{A}}(\lambda,0)=1]-\Pr[\mathsf{Exp}^{\mathsf{priv}}_{\Pi,\mathcal{A}}(\lambda,1)=1]| \leq \mathsf{negl}(\lambda)$$

Theorem 4. *Let* H *be modeled as a quantum random oracle. The voting scheme* VS *satisfies practical everlasting privacy (Definition 14). For all QPT adversaries* $\mathcal{A}_1$ *(during election) and all computationally unbounded adversaries* $\mathcal{A}_2^\infty$ *(post-election)* $\left|\Pr[\mathsf{Exp}^{\mathsf{priv}}_{\mathsf{VS},\mathcal{A}}(\lambda,0)=1]-\Pr[\mathsf{Exp}^{\mathsf{priv}}_{\mathsf{VS},\mathcal{A}}(\lambda,1)=1]\right| \leq \mathsf{negl}(\lambda)$.

The public view $\mathsf{view}_{\mathcal{A}} = (\mathsf{pp},\mathsf{BB},\mathsf{result},\{\delta_i\}_{i\in S})$ contains only the aggregate ciphertext C and its decryption, never individual vote ciphertexts. The (t,n)-threshold structure of VSS ensures that without t key shares, individual votes are information-theoretically hidden within the homomorphic sum, even against unbounded adversaries.

5.5 Dispute Resolution

Definition 15 (Dispute Resolution [15]). *Let the mini-voting scheme* $\Pi = (\mathsf{Setup},\mathsf{Register},\mathsf{Vote},\mathsf{Tally},\mathsf{Verify})$ *with authority* Auth*, voters* $\mathcal{V}$*, and bulletin board* BB*. Let* $\mathsf{Faulty}(\mathsf{Auth},b)$ *denote a publicly verifiable set of traces where authority* Auth *is considered dishonest with respect to ballot* bal*. The individual dispute resolution properties are as follows.*
- Cast ballot protection ($\mathsf{VoterC}(\mathsf{Auth})$*):*

$$\begin{aligned}\mathit{VoterC(Auth)} := \{tr \mid \mathsf{verifyC}(H,\mathsf{bal}) \in tr \Rightarrow \\ (\exists [b].\mathsf{BB}_{\mathsf{rec}}([b]) \in tr \wedge b \in [b]) \vee tr \in \mathsf{Faulty}(\mathsf{Auth},b)\}\end{aligned}$$

- Timeliness property ($\mathsf{TimelyP}(\mathsf{Auth})$*):*

$$\begin{aligned}\mathit{TimelyP(Auth)} := \{tr \mid \exists tr',tr''.tr = tr' \cdot tr'' \wedge \mathsf{Ballot}(H,b) \in tr' \\ \wedge \mathsf{End} \in tr'' \Rightarrow (\exists [b].\mathsf{BB}_{\mathsf{rec}}([b]) \in tr'' \wedge b \in [b]) \vee tr \in \mathsf{Faulty}(\mathsf{Auth},b)\}\end{aligned}$$

- *Abstaining voter protection (*$\mathsf{VoterA}(\mathsf{Auth})$*):*

$$\begin{aligned}\mathit{VoterA(Auth)} := \{tr \mid \mathsf{verifyA}(H,b_H) \in tr \\ \Rightarrow \neg(\exists [b], b.\mathsf{BB}_{\mathsf{rec}}([b]) \in tr \wedge b \in [b] \wedge \mathsf{castBy}(b) \\ = H \wedge b \notin b_H) \vee \exists b.tr \in \mathsf{Faulty}(\mathsf{Auth},b)\}\end{aligned}$$

- *Authority protection (*$\mathsf{AuthP}(\mathsf{Auth})$*):*

$$\mathsf{AuthP}(\mathsf{Auth}) := \{tr \mid \mathsf{hon}(\mathsf{Auth}) \in tr \Rightarrow \forall b.tr \notin \mathsf{Faulty}(\mathsf{Auth},b)\}$$

Protocol Π *satisfies* dispute resolution $\mathsf{DR}(\Pi,T,p_H,p_{\mathsf{Auth}},p_f)$ *in topology* T *if:*

$$\begin{aligned}\mathit{TR}(\Pi,T^{\mathsf{Auth}+\mathsf{H}+}) \subseteq p_H \cap p_{\mathsf{Auth}}, \quad \mathsf{TR}(\Pi,T^{H+}) \subseteq p_H \\ \mathit{TR}(\Pi,T^{\mathsf{Auth}+}) \subseteq p_{\mathsf{Auth}}, \quad \mathsf{TR}(\Pi,T^{\mathsf{Auth}+\mathsf{H}+}) \cap p_f \neq \emptyset\end{aligned}$$

where $T^{\mathsf{Auth}+}$ *denotes* T *with trusted authority,* T^{H+} *with trusted voter* H*, and* p_f *is a functional property.*

Definition 16 (Faulty for $\mathsf{VoteSentinel}$**).** *For* $\mathsf{VoteSentinel}$*, we define* $\mathsf{Faulty}(\mathsf{Auth},b)$ *as the set of traces where:*

1. *Evidence exists that* Auth *received ballot* b *but* b *is not in recorded ballots:*

$$\{tr \mid \exists \mathsf{nullifier}_j, h_1, h_2, h_3, [b].\mathsf{Pub}(\mathsf{Auth},\mathsf{BB},(b,h_1,h_2,h_3,\mathsf{nullifier}_j)) \in tr \wedge \mathsf{BB}_{\mathsf{rec}}([b]) \in tr \wedge b \notin [b]\}$$

2. *OR invalid nullifier resolution occurred:*
$\{tr \mid \exists \mathsf{null}_v, j, j', b, b'.\mathsf{nullifier}_j = H(\mathsf{null}_v \| j) \in tr \wedge \mathsf{nullifier}_{j'} = H(\mathsf{null}_v \| j') \in tr \wedge j > j' \wedge b \in \mathcal{B}_{final}\}$

Theorem 5. *Let* T *be the protocol topology with authority* EA*, trustees* $\{T_i\}_{i\in[n]}$*, and voters* $\mathcal{V}$*. The voting scheme* VS *satisfies dispute resolution* $\mathsf{DR}(\mathsf{VS},T,p_H,p_{\mathsf{Auth}},p_f)$ *(Definition 15) where:*

$$\begin{aligned}p_H &= \mathsf{VoterC}(\mathsf{EA}) \cap \mathsf{TimelyP}(\mathsf{EA}) \cap \mathsf{VoterA}(\mathsf{EA}) \\ p_{\mathsf{Auth}} &= \mathsf{AuthP}(\mathsf{EA}) \cap \bigcap_{i\in[n]} \mathsf{AuthP}(T_i)\end{aligned}$$

Under collision resistance of H *in QROM, for evidence types* $\mathsf{type} \in \{\mathsf{drop}, \mathsf{bad\text{-}pdec}, \mathsf{not\text{-}counted}, \mathsf{bad\text{-}share}\}$*:* $\Pr[\mathsf{Jdg}(\mathsf{pp},\mathsf{BB},E) = (P,\mathsf{err}) \wedge P \notin \mathsf{Faulty}] \leq \mathsf{negl}(\lambda)$*.*

The commitment $\gamma = H(c\|c'\|\nu_j\|\ell\|r)$ binds voters to their ballots, enabling them to prove ballot submission. The VSS commitments $\{W_{i,\ell}\}_\ell$ and key commitments $\Gamma_i = H(s_i\|\omega_i)$ enable detection of malicious trustees. The judge Jdg algorithm verifies evidence against public commitments, assigning blame only when cryptographic proofs of misbehavior exist.

5.6 Accountability

Definition 17 (Accountability [16, 17]). *Let the mini-voting scheme* $\Pi = (\mathsf{Setup}, \mathsf{Reg}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Vf}, \mathsf{Jdg})$ *satisfies accountability if for all PPT adversaries* $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{acc}}_{\mathcal{A},\Pi}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{acc}}_{\Pi,\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where the experiment* $\mathsf{Exp}^{\mathsf{acc}}_{\Pi,\mathcal{A}}(\lambda)$ *(Figure 3) returns 1 iff:*

- ***Fairness violation:*** *Judge blames honest party, i.e.,* $B \leftarrow \mathsf{Jdg}(\mathsf{pd},\mathsf{BB},E)$ *with* $B \neq \perp$ *but* $B \notin \mathsf{Bad}(\mathsf{pd},\mathsf{BB},E,V)$*, or*
- ***Completeness violation:*** *Result inconsistent with honest votes but no party blamed, i.e.,* $B = \perp$ *but* $|\mathsf{BB}_{\mathsf{vote}}| > N$ *or* $|\mathcal{F}| > N$ *or honest votes not in result.*

Definition 18 (Evidence [17]). *Evidence* E *can be seen as a set of tuples* $(\mathsf{type},\mathsf{data})$ *which enable public verification of misbehavior, where the values of type can get assign as* $\mathsf{type} \in \{\mathsf{drop},\mathsf{not\text{-}counted},\mathsf{bad\text{-}pdec},\mathsf{bad\text{-}share}\}$.

$\mathsf{Exp}^{\mathsf{acc}}_{\Pi,\mathcal{A}}(\lambda)$

1: $\mathsf{pd} \leftarrow \mathcal{A}(1^\lambda)$; $V \leftarrow \emptyset$
2: $(\mathsf{BB},\mathsf{tL}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{vote}}}(\mathsf{pd})$
3: **for** $(\beta,\sigma) \in \mathsf{tL} : \mathsf{ASVerify}(\mathsf{pd},\beta,\sigma) = 0$ **then return** 1
4: $E \leftarrow \{\mathsf{VSDVerify}(s,\mathsf{pd},\mathsf{BB}) : s \in V\} \cup \mathcal{A}(E)$
5: $B \leftarrow \mathsf{Jdg}(\mathsf{pd},\mathsf{BB},E)$; $B^* \leftarrow \mathsf{Bad}(\mathsf{pd},\mathsf{BB},E,V)$
6: **return** $(B \neq \perp \wedge B \notin B^*) \vee (B = \perp \wedge \neg \mathsf{Valid}(V,\mathsf{BB}))$

$\mathcal{O}_{\mathsf{vote}}(j,v)$

1: $(\beta,\mathsf{state}) \leftarrow \mathsf{Vote}(\mathsf{pp},j,v)$
2: $\sigma \leftarrow \mathsf{ASCreate}(\mathsf{ask},\beta)$
3: $V \leftarrow V \cup \{(\mathsf{state},\beta,\sigma)\}$
4: **return** (β,σ)

Fig. 3: Accountability experiment where $\mathsf{Valid}(V,\mathsf{BB})$ checks result consistency.

Theorem 6. *Let* H *be modeled QROM. Assuming collision resistance of* H *and binding of* Com *against QPT adversaries, the voting scheme* VS *satisfies accountability (Definition 17). For all QPT adversaries* $\mathcal{A}$*:* $\mathsf{Adv}^{\mathsf{acc}}_{\mathcal{A},\mathsf{VS}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{acc}}_{\mathsf{VS},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$ *where* $\mathsf{Exp}^{\mathsf{acc}}_{\mathsf{VS},\mathcal{A}}$ *returns* 1 *iff* (Jdg *blames honest party*) $\vee$ (Jdg *returns* $\perp$ *on inconsistent result*).

Fairness holds because Jdg only blames party P when verifiable evidence exists: commitment openings for voters, VSS verification failures for trustees, or partial decryption inconsistencies. Completeness holds because any deviation from honest executionballot dropping, incorrect nullifier resolution, or malicious partial decryptionproduces publicly verifiable evidence against the responsible party.

5.7 Ideal Functionality

Ideal voting functionality for a set of identities I and a result function $\rho : (I \times \mathbb{V})^* \rightarrow R$ can get described as in Definition 21, and this property can be achieved if we have Ballot Privacy + Strong Consistency $\wedge$ Correctness $\overset{[18]}{\Rightarrow}$ Ideal Functionality.

Definition 19 (Strong Consistency). *A voting scheme* V *is* strongly consistent *if there exist algorithms* $\mathsf{Extract}(b,\mathsf{sk})$ *returning* (upk,v) *from ballot* b*, and* $\mathsf{ValidInd}(b,\mathsf{pk})$ *checking ballot well-formedness, satisfying:*

- *For any* $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(\lambda)$ *and* $(id,v,\mathsf{usk}_\cdot)$*, if* $b \leftarrow \mathsf{Vote}(id,v,\mathsf{pk},\mathsf{usk}_\cdot)$ *then* $\mathsf{Extract}(b,\mathsf{sk}) = (\mathsf{upk},v)$ *with overwhelming probability.*
- *For any* (BB,b)*, if* $\mathsf{Valid}(\mathsf{BB},b,\mathsf{pk}) = 1$ *then* $\mathsf{ValidInd}(b,\mathsf{pk}) = 1$.
- *For any PPT* $\mathcal{A}$*:* $\Pr[\mathsf{Exp}^{\mathsf{consis}}_{\mathcal{A},\mathsf{V}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Definition 20 (Strong Correctness). *A voting scheme* $\mathcal{V}$ *is* strongly correct *if the advantage of any efficient adversary* $\mathcal{B}$*, defined by* $\mathcal{A}^{\mathsf{corr}}_{\mathcal{B},\mathcal{V},I}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{B},\mathcal{V},I}(\lambda) = 1]$ *(where* $\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{B},\mathcal{V},I}(\lambda)$ *is defined in Figure 4b) is negligible as a function of* λ*.*

$\mathsf{Exp}^{\mathsf{consis}}_{\mathcal{A},\mathcal{V},I}(\lambda)$

1: $\mathsf{uL} \leftarrow \mathsf{empty}$
2: $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\forall id \in I$ **do** $\mathsf{uL}.[id] \leftarrow \mathsf{Register}(id)$
4: $\mathsf{BB} \leftarrow \mathcal{A}(\mathsf{pk},\mathsf{uL})$
5: $e \leftarrow \forall b \in \mathsf{BB}.\,\mathsf{ValidInd}(b,\mathsf{pk})$
6: $(r,\Pi) \leftarrow \mathsf{Tally}(\mathsf{BB},\mathsf{sk})$
7: **for** i **in** $1..|\mathsf{BB}|$ **do**
8: $\quad d\mathsf{BB}[i] \leftarrow \mathsf{Extract}(\mathsf{BB}[i],\mathsf{sk})$
9: $r' \leftarrow \mathsf{Counting} \circ \mathsf{Policy}(d\mathsf{BB})$
10: **return** $(r \neq r' \wedge e)$

(a) The Strong Consistency experiment

$\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A},\mathcal{V},I}(\lambda)$

1: $\mathsf{uL} \leftarrow \mathsf{empty}$; $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2: $\forall id \in I$ **do** $\mathsf{uL}.[id] \leftarrow \mathsf{Register}(id)$
3: $(id,v,\mathsf{BB}) \leftarrow \mathcal{A}(\mathsf{pk},\mathsf{uL})$; $e \leftarrow \mathsf{false}$
4: **if** $(id \in I)$ **then**
5: $\quad (\mathsf{usk},\mathsf{upk}) \leftarrow \mathsf{uL}.[id]$
6: $\quad e_1 \leftarrow \forall x.\forall y.\ (x,y,\star,\star) \in \mathsf{BB} \implies$
7: $\quad\quad (x,y) = (id,\mathsf{upk}) \vee (x \neq id \wedge y \neq \mathsf{upk})$
8: $\quad b \leftarrow \mathsf{Vote}(id,v,\mathsf{pk},\mathsf{usk})$
9: $\quad e_2 \leftarrow \mathsf{Valid}(\mathsf{BB},b,\mathsf{pk})$; $e \leftarrow e_1 \wedge \neg e_2$
10: **return** e

(b) The Strong Correctness experiment

Fig. 4: Security experiments for Strong Consistency and Strong Correctness

Definition 21 (Ideal Functionality [18]). *The functionality* F_{voting} (ρ)*, interacting with an environment* $\mathfrak{E}$ *and an adversary* S*, proceeds as follows:*
1) On input vote (id,v) *from* $\mathfrak{E}$ *or* S*, store* (id,v) *and send ack* (id) *to* S*.*
2) On input tally from S*, return to both* $\mathfrak{E}$ *and* S *the result of the function* ρ *applied on the sequence of* (id,v) *pairs received, then halt.*

Theorem 7. *Let* H *be modeled as QROM. Assuming the* $\mathsf{RLWE}_{n,q,\chi}$ *problem is hard for QPT adversaries, the voting scheme* VS *securely realizes the ideal functionality* $\mathcal{F}_{\mathsf{voting}}(\rho)$ *where* ρ *is the counting function. There exists a QPT simulator* $\mathcal{S}$ *such that for all QPT environments* $\mathfrak{E}$*:* $\mathsf{EXEC}_{\mathsf{VS},\mathcal{A},\mathfrak{E}}(\lambda) \approx_c \mathsf{IDEAL}_{\mathcal{F}_{\mathsf{voting}}(\rho),\mathcal{S},\mathfrak{E}}(\lambda)$ *where* $\approx_c$ *denotes computational indistinguishability.*

By Theorem 1, VS achieves ballot privacy. Strong consistency (Definition 19) holds because $\mathsf{Extract}$ recovers (upk,v) from any valid ballot via threshold decryption, and $\mathsf{ValidInd}$ checks ciphertext well-formedness. Strong correctness follows from the deterministic vote encoding and homomorphic properties of BFV. The composition theorem of [18] then implies VS realizes $\mathcal{F}_{\mathsf{voting}}(\rho)$.

6 Conclusion and Future Directions

We presented VoteSentinel, a post-quantum voting scheme achieving a wide range of properties under a unified framework built on ML-KEM, BFV, and VSS. A notable design aspect is that VoteSentinel avoids heavyweight machinery: no zero-knowledge proofs are required for ballot validity, and no signature schemes are used for voter authentication. Instead, validity is enforced through constraintsthe sum $\sum_{i=0}^{k-1} r_i = |\mathcal{F}|$ for single-choice or $\sum_{i=0}^{k-1} r_i = |\mathcal{F}| \cdot k(k-1)/2$ for ranked-choice votingverified only after homomorphic aggregation. The ML-KEM-based secure channels between trustees during Setup provide post-quantum confidentiality for share distribution without persistent key infrastructure, and the structured error codes in Jdg (err_1–err_7) provide fine-grained accountability, distinguishing EA failures from trustee misbehavior. As future directions, we aim to eliminate the distributed key generation phase via silent-setup threshold cryptosystems [66]; however, current constructions rely on witness encryption, which remains impractical and quantum-vulnerable [67–69]. Designing a post-quantum silent-setup scheme supporting homomorphic operations is an open problem. Additionally, unlike other post-quantum schemes [3,6,7,24,52–56], we plan to deploy an actual production-ready implementation of VoteSentinel as an online web-based voting platform.

A Appendix

A.1 Proof of Theorem 1

Proof. We prove that VS satisfies coercion resistance by constructing a fakecred algorithm and showing computational indistinguishability through a sequence of games. On input $(\mathsf{pk}, pd_j, sd_j)$ where $sd_j = (\xi_j, \nu_j)$, the algorithm fakecred proceeds as follows:

1. Sample fresh randomness $\tilde{\xi}_j \leftarrow\!\!\$ \{0,1\}^{2\lambda}$
2. Compute fake nullifier $\tilde{\nu}_j \leftarrow H(\tilde{\xi}_j \| j \| \mathsf{0x00})$
3. Return $\widetilde{sd}_j = (\tilde{\xi}_j, \tilde{\nu}_j)$

We define a sequence of games G_0, G_1, G_2, G_3 where $G_0 = \mathsf{JCJ}$ with $\beta = 0$ and $G_3 = \mathsf{JCJ}$ with $\beta = 1$.

Game G_0: This is $\mathsf{JCJ}(\mathsf{VS}, \mathcal{A}, \ldots)$ with $\beta = 0$. The challenger:

- If $v \neq \phi$: creates ballot $b \leftarrow \mathsf{Vote}(sd_j, \mathsf{pk}, v, \ell)$ using real credential $sd_j = (\xi_j, \nu_j)$
- Computes $\widetilde{sd}_j \leftarrow \mathsf{fakecred}(\mathsf{pk}, pd_j, sd_j)$ and gives $\widetilde{sd}_j$ to $\mathcal{A}$
- $\mathcal{A}$ produces bb_2 (potentially using $\widetilde{sd}_j$)
- Tallying uses nullifier resolution on $bb_1 \cup bb_2$

Game G_1: Identical to G_0, except we modify the identifier ciphertext in the honest voter's ballot. Instead of encrypting $\mathsf{EncID}(\nu_j, \ell)$, we encrypt $\mathsf{EncID}(\tilde{\nu}_j, \ell)$ where $\tilde{\nu}_j$ is from $\widetilde{sd}_j$.

Claim. $|\Pr[G_0=1]-\Pr[G_1=1]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)$

Proof: We construct a reduction $\mathcal{R}$ to the IND-CPA security of BFV. Given an IND-CPA challenger with public key pk^*:

1. $\mathcal{R}$ sets $\mathsf{pk}=\mathsf{pk}^*$ and runs JCJ setup
2. When creating the honest voter's ballot, $\mathcal{R}$ submits the challenge messages as $(m_0,m_1)=(\mathsf{EncID}(\nu_j,\ell),\mathsf{EncID}(\tilde{\nu}_j,\ell))$
3. $\mathcal{R}$ receives challenge ciphertext c'^* and uses it as the identifier ciphertext
4. $\mathcal{R}$ simulates tallying using its knowledge of both ν_j and $\tilde{\nu}_j$
5. $\mathcal{R}$ outputs $\mathcal{A}$'s guess

If c'^* encrypts m_0, this is G_0; if c'^* encrypts m_1, this is G_1. The IND-CPA security of BFV under RLWE in QROM (cf. [48]) gives the bound. ■

Game G_2: Identical to G_1, except we also modify the vote ciphertext. Instead of encrypting the coerced vote v, we encrypt a dummy vote $v^*=1$ (or any fixed valid vote).

Claim. $|\Pr[G_1=1]-\Pr[G_2=1]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)$

Proof: Analogous reduction to IND-CPA security. The reduction submits $(m_0,m_1)=(\mathsf{EncS}(v),\mathsf{EncS}(v^*))$ as challenge messages for the vote ciphertext. Since the honest voter's ballot now uses nullifier $\tilde{\nu}_j$ (from G_1), and $\mathcal{A}$ receives $\widetilde{sd}_j$ containing $\tilde{\nu}_j$, any ballot $\mathcal{A}$ creates with $\widetilde{sd}_j$ will have the same nullifier. The nullifier resolution will count whichever has higher counter, making the content of the original ballot irrelevant to the final tally (if $\mathcal{A}$ re-votes with higher counter) or counted (if $\mathcal{A}$ does not re-vote). ■

Game G_3: This is $\mathsf{JCJ}(\mathsf{VS},\mathcal{A},...)$ with $\beta=1$. The challenger gives $\mathcal{A}$ the real credential sd_j directly (no $\mathsf{fakecred}$).

Claim. $|\Pr[G_2=1]-\Pr[G_3=1]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{H}(\lambda)$

Proof: In G_2, the adversary receives $\widetilde{sd}_j=(\tilde{\xi}_j,\tilde{\nu}_j)$ where $\tilde{\nu}_j=H(\tilde{\xi}_j\|j\|\mathtt{0x00})$. In G_3, the adversary receives $sd_j=(\xi_j,\nu_j)$ where $\nu_j=H(\xi_j\|j\|\mathtt{0x00})$. In G_2, the honest voter's ballot uses nullifier $\tilde{\nu}_j$ (from the G_1 modification). In G_3, there is no honest voter ballot with the targeted vote. The observation is that in both games:

- The credential given to $\mathcal{A}$ is a fresh, properly-formed credential
- In G_2: $\mathcal{A}$ gets $(\tilde{\xi}_j,\tilde{\nu}_j)$ and honest ballot uses $\tilde{\nu}_j$
- In G_3: $\mathcal{A}$ gets (ξ_j,ν_j) and no honest ballot is created for voter j

Since both ξ_j and $\tilde{\xi}_j$ are uniformly random in $\{0,1\}^{2\lambda}$, and H is modeled as a quantum random oracle, the values of distributions $(\tilde{\xi}_j,H(\tilde{\xi}_j\|j\|\mathtt{0x00}))$ and $(\xi_j,H(\xi_j\|j\|\mathtt{0x00}))$ are identically distributed. The QROM security of H as a PRF ensures this holds against QPT adversaries. ■

We must verify that the result function ρ applied to votes produces the same distribution in both games. In G_0 (real game with $\beta=0$), voter j's coerced vote v is counted unless $\mathcal{A}$ submits a higher-counter ballot. In G_3 (real game with $\beta=1$),

$\mathcal{A}$ controls sd_j and directly determines voter j's contribution. The JCJ-$ game's Ideal-Tally correctly accounts for this by counting $\mathcal{A}$'s ballots for corrupted voters, matching the real tally distribution. Combining the hybrid steps:

$$|\Pr[\mathsf{JCJ}(\mathsf{VS},\mathcal{A},...,\mathsf{fakecred},...)=1]-\Pr[\mathsf{JCJ\text{-}\$}(\mathsf{VS},\mathcal{B},...)=1]|$$
$$\leq 2\cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)+\mathsf{Adv}^{\mathsf{PRF}}_{H}(\lambda)\leq \mathsf{negl}(\lambda)$$

where the final inequality follows from the RLWE-based IND-CPA security of BFV and the PRF security of hash function in the QROM against QPT adversaries. □

A.2 Proof of Theorem 2

Proof. We prove $(\mathsf{negl}(\lambda),L)$-participation privacy by showing that no QPT adversary can distinguish whether voter id_0 voted (up to L times) while id_1 abstained, or vice versa. Consider the experiment $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathsf{VS},L}(\lambda)$ from Definition 10. The challenger:

1. Runs $(\mathsf{pp},\{(s_i,\omega_i)\}_{i\in\mathcal{Q}})\leftarrow\mathsf{Setup}(1^\lambda,n,t,k,N,L,\mathsf{mode})$
2. Initializes $\mathsf{BB}_0,\mathsf{BB}_1\leftarrow\emptyset$
3. Registers voters including id_0,id_1 with credentials $\mathsf{cred}_0,\mathsf{cred}_1$
4. Samples $\beta\leftarrow\$\{0,1\}$
5. Answers oracle queries, maintaining two bulletin boards
6. Returns $(\beta'=\beta)$ where β' is $\mathcal{A}$'s guess

The key oracle is $\mathcal{O}\mathsf{VoteAbstain}(v_1,...,v_{k'})$ which:

- Creates dummy ballots for both id_0 and id_1 (to hide timing)
- Creates real ballots: $b'_{0,j}$ for id_0 and $b'_{1,j}$ for id_1 with votes v_j
- Appends to BB_0: dummy ballots for id_0, real ballots for id_0
- Appends to BB_1: dummy ballots for id_1, real ballots for id_1
- Adversary sees BB_β

We define hybrids $H_0,H_1,...,H_{2k'}$ for a query with k' votes.

Hybrid H_0: This is $\mathsf{Exp}^{\mathsf{ppriv},0}$. Adversary sees BB_0 containing:

- Dummy ballots with $\nu_0=H(\xi_0\|id_0\|\mathtt{0x00})$
- Real ballots $(c_{0,j},c'_{0,j},\gamma_{0,j})$ where $c'_{0,j}=\mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncID}(\nu_0,\ell_j))$

Hybrid H_j for $j\in[1,k']$: We replace the j-th identifier ciphertext. Define:

$$c'_{0,j}\rightarrow c'_{*,j}=\mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncID}(\nu_1,\ell_j))$$

i.e., encrypt id_1's nullifier instead of id_0's.

Claim. For each $j\in[1,k']$: $|\Pr[H_{j-1}=1]-\Pr[H_j=1]|\leq\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)$

Proof: Standard reduction to IND-CPA. The reduction $\mathcal{R}$:

1. Receives pk^* from IND-CPA challenger

2. Sets $\mathsf{pk}=\mathsf{pk}^*$ in VS (simulating setup by programming trustees' shares)
3. For the j-th ballot, submits $(m_0,m_1)=(\mathsf{EncID}(\nu_0,\ell_j),\mathsf{EncID}(\nu_1,\ell_j))$
4. Uses challenge ciphertext c'^* as $c'_{*,j}$
5. Simulates $\mathcal{O}\mathsf{Tally}$ using $\mathsf{SimTally}$ (which only needs the result, not individual votes)

The IND-CPA advantage bounds the distinguishing probability. ■

Hybrid $H_{k'+j}$ for $j\in[1,k']$: We additionally replace the j-th vote ciphertext:

$$c_{0,j}\to c_{*,j}=\mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncS}(v_j);\rho'_1)$$

where $\rho'_1=H(\xi_1\|\ell_j\|\texttt{0x01})$ (using id_1's credential randomness).

Claim. For each $j\in[1,k']$: $|\Pr[H_{k'+j-1}=1]-\Pr[H_{k'+j}=1]|\le\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)$

Proof: The vote content is the same (v_j), only the encryption randomness differs. However, $\rho_1=H(\xi_0\|\ell_j\|\texttt{0x01})$ and $\rho'_1=H(\xi_1\|\ell_j\|\texttt{0x01})$ are both pseudorandom under QROM (since ξ_0,ξ_1 are secret).

We reduce to IND-CPA: submit $(m_0,m_1)=(\mathsf{EncS}(v_j),\mathsf{EncS}(v_j))$ (same message). The challenge ciphertext uses fresh randomness from the challenger. We then argue that using ρ_1 vs. ρ'_1 vs. fresh randomness is indistinguishable:

$$\mathsf{BFV.Enc}(\mathsf{pk},m;\rho_1)\approx_c\mathsf{BFV.Enc}(\mathsf{pk},m;r)\approx_c\mathsf{BFV.Enc}(\mathsf{pk},m;\rho'_1)$$

where $r\leftarrow\!\$\,\{0,1\}^*$ and the indistinguishability follows from PRF security of H in QROM. ■

Hybrid $H_{2k'}$: After all replacements, BB_0 contains ballots that are identically distributed to what BB_1 would contain (ballots for id_1). This equals $\mathsf{Exp}^{\mathsf{ppriv},1}$. The $\mathsf{VoteDummy}$ procedure creates ballots with encrypted identifiers. By analogous IND-CPA arguments, dummy ballots for id_0 are indistinguishable from dummy ballots for id_1. The oracle $\mathcal{O}\mathsf{Tally}$ must produce consistent results. When $\beta=0$, the real tally uses BB_0. The simulator $\mathsf{SimTally}(\mathsf{BB}_1,R)$ must produce a proof Π' for result R computed from BB_0. Since the result function ρ only depends on vote contents (not identities), and we maintain $\rho(\mathsf{BB}_0)=\rho(\mathsf{BB}_1)$ by construction (same votes, different voter labels), the simulation is straightforward: $\mathsf{SimTally}$ produces partial decryptions consistent with the published commitments $\{\Gamma_i\}_{i\in\mathcal{Q}}$ for the result R.

$$\begin{aligned}&\left|\Pr[\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathsf{VS},L}=1]-\Pr[\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathsf{VS},L}=1]\right|\\&\le 2k'\cdot\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)+2k'\cdot\mathsf{Adv}^{\mathsf{PRF}}_{H}(\lambda)\\&\le 4L\cdot(\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)+\mathsf{Adv}^{\mathsf{PRF}}_{H}(\lambda))\le\mathsf{negl}(\lambda)\end{aligned}$$

since $k'\le L$ and both advantages are negligible under RLWE in QROM. □

A.3 Proof of Theorem 3

Proof. We prove both qualitative verifiability (Definition 12) with $\delta = 0$ and quantitative verifiability (Definition 13) with $\delta = \mathsf{negl}(\lambda)$. Let φ_t denote: (i) at least t trustees in $\mathcal{Q}$ are honest, (ii) BB is append-only and publicly readable, and (iii) the hash function H is collision-resistant.

Lemma 1. *If honest voter V_j with credential $\mathsf{cred}_j = (\xi_j, \nu_j)$ votes v at counter ℓ and $\mathsf{Vf}(\mathsf{pp},\mathsf{BB},\mathsf{ind},(\mathsf{cred}_j,v,\ell,r)) = 1$, then:*

(a) The ballot (c,c',γ) encoding vote v is on BB
(b) The ciphertext c is in the final set $\mathcal{F}$ (will be tallied)

Proof: The verification algorithm Vf with $\mathsf{mode} = \mathsf{ind}$ performs:

1. Recomputes $\rho_1 = H(\xi_j \| \ell \| \mathtt{0x01})$, $\rho_2 = H(\xi_j \| \ell \| \mathtt{0x02})$
2. Recomputes $c = \mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncS}(v);\rho_1)$ (deterministic given ρ_1)
3. Recomputes $c' = \mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncID}(\nu_j,\ell);\rho_2)$
4. Recomputes $\gamma = H(c \| c' \| \nu_j \| \ell \| r)$
5. Checks $(\mathsf{ballot},(c,c',\gamma)) \in \mathsf{BB}$ and $c \in \mathcal{F}$

Since $\mathsf{BFV.Enc}$ is deterministic given randomness, and H is deterministic, the recomputed (c, c', γ) exactly matches what Vote produced. The check actually $(\mathsf{ballot},(c,c',\gamma)) \in \mathsf{BB}$ confirms recording; $c \in \mathcal{F}$ confirms the ballot survived nullifier resolution. For (b), if $c \in \mathcal{F}$, then by construction of Tally, the ballot with nullifier ν_j and counter ℓ had the highest counter among all ballots with nullifier ν_j. The voter knows their own ℓ and can verify no higher-counter ballot exists for their nullifier. ■

Lemma 2. *If the sub-procedure operates as $\mathsf{Vf}(\mathsf{pp},\mathsf{BB},\mathsf{univ},(\mathsf{res},C,\mathcal{F},\{\delta_i\}_{i\in S})) = 1$ and φ_t holds, then $\mathsf{res} = \rho(\{v : c_v \in \mathcal{F}\})$ where c_v encrypts vote v.*

Proof: The verification algorithm Vf with $\mathsf{mode} = \mathsf{univ}$ checks:

1. $|\mathcal{F}| \leq N$ (no ballot stuffing beyond voter count)
2. $C = \sum_{c\in\mathcal{F}} c$ via $\mathsf{BFV.Add}$ (correct aggregation)
3. For each $i \in S$: $\pi_i = H(\delta_i \| \Gamma_i)$ where $(\mathsf{pdec},i,\delta_i,\pi_i) \in \mathsf{BB}$
4. Sum constraint: $\sum_{j=0}^{k-1} r_j = |\mathcal{F}|$ (single-choice) or $= |\mathcal{F}| \cdot k(k-1)/2$ (ranked)

The homomorphic property of BFV ensures:

$$\mathsf{BFV.Dec}(\mathsf{sk},C) = \mathsf{BFV.Dec}\left(\mathsf{sk}, \sum_{c\in\mathcal{F}} c\right) = \sum_{c\in\mathcal{F}} \mathsf{BFV.Dec}(\mathsf{sk},c) = \sum_{c\in\mathcal{F}} m_c$$

where m_c is the plaintext vote in ciphertext c. Each $\delta_i = C_1 \cdot s_i$ is verified against commitment $\Gamma_i = H(s_i \| \omega_i)$. Under φ_t, at least t honest trustees produce correct δ_i. The verification $\pi_i = H(\delta_i \| \Gamma_i)$ ensures:

– If trustee T_i is honest: $\delta_i = C_1 \cdot s_i$ is correct by protocol

– If trustee T_i is malicious but π_i verifies: by collision resistance of H, T_i cannot produce valid π_i for incorrect δ_i without knowing s_i and ω_i satisfying $\Gamma_i = H(s_i \| \omega_i)$

$\mathsf{VSS.Comb}$ computes $\tilde{s} = \sum_{i \in S} \mu_i \cdot s_i \quad \text{where} \quad \mu_i = \prod_{j \in S \setminus \{i\}} \frac{j}{j-i}$. By properties of Shamir secret sharing, $\tilde{s} = \mathsf{sk}$ when $|S| \geq t$. For single-choice voting with one-hot encoding $\mathsf{EncS}(v) = X^{v-1}$: $\sum_{c \in \mathcal{F}} m_c = \sum_{c \in \mathcal{F}} X^{v_c - 1} = \sum_{j=0}^{k-1} r_j \cdot X^j$, where r_j counts votes for candidate $j+1$. The constraint $\sum_j r_j = |\mathcal{F}|$ ensures each ballot contributes exactly one vote, detecting any manipulation that changes vote counts without changing the total. ■

For qualitative goal $\gamma_{ql}(\varphi_t)$, assume φ_t holds and judge accepts. We verify conditions of Definition 12:

1. *Result published equals $\rho(\tilde{c}_1, \ldots, \tilde{c}_n)$:* By Lemma 2, the published result correctly aggregates votes in $\mathcal{F}$.
2. *Multiset composition:*
   – Honest voters who verify: By Lemma 1, their votes are in $\mathcal{F}$
   – Honest voters who don't verify: Their ballots may or may not be in $\mathcal{F}$ (subset condition satisfied)
   – Dishonest voters: At most n_d additional votes (bounded by $|\mathcal{F}| \leq N$ and registration)

The probability of violating $\gamma_{ql}(\varphi_t)$ while the judge accepts is 0, as any violation requires either:

– Breaking collision resistance of H (to forge commitments)
– Corrupting $\geq t$ trustees (violates φ_t)
– Modifying append-only BB (violates φ_t)

The quantitative goal with $k = 0$ requires $d(\mathbf{c}, \tilde{\mathbf{c}}) = 0$, i.e., perfect correspondence between honest votes and tallied votes.

Claim. Under φ_t: $\Pr[\pi \to \neg\gamma_{qn}(0, \varphi_t), (J : \mathsf{accept})] \leq \mathsf{negl}(\lambda)$

Proof: Suppose $\gamma_{qn}(0, \varphi_t)$ is violated, meaning some honest vote c_i is modified to $\tilde{c}_i \neq c_i$ in the tally. This requires one of:

1. *Ballot modification on BB:* Violates append-only property in φ_t.
2. *Incorrect nullifier resolution:* The adversary must cause a ballot with nullifier ν_j and counter ℓ to be excluded despite having the highest counter. This requires forging identifier encryption, breaking IND-CPA of BFV under RLWE.
3. *Incorrect aggregation:* Universal verification recomputes $C = \sum_{c \in \mathcal{F}} c$, detecting any discrepancy with probability 1.
4. *Incorrect decryption:* Requires $\geq t$ malicious trustees (violates φ_t) or forging partial decryption proofs (breaking collision resistance of H).

Each attack vector either violates φ_t or requires breaking a cryptographic assumption, giving negligible probability. ■

Combining all parts and bounds, we will have $\Pr[\pi \to \neg\gamma_{ql}(\varphi_t), (J : \mathsf{accept})] = 0$, and $\Pr[\pi \to \neg\gamma_{qn}(0, \varphi_t), (J : \mathsf{accept})] \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda) + \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) \leq \mathsf{negl}(\lambda)$. □

A.4 Proof of Theorem 4

Proof. We prove that VS satisfies practical everlasting privacy against a two-stage adversary $(\mathcal{A}_1,\mathcal{A}_2^\infty)$ where $\mathcal{A}_1$ is QPT (during election) and $\mathcal{A}_2^\infty$ is computationally unbounded (post-election). In $\mathsf{Exp}^{\mathsf{priv}}_{\mathsf{VS},\mathcal{A}}(\lambda,b)$:

1. Setup produces $(\mathsf{pp},\{(s_i,\omega_i)\}_{i\in\mathcal{Q}})$
2. $\mathcal{A}_1^{\mathcal{O}_{\mathsf{reg}},\mathcal{O}_{\mathsf{vote}}}$ outputs (i_0,i_1,v_0,v_1) with $\rho(...,v_0,...)=\rho(...,v_1,...)$
3. Challenger creates ballot for voter i_b with vote v_b
4. Tally is executed, producing $(\mathsf{result},\{\delta_i\}_{i\in S})$
5. $\mathcal{A}_2^\infty$ receives $\mathsf{view}=(\mathsf{pp},\mathsf{BB},\mathsf{result},\{\delta_i\}_{i\in S})$ and outputs guess b'

The view view contains:

- pp: Public parameters including $\mathsf{pk}=(a,b)$, commitments $\{\Gamma_i\}$
- BB: All ballots (c,c',γ) including the challenge ballot
- result: The election outcome $(r_0,...,r_{k-1})$
- $\{\delta_i\}_{i\in S}$: Partial decryptions of the *aggregate* C

Crucially, individual vote ciphertexts are *never decrypted*. Only the aggregate $C=\sum_{c\in\mathcal{F}}c$ is decrypted.

Lemma 3. *Let $\mathcal{F}=\{c_1,...,c_m\}$ where $c_j=\mathsf{BFV.Enc}(\mathsf{pk},m_j)$ and $m=\sum_j m_j$. Given only m (the aggregate plaintext), the individual plaintexts $\{m_j\}$ are information-theoretically hidden when $|\mathcal{F}|\geq 2$.*

Proof: Consider challenge ballot $c^*\in\mathcal{F}$ encrypting either v_0 or v_1 with $\mathsf{EncS}(v_0)=X^{v_0-1}$ or $\mathsf{EncS}(v_1)=X^{v_1-1}$.The aggregate plaintext is $M=\sum_{c\in\mathcal{F}}m_c=\mathsf{EncS}(v_b)+\sum_{c\in\mathcal{F}\setminus\{c^*\}}m_c$. Since $\rho(...,v_0,...)=\rho(...,v_1,...)$ by experiment constraint, we have: $\sum_{j=0}^{k-1}r_j^{(0)}=\sum_{j=0}^{k-1}r_j^{(1)}$, where $r_j^{(b)}$ is the count for candidate j when challenge vote is v_b. For the aggregate to be consistent, we need $\mathsf{EncS}(v_0)+\sum_{c\neq c^*}m_c=\mathsf{EncS}(v_1)+\sum_{c\neq c^*}m'_c$, for some configuration of other votes. The result result is identical in both cases (by the ρ constraint), so $\mathcal{A}_2^\infty$ sees the same M regardless of b, i.e., define the set of valid vote configurations:

$$\mathcal{V}_b=\{(m_1,...,m_m):m_1=\mathsf{EncS}(v_b),\sum_j m_j=M,\forall j:m_j\in\mathsf{Im}(\mathsf{EncS})\}/$$

By the constraint $\rho(...,v_0,...)=\rho(...,v_1,...)$ and $|\mathcal{F}|\geq 2$, both $|\mathcal{V}_0|\geq 1$ and $|\mathcal{V}_1|\geq 1$. The adversary cannot determine which configuration produced M. ■

Lemma 4. *The ciphertexts $\{c_j\}_{j\in\mathcal{F}}$ on BB provide no additional information about individual votes to $\mathcal{A}_2^\infty$ beyond what is revealed by M.*

Proof: Each ciphertext $c=(c_0,c_1)\in R_q^2$ satisfies: $c_0=a\cdot u+e_0+\Delta\cdot m$, $c_1=b\cdot u+e_1$, where $u,e_0,e_1\leftarrow\chi_{\mathsf{err}}$ and $\Delta=\lfloor q/p\rfloor$. For $\mathcal{A}_2^\infty$ to extract m from c, it needs to compute:

$$m=\lfloor p(c_0+c_1\cdot s)/q\rceil \mod p$$

where $s=\sum_{i\in\mathcal{Q}}s_i$ is the secret key.

Case 1: If $\mathcal{A}_2^\infty$ can recover s, it can decrypt all ciphertexts. However, s is (t,n)-secret shared via VSS. The view contains only:

- Commitments $\{W_{i,\ell}\}_{\ell\in[0,t-1]}$ for each trustee i (hash-based, reveal nothing about shares)
- Public key contributions $b_i = -a\cdot s_i + \varepsilon_i$
- Key commitments $\Gamma_i = H(s_i \| \omega_i)$
- Partial decryptions $\delta_i = C_1 \cdot s_i$ for the aggregate only

From $\{b_i\}$, recovering $\{s_i\}$ requires solving RLWE instances. For unbounded $\mathcal{A}_2^\infty$, this is possible. However, having $\{s_i\}$ only allows decrypting ciphertextswhich brings us back to the aggregate hiding argument.

Case 2: Suppose $\mathcal{A}_2^\infty$ decrypts all individual ciphertexts. It obtains $\{m_j\}_{j\in\mathcal{F}}$. To identify the challenge ballot, $\mathcal{A}_2^\infty$ must determine which $m_j \in \{\mathsf{EncS}(v_0),\mathsf{EncS}(v_1)\}$ corresponds to the challenge voter i_b. The ballot structure (c,c',γ) includes identifier ciphertext $c' = \mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncID}(\nu_j,\ell))$. Decrypting c' reveals ν_j. However, $\mathcal{A}_2^\infty$ does not know the mapping $\nu_j \mapsto i_j$ because:

- $\nu_j = H(\xi_j \| j \| \mathtt{0x00})$ where ξ_j is secret credential randomness
- $\mathcal{A}_1$ (during election) may have queried oracles but $\mathcal{A}_2^\infty$ only receives view
- The registration oracle does not reveal credentials for honest uncorrupted voters i_0,i_1

Thus, even with unbounded computation, $\mathcal{A}_2^\infty$ cannot link decrypted votes to specific voters. ■

The QPT adversary $\mathcal{A}_1$ operates during the election and must be handled computationally.

Claim. For QPT $\mathcal{A}_1$: The challenge ballot (c^*,c'^*,γ^*) is computationally indistinguishable from an encryption of any valid vote.

Proof: By IND-CPA security of BFV under RLWE in QROM:

$$\mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncS}(v_0)) \approx_c \mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncS}(v_1))$$

The state state_1 output by $\mathcal{A}_1$ cannot encode information distinguishing v_0 from v_1 except with negligible probability. ■

Combining the computational security against $\mathcal{A}_1$ and information-theoretic security against $\mathcal{A}_2^\infty$:

$$\begin{aligned}
&\left|\Pr[\mathsf{Exp}^{\mathsf{priv}}_{\mathsf{VS},\mathcal{A}}(\lambda,0)=1] - \Pr[\mathsf{Exp}^{\mathsf{priv}}_{\mathsf{VS},\mathcal{A}}(\lambda,1)=1]\right| \\
&\leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda) + 0 \quad \text{(information-theoretic hiding in aggregate)} \\
&\leq \mathsf{negl}(\lambda)
\end{aligned}$$

The zero term reflects that once $\mathcal{A}_1$'s computational advantage is bounded, $\mathcal{A}_2^\infty$ gains no additional advantage from the everlasting view due to Lemmas 3 and 4. □

A.5 Proof of Theorem 5

Proof. We prove that VS satisfies the dispute resolution properties from Definition 15 by analyzing each evidence type handled by Jdg. Let T denote the topology with:

- Authority EA: Manages registration, maintains BB
- Trustees $\{T_i\}_{i\in[n]}$: Hold key shares, perform partial decryptions
- Voters $\mathcal{V}=\{V_1,...,V_N\}$: Submit ballots

Lemma 5. *If honest voter H successfully executes* $\mathsf{verifyC}(H,\mathsf{bal})$*, then either* $\mathsf{bal}\in\mathsf{BB}$ *or the trace is in* $\mathsf{Faulty}(\mathsf{EA},\mathsf{bal})$*.*

Proof: The verification verifyC for voter $H=V_j$ with ballot $\mathsf{bal}=(c,c',\gamma)$ checks:

1. Voter recomputes (c,c',γ) from $(\mathsf{cred}_j,v,\ell,r)$
2. Voter checks $(\mathsf{ballot},(c,c',\gamma))\in\mathsf{BB}$

If check (2) fails, butthe voter has a valid commitment opening:

- Voter possesses $(c,c',\gamma,\nu_j,\ell,r)$ such that $\gamma=H(c\|c'\|\nu_j\|\ell\|r)$
- Voter can prove they computed a valid ballot that should be on BB

The evidence $E=(\mathsf{drop},(j,c,c',\gamma,\nu_j,\ell,r))$ triggers Jdg lines 7-11:

1. Jdg verifies $\mathsf{Com.Opn}(\gamma,c\|c'\|\nu_j\|\ell,r)=1$
2. Jdg checks $(\mathsf{ballot},(c,c',\gamma))\notin\mathsf{BB}$
3. If both hold: Jdg returns $(\mathsf{EA},\mathsf{err}_4,j)$

This places the trace in $\mathsf{Faulty}(\mathsf{EA},\mathsf{bal})$ as required. ■

Lemma 6. *If honest voter H submits* $\mathsf{Ballot}(H,b)$ *before* End*, then either $b\in\mathsf{BB}$ in the final recording or the trace is in* $\mathsf{Faulty}(\mathsf{EA},b)$*.*

Proof: The argument is identical to $\mathsf{VoterC}(\mathsf{EA})$. If ballot b was submitted (witnessed by network layer) but not recorded, the voter produces evidence (drop,...) and Jdg blames EA. Timeliness is guaranteed because:

- The Vote protocol outputs (β,r) where r is the commitment randomness
- Voter retains r as receipt of intended ballot
- If EA fails to record before End, evidence remains valid

■

Lemma 7. *If honest voter H verifies abstention via* $\mathsf{verifyA}(H,b_H)$ *where b_H is their set of submitted ballots, then no ballot $b\notin b_H$ attributed to H appears in* BB*, or the trace is faulty.*

Proof: Suppose ballot $b=(c,c',\gamma)$ appears on BB with $\mathsf{castBy}(b)=H$ but $b\notin b_H$. For b to be attributed to $H=V_j$, the identifier ciphertext c' must decrypt to $\mathsf{EncID}(\nu_j,\ell)$ for some ℓ, where $\nu_j=H(\xi_j\|j\|\mathtt{0x00})$. However, ξ_j is known only to V_j (from registration). For an adversary to create b with correct ν_j:

1. *Forge ν_j:* Requires finding ξ' such that $H(\xi' \| j \| \texttt{0x00}) = \nu_j$. By collision resistance of H in QROM, this succeeds with probability $\leq \mathsf{negl}(\lambda)$.
2. *Steal ξ_j:* Requires compromising voter's credential storage (outside protocol scope).

If neither attack succeeds, any ballot attributed to H must have been created by H, hence $b \in b_H$. If attack (1) succeeds (negligible probability), the resulting trace is in Faulty by definition (cryptographic assumption violated). ■

Lemma 8. *If* EA *is honest (denoted* $\mathsf{hon}(\mathsf{EA}) \in tr$*), then for all ballots* b*:* $tr \notin \mathsf{Faulty}(\mathsf{EA}, b)$.

Proof: We show honest EA cannot be blamed by analyzing each error type:

err_1 (Invalid pk): Honest EA runs Setup correctly, producing $\mathsf{pk} \in R_q^2$. Check at line 1 passes.

err_2 (Too many ballots): Honest EA only appends valid ballots from registered voters. With N voters and L max ballots each, $|\mathcal{B}| \leq N \cdot L$. Check at line 3 passes.

err_4 (Dropped ballot): Honest EA appends every received valid ballot to BB. If voter claims $(\mathsf{drop}, ...)$ with valid commitment opening, honest EA would have recorded it. The evidence is false (voter lying), and by binding of Com:

$$\Pr[\mathsf{Com.Opn}(\gamma, m, r) = 1 \wedge \mathsf{Com.Opn}(\gamma, m', r') = 1 \wedge m \neq m'] \leq \mathsf{negl}(\lambda)$$

Thus, valid evidence against honest EA exists with negligible probability. ■

Lemma 9. *If trustee* T_i *is honest, then* $tr \notin \mathsf{Faulty}(T_i, \cdot)$.

Proof: Trustees can be blamed via err_6 (bad partial decryption) or err_7 (bad VSS share).

err_6: Evidence is $(i, \delta_i, s_i, \omega_i)$. Check requires:

- $H(s_i \| \omega_i) = \Gamma_i$ (committed key matches)
- $\delta_i \neq \mathsf{VSS.PDec}(s_i, C)$ (published partial decryption wrong)

Honest T_i publishes $\delta_i = C_1 \cdot s_i$ correctly. For evidence to blame honest T_i:

- Adversary needs (s_i, ω_i) satisfying $H(s_i \| \omega_i) = \Gamma_i$
- But T_i chose ω_i randomly; finding another (s_i', ω_i') with same hash requires breaking collision resistance of H

err_7: Evidence is $(i, j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell)$. Check requires:

- $(\mathsf{com\text{-}vss}, i, \{W_{i,\ell}\}_\ell) \in \mathsf{BB}$ (commitments published)
- $\mathsf{VSS.Vf}(j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell) = 0$ (share invalid)

Honest T_i runs $\mathsf{VSS.Shr}$ correctly, producing consistent shares and commitments. By correctness of VSS, all shares verify. Adversary cannot produce invalid share with valid-looking commitments without breaking the binding property of hash-based commitments. ■

Lemma 10. *If* $|\mathcal{F}| > N$, *then* Jdg *returns* (Trustees,err_3).

Proof: The final set $\mathcal{F}$ is computed by trustees during Tally via nullifier resolution. If $|\mathcal{F}| > N$, either:

- More than N distinct nullifiers exist (impossible if registration is correct)
- Trustees incorrectly resolved nullifiers (malicious behavior)

Since registration is managed by EA and checked at err_2, exceeding N final ballots indicates trustee malfeasance during tally. Line 5 catches this. ■

Lemma 11. *If honest voter's ballot is not counted despite having highest counter, evidence* (not-counted,...) *blames trustees.*

Proof: Evidence (j,c,c',ν_j,ℓ) triggers lines 12-16:

- Check: ballot $(c,c',\cdot) \in \mathsf{BB}$ but $c \notin \mathcal{F}$
- Check: no higher-counter ballot exists for ν_j

If both checks pass, trustees incorrectly excluded a valid highest-counter ballot. This is detectable because:

- Voter knows their ν_j and ℓ
- Voter can scan BB for any ballot with same ν_j and higher ℓ
- If none exists, exclusion was erroneous

Jdg returns (Trustees,err_5,j). ■
We verify the four conditions:

1. $\mathsf{TR}(\mathsf{VS}, T^{\mathsf{EA}+\mathsf{H}+}) \subseteq p_H \cap p_{\mathsf{Auth}}$: When both EA and voter H are honest, all protections hold (proven above), and no honest party is blamed.
2. $\mathsf{TR}(\mathsf{VS}, T^{H+}) \subseteq p_H$: When voter H is honest, voter protections (VoterC, TimelyP, VoterA) hold regardless of EA honesty, with blame correctly assigned if violated.
3. $\mathsf{TR}(\mathsf{VS}, T^{\mathsf{EA}+}) \subseteq p_{\mathsf{Auth}}$: When EA is honest, authority protection holds; EA is not blamed.
4. $\mathsf{TR}(\mathsf{VS}, T^{\mathsf{EA}+\mathsf{H}+}) \cap p_f \neq \emptyset$: The functional property p_f (correct tallying) is achieved when all parties are honest, as verified by universal verifiability (Theorem 3).

Eventually, we have the final Bound as:

$$\Pr[\mathsf{Jdg}(\mathsf{pp},\mathsf{BB},E) = (P,\mathsf{err}) \wedge P \text{ is honest}] \leq \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{Adv}^{\mathsf{bind}}_{\mathsf{Com}}(\lambda) \leq \mathsf{negl}(\lambda)$$

under collision resistance of H and binding of Com in QROM. □

A.6 Proof of Theorem 6

Proof. We prove accountability by showing both fairness (honest parties not blamed) and completeness (misbehavior detected) hold except with negligible probability. In $\mathsf{Exp}^{\mathsf{acc}}_{\mathsf{VS},\mathcal{A}}(\lambda)$ (Figure 3):

1. Adversary $\mathcal{A}$ produces public data pd
2. Adversary with $\mathcal{O}_{\mathsf{vote}}$ access produces (BB,tL) where tL is trustee list
3. Honest votes V are recorded via oracle
4. Evidence E is collected from honest voters and adversary
5. Jdg evaluates; $B^* = \mathsf{Bad}(\mathsf{pd},\mathsf{BB},E,V)$ is the set of provably misbehaving parties
6. Experiment returns 1 iff fairness or completeness violated

Lemma 12. *For all QPT $\mathcal{A}$:* $\Pr[\mathsf{Jdg}$ *returns* (B,err) *with* $B \notin B^*] \leq \mathsf{negl}(\lambda)$

Proof: We analyze each error type and show honest parties cannot be falsely blamed.

err_1: Invalid public key. Jdg checks $\mathsf{pk} \in R_q^2$. This is a syntactic check independent of party behavior. If pk is malformed, EA is blamed. An honest EA produces valid pk by construction. No false blame possible.

err_2: Excessive ballots. Jdg checks $|\mathcal{B}| \leq N \cdot L$. Honest EA only records ballots from N registered voters with at most L each. Adversary might flood BB (if EA is corrupted), correctly blaming EA. No false blame possible.

err_3: Final set too large. Jdg checks $|\mathcal{F}| \leq N$. If violated, trustees are blamed. Honest trustees correctly perform nullifier resolution, yielding $|\mathcal{F}| \leq N$. Violation requires majority trustee corruption (correctly blamed) or adversarial manipulation of BB (blamed on EA via err_2).

err_4: Dropped ballot. Evidence: $(j,c,c',\gamma,\nu_j,\ell,r)$ with $\mathsf{Com.Opn}(\gamma,c\|c'\|\nu_j\|\ell,r) = 1$ but $(\mathsf{ballot},(c,c',\gamma)) \notin \mathsf{BB}$.

For honest EA to be blamed, adversary must produce valid commitment opening for a ballot EA never received. This requires:

- Forging a commitment: Find (m,r) with $H(m\|r) = \gamma$ for some γ on BB. By collision resistance of H, probability $\leq \mathsf{negl}(\lambda)$.
- Or claiming ballot not on BB when it is: Contradicts the check.

err_5: Ballot not counted. Evidence: (j,c,c',ν_j,ℓ) with ballot on BB, $c \notin \mathcal{F}$, and no higher-counter ballot for ν_j. For honest trustees to be blamed, they must have incorrectly excluded the highest-counter ballot. Honest trustees:

1. Decrypt all identifier ciphertexts correctly (by VSS correctness)
2. Build map $\mathcal{M}$ keeping highest counter per nullifier
3. Include all max-counter ballots in $\mathcal{F}$

Incorrect exclusion requires decryption error (negligible under RLWE) or map manipulation (requires corrupted majority). No false blame against honest trustees.

err_6: Bad partial decryption. Evidence: $(i,\delta_i,s_i,\omega_i)$ with $H(s_i\|\omega_i) = \Gamma_i$ and $\delta_i \neq \mathsf{VSS.PDec}(s_i,C)$. For honest T_i to be blamed:

- Adversary needs (s_i,ω_i) satisfying commitment Γ_i
- T_i chose $\omega_i \leftarrow\$ \{0,1\}^\lambda$ uniformly
- Finding $(s_i',\omega_i') \neq (s_i,\omega_i)$ with $H(s_i'\|\omega_i') = \Gamma_i$ breaks collision resistance

Probability of false blame: $\leq \mathsf{Adv}_H^{\mathsf{CR}}(\lambda)$.

$\underline{\mathsf{err}_7\text{: Bad VSS share.}}$ Evidence: $(i, j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell)$ with commitments on BB and $\underline{\mathsf{VSS.Vf}(j, \hat{u}_{i,j}, \{W_{i,\ell}\}_\ell) = 0}$. Honest T_i produces shares consistent with commitments by VSS correctness. The adversary cannot:

- Produce invalid share with matching commitments (violates VSS soundness)
- Modify commitments on BB (append-only)

Probability of false blame will be $\leq \mathsf{Adv}_{\mathsf{VSS}}^{\mathsf{sound}}(\lambda)$, and combining all cases gives:

$$\Pr[\text{Fairness violated}] \leq \mathsf{Adv}_H^{\mathsf{CR}}(\lambda) + \mathsf{Adv}_{\mathsf{Com}}^{\mathsf{bind}}(\lambda) + \mathsf{Adv}_{\mathsf{VSS}}^{\mathsf{sound}}(\lambda) \leq \mathsf{negl}(\lambda)$$

■

Lemma 13. *For all QPT $\mathcal{A}$:* $\Pr[\mathsf{Jdg}$ *returns* $\perp$ *but* $\neg\mathsf{Valid}(V,\mathsf{BB})] \leq \mathsf{negl}(\lambda)$

Proof: We must show that any result inconsistency produces evidence blaming some party.

$\mathsf{Valid}(V,\mathsf{BB})$ checks:

(V1) All honest votes in V are in final set $\mathcal{F}$
(V2) Result res correctly aggregates $\mathcal{F}$
(V3) $|\mathcal{F}| \leq N$ (no ballot stuffing)

$\underline{\text{Violation of (V1):}}$ Some honest vote $(\mathsf{state},\beta,\sigma) \in V$ has $\beta = (c,c',\gamma)$ with $c \notin \mathcal{F}$.

$\underline{\text{Case 1: } (\mathsf{ballot},\beta) \notin \mathsf{BB}}$. Honest voter produces evidence $(\mathsf{drop},\ldots)$ using state containing $(\mathsf{cred}_j,v,\ell,r)$. By binding of Com, this is valid evidence. Jdg blames EA.

Case 2: $(\mathsf{ballot},\beta) \in \mathsf{BB}$ but $c \notin \mathcal{F}$. Either:

- Higher-counter ballot exists for same nullifier: Voter's fault (revoted), not a violation
- No higher-counter ballot: Evidence $(\mathsf{not\text{-}counted},\ldots)$ blames trustees

$\underline{\text{Violation of (V2):}}$ Published $\mathsf{res} \neq \rho(\mathcal{F})$.

Since universal verification (Theorem 3) ensures anyone can verify the value of $C = \sum_{c\in\mathcal{F}} c$ and $\mathsf{res} = \mathsf{Dec}(C)$; inconsistency requires either:

- Wrong aggregation C: Publicly detectable, Vf fails, blame on party publishing wrong C
- Wrong decryption: Some δ_i is incorrect. Evidence $(\mathsf{bad\text{-}pdec},\ldots)$ from any party knowing correct (s_i,ω_i) blames T_i. With $\geq t$ honest trustees, at least one produces correct δ_i; combining reveals the discrepancy.

Violation of (V3): $|\mathcal{F}| > N$.
Directly caught by Jdg line 5, returning $(\mathsf{Trustees}, \mathsf{err}_3)$.

In all cases, violation of $\mathsf{Valid}(V, \mathsf{BB})$ produces valid evidence, and Jdg returns non-$\perp$ blame. Completeness violation probability:

$$\Pr[\text{Completeness violated}] \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda) \leq \mathsf{negl}(\lambda)$$

(The IND-CPA term bounds the probability that the adversary creates an undetectable inconsistency in encrypted values.) ■
Now we can have the final bound as:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{acc}}_{\mathcal{A},\mathsf{VS}}(\lambda) &= \Pr[\mathsf{Exp}^{\mathsf{acc}}_{\mathsf{VS},\mathcal{A}}(\lambda) = 1] \\ &\leq \Pr[\text{Fairness violated}] + \Pr[\text{Completeness violated}] \\ &\leq \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{Adv}^{\mathsf{bind}}_{\mathsf{Com}}(\lambda) + \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{VSS}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda) \\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

under collision resistance of H, binding of Com, soundness of VSS, and IND-CPA security of BFV, all in QROM. □

A.7 Proof of Theorem 7

Proof. We prove that VS securely realizes the ideal functionality $\mathcal{F}_{\mathsf{voting}}(\rho)$ by establishing ballot privacy, strong consistency, and strong correctness, then applying the composition theorem of [18]. $\mathcal{F}_{\mathsf{voting}}(\rho)$ interacts with environment $\mathfrak{E}$ and simulator $\mathcal{S}$:

1. On input $\mathsf{vote}(id, v)$ from $\mathfrak{E}$ or $\mathcal{S}$: store (id, v), send $\mathsf{ack}(id)$ to $\mathcal{S}$
2. On input tally from $\mathcal{S}$: return $\rho(\{(id, v)\})$ to both $\mathfrak{E}$ and $\mathcal{S}$, halt

Lemma 14. VS *satisfies ballot privacy in the sense of [18]: for all QPT* $\mathcal{A}$*, there exists QPT* $\mathcal{S}$ *such that* $\left|\Pr[\mathsf{Exp}^{\mathsf{bpriv},0}_{\mathsf{VS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{bpriv},1}_{\mathsf{VS},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$.

Proof: The ballot privacy experiment presents $\mathcal{A}$ with either real ballots (in world 0) or simulated ballots (in world 1), where simulation uses $\mathsf{SimVote}$ producing ballots with dummy content and $\mathsf{SimTally}$ producing consistent proofs.

We construct the simulator as follows.

- $\mathcal{S}.\mathsf{SimVote}(id)$: Sample ρ_1, ρ_2, r uniformly. Compute:

  $$c \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, \mathsf{EncS}(1); \rho_1),\ c' \leftarrow \mathsf{BFV.Enc}(\mathsf{pk}, 0; \rho_2),\ \gamma \leftarrow H(c \| c' \| 0 \| 0 \| r)$$

  Return (c, c', γ).
- $\mathcal{S}.\mathsf{SimTally}(\mathsf{BB}, R)$: Given result $R = (r_0, \ldots, r_{k-1})$ and BB:
  1. Compute $\mathcal{F}$ as in real protocol (using real identifier decryptions)
  2. Compute aggregate $C = \sum_{c \in \mathcal{F}} c$
  3. For each $i \in S$: compute $\delta_i = C_1 \cdot s_i$ and $\pi_i = H(\delta_i \| \Gamma_i)$

4. Verify $\mathsf{VSS.Comb}$ yields M with $\mathsf{DecS}(M) = R$

We use a hybrid argument.
H_0: Real experiment. Ballots encrypt real votes.
H_1: Replace vote ciphertexts with encryptions of dummy vote $\mathsf{EncS}(1)$.

$$|Pr[H_0 = 1] - \Pr[H_1 = 1]| \leq Q_v \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)$$

where Q_v is the number of $\mathcal{O}\mathsf{Vote}$ queries.
H_2: Replace identifier ciphertexts with encryptions of 0.

$$|\Pr[H_1 = 1] - \Pr[H_2 = 1]| \leq Q_v \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)$$

H_3: Use $\mathsf{SimTally}$ instead of real tally.
The observation is that $\mathsf{SimTally}$ produces the same distribution as a real tally when:

- The result R is computed from real votes (known to sthe imulator via ideal functionality)
- Partial decryptions are computed correctly using real key shares

Since H_2 already uses dummy ciphertexts, the only information in the tally is the aggregate plaintext, which equals R by construction; $|\Pr[H_2 = 1] - \Pr[H_3 = 1]| = 0$. The H_3 is the simulated experiment. Total advantage will be: $\leq 2Q_v \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda) \leq \mathsf{negl}(\lambda)$ ■

Lemma 15. VS *is strongly consistent (Definition 19).*

Proof: We construct the required algorithms:
$\mathsf{Extract}(b,\mathsf{sk})$: Given ballot $b = (c,c',\gamma)$ and secret key $\mathsf{sk} = \sum_{i \in \mathcal{Q}} s_i$:

1. Decrypt identifier: $m' \leftarrow \mathsf{BFV.Dec}(\mathsf{sk},c')$
2. Parse: $(\nu,\ell) \leftarrow \mathsf{DecID}(m')$
3. Decrypt vote: $m \leftarrow \mathsf{BFV.Dec}(\mathsf{sk},c)$
4. Decode: $v \leftarrow \mathsf{DecS}(m)$ or $\mathsf{DecR}(m)$
5. Return (ν,v) where ν serves as upk

$\mathsf{ValidInd}(b,\mathsf{pk})$: Check that $b = (c,c',\gamma)$ has:

- $c,c' \in R_q^2$ (valid ciphertext format)
- $\gamma \in \{0,1\}^\lambda$ (valid commitment format)

Correctness of $\mathsf{Extract}$: For honestly generated ballot $b \leftarrow \mathsf{Vote}(id,v,\mathsf{pk},\mathsf{cred})$:

$$\begin{aligned} \mathsf{BFV.Dec}(\mathsf{sk},c) &= \mathsf{BFV.Dec}(\mathsf{sk},\mathsf{BFV.Enc}(\mathsf{pk},\mathsf{EncS}(v);\rho_1)) \\ &= \mathsf{EncS}(v) \quad \text{(by correctness of BFV)} \end{aligned}$$

Thus $\mathsf{Extract}$ recovers v correctly with overwhelming probability.
Consistency Experiment: In $\mathsf{Exp}^{\mathsf{consis}}_{\mathcal{A},\mathsf{VS},I}(\lambda)$ (Figure 4a):

- Adversary produces BB with all ballots passing $\mathsf{ValidInd}$
- Real tally: $(r,\Pi) \leftarrow \mathsf{Tally}(\mathsf{BB},\mathsf{sk})$
- Extracted tally: $r' \leftarrow \mathsf{Counting} \circ \mathsf{Policy}(d\mathsf{BB})$ where $d\mathsf{BB}[i] = \mathsf{Extract}(\mathsf{BB}[i],\mathsf{sk})$

We need $r = r'$ with overwhelming probability. Both computations:

1. Decrypt all identifier ciphertexts to determine nullifiers and counters
2. Apply nullifier resolution to get final set
3. Aggregate and decrypt vote ciphertexts
4. Apply counting function

The only difference is:

- Real tally: Homomorphic aggregation then threshold decryption
- Extracted: Individual decryption then aggregation

By homomorphic property of BFV $\mathsf{BFV.Dec}\left(\mathsf{sk},\sum_{c\in\mathcal{F}} c\right) = \sum_{c\in\mathcal{F}} \mathsf{BFV.Dec}(\mathsf{sk},c)$. Thus $r = r'$ always, and $\Pr[\mathsf{Exp}^{\mathsf{consis}} = 1] = 0 \leq \mathsf{negl}(\lambda)$. ■

Lemma 16. VS *is strongly correct (Figure 4b).*

Proof: In $\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A},\mathsf{VS},I}(\lambda)$:

- Adversary produces (id,v,BB) with $id \in I$
- Condition e_1: No ballot in BB is attributed to id (different identity or different upk)
- Honest ballot: $b \leftarrow \mathsf{Vote}(id,v,\mathsf{pk},\mathsf{cred}_{id})$
- Condition e_2: $\mathsf{Valid}(\mathsf{BB},b,\mathsf{pk}) = 1$
- Experiment returns $e_1 \wedge \neg e_2$

We show $e_1 \wedge \neg e_2$ occurs with negligible probability. $e_2 = 0$ means $\mathsf{Valid}(\mathsf{BB},b,\mathsf{pk}) = 0$, i.e., ballot b is rejected. Valid checks:

1. $\mathsf{ValidInd}(b,\mathsf{pk}) = 1$: Honest ballot satisfies format checks
2. No conflicting ballot for same voter: e_1 ensures no ballot for id in BB

Since honest Vote produces well-formed ballot and e_1 guarantees no conflicts, $\mathsf{Valid}(\mathsf{BB}, b, \mathsf{pk}) = 1$ always. Thus $e_2 = 1$, and $e_1 \wedge \neg e_2 = e_1 \wedge 0 = 0$; so, $\Pr[\mathsf{Exp}^{\mathsf{corr}} = 1] = 0 \leq \mathsf{negl}(\lambda)$ ■

By [18], a voting scheme satisfying:

- Ballot privacy (Lemma 14)
- Strong consistency (Lemma 15)
- Strong correctness (Lemma 16)

securely realizes $\mathcal{F}_{\mathsf{voting}}(\rho)$. The simulator $\mathcal{S}$ for ideal-world execution:

1. On $\mathsf{ack}(id)$ from $\mathcal{F}_{\mathsf{voting}}$: Run $\mathcal{S}.\mathsf{SimVote}(id)$, add to simulated BB
2. On receiving $R = \rho(\cdot)$ from $\mathcal{F}_{\mathsf{voting}}$: Run $\mathcal{S}.\mathsf{SimTally}(\mathsf{BB},R)$
3. Forward all other messages appropriately

For all QPT environments $\mathfrak{E}$:

$$\begin{aligned}
&|\Pr[\mathsf{EXEC}_{\mathsf{VS},\mathcal{A},\mathfrak{E}}(\lambda)=1]-\Pr[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{voting}}(\rho),\mathcal{S},\mathfrak{E}}(\lambda)=1]|\\
&\leq \mathsf{Adv}^{\mathsf{bpriv}}_{\mathsf{VS}}(\lambda)+\mathsf{Adv}^{\mathsf{consis}}_{\mathsf{VS}}(\lambda)+\mathsf{Adv}^{\mathsf{corr}}_{\mathsf{VS}}(\lambda)\\
&\leq 2Q_v\cdot\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BFV}}(\lambda)+0+0\\
&\leq \mathsf{negl}(\lambda)
\end{aligned}$$

This establishes $\mathsf{EXEC}_{\mathsf{VS},\mathcal{A},\mathfrak{E}}(\lambda)\approx_c \mathsf{IDEAL}_{\mathcal{F}_{\mathsf{voting}}(\rho),\mathcal{S},\mathfrak{E}}(\lambda)$ under the RLWE assumption in QROM. □

A.8 Efficiency Analysis of VoteSentinel

Time Cost. The setup phase requires $O(n^2)$ operations for distributed key generation, dominated by VSS share distribution and verification among n trustees, where each trustee verifies $n-1$ shares and performs ML-KEM encryptions. Registration is linear in voter count: $O(N)$ for N voters, as each registration involves constant-time hash operations. The voting phase achieves $O(d+k)$ per ballot for polynomial degree d and k candidates, where BFV encryption operations over R_q dominate with two encryptions per ballot at cost $O(d)$ each. Tallying represents the most computationally intensive phase at $O(N\cdot L\cdot t\cdot d)$ for N voters with maximum L votes and threshold t, driven by threshold decryption of identifier ciphertexts for all submitted ballots, followed by nullifier resolution and homomorphic aggregation of the final set. Individual verification requires $O(d+k)$ to recompute a single ballot deterministically, while universal verification costs $O(N\cdot d+t+k)$ to recompute the homomorphic aggregate, verify t partial decryption proofs, and validate the tally constraint. Dispute resolution operates in $O(d)$ time, as evidence verification involves commitment opening and partial decryption recomputation, both polynomial in d. The overall protocol complexity is dominated by the tallying phase, making $O(N\cdot L\cdot t\cdot d)$ the bottleneck for large-scale elections.

Space Complexity. Storage requirements in VoteSentinel are primarily determined by the bulletin board and distributed trustee state. The setup phase produces $O(n\cdot d+n\cdot t)$ public data: VSS commitments require $O(n\cdot t)$ space for t commitments per trustee, public key shares consume $O(n\cdot d)$ for n ring elements, and key commitments add $O(n\cdot\lambda)$ for λ-bit hash outputs. Each trustee stores $O(d)$ private state for their key share. The bulletin board grows to $O(N\cdot L\cdot d)$ during voting as each of N voters may submit up to L ballots, with each ballot containing two BFV ciphertexts of size $O(d)$ plus a constant-size commitment. Tallying adds $O(N\cdot d+t\cdot d)$ to the bulletin board: the final set $\mathcal{F}$ is published at size $O(N\cdot d)$, the aggregate ciphertext costs $O(d)$, and t partial decryption shares with proofs require $O(t\cdot d)$ space. Registration credentials require $O(N\cdot\lambda)$ storage at the election authority for dispute resolution purposes. The dominant space cost is ballot storage on the bulletin board at $O(N\cdot L\cdot d)$, which can be optimized in practice by allowing voters to overwrite previous ballots rather than appending, reducing storage to $O(N\cdot d)$ when only the most recent ballot per voter is retained during voting, though full history may be maintained for auditability.

References

1. Daniel Rausch, Nicolas Huber, and Ralf Kusters. Verifiable E-Voting with a Trustless Bulletin Board . In *2025 IEEE 38th Computer Security Foundations Symposium (CSF)*, pages 457–472, Los Alamitos, CA, USA, June 2025. IEEE Computer Society.
2. Malte Bernhard and et al. Public evidence from secret ballots. In Robert Krimmer, Melanie Volkamer, Nadja Braun Binder, Norbert Kersting, Olivier Pereira, and Carsten Schürmann, editors, *Electronic Voting. E-Vote-ID 2017*, volume 10615 of *LNCS*. Springer, Cham, 2017.
3. Gilles Kaim, Sylvain Canard, Angèle Roux-Langlois, and Jacques Traoré. Post-quantum online voting scheme. In Malte Bernhard and et al., editors, *Financial Cryptography and Data Security. FC 2021 International Workshops. FC 2021*, volume 12676 of *LNCS*. Springer, 2021.
4. Ralf Kusters, Tomasz Truderung, and Andreas Vogt. Clash attacks on the verifiability of e-voting systems. In *2012 IEEE Symposium on Security and Privacy*, pages 395–409, 2012.
5. Josh Benaloh, Michael Naehrig, Olivier Pereira, and Dan S. Wallach. ElectionGuard: a cryptographic toolkit to enable verifiable elections. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5485–5502, Philadelphia, PA, August 2024. USENIX Association.
6. Rafieh MOSAHEB. *Quantum-Safe Electronic Voting Schemes*. PhD thesis, Université du Luxembourg, 2025.
7. Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. More efficient lattice-based electronic voting from NTRU. *IACR Communications in Cryptology*, 1(4), 2025.
8. Véronique Cortier, Joseph Lallemand, and Bogdan Warinschi. Fifty shades of ballot privacy: Privacy against a malicious board. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 17–32, 2020.
9. David Bernhard, Oksana Kulyk, and Melanie Volkamer. Security proofs for participation privacy, receipt-freeness and ballot privacy for the helios voting scheme. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17. ACM, 2017.
10. Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 779–798, 2016.
11. Thomas Haines and Ben Smyth. Surveying definitions of coercion resistance. Cryptology ePrint Archive, Paper 2019/822, 2019.
12. Thomas Haines, Rafieh Mosaheb, Johannes Müller, and Ivan Pryvalov. Sok: Secure e-voting with everlasting privacy. *Proceedings on Privacy Enhancing Technologies*, 2023(1):279–293, 2023.
13. Maxime Meyer and Ben Smyth. Exploiting re-voting in the helios election system. *Information Processing Letters*, 143:23–28, 2018.
14. Feng Hao, Luke Harrison, Saverio Veltri, Irene Pugliatti, Chris Sinclair, and Gareth Nixon. Experience from UNITA elections: Reconciling revote, E2E verifiability and low coercion. Cryptology ePrint Archive, Paper 2025/1669, 2025.
15. David Basin, Saa Radomirovi, and Lara Schmid. Dispute resolution in voting. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 1–16, 2020.
16. Mike Graf, Ralf Kusters, Daniel Rausch, Simon Egger, Marvin Bechtold, and Marcel Flinspach. Accountable Bulletin Boards: Definition and Provably Secure Implementation . In *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*, pages 201–216, Los Alamitos, CA, USA, July 2024. IEEE Computer Society.

17. Constantin-Cătălin Drăgan, François Dupressoir, Kristian Gjøsteen, Thomas Haines, Peter B. Rønne, and Morten Ryberg Solberg. Machine-checked proofs of accountability: How to sElect who is to blame. In Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis, editors, *Computer Security – ESORICS 2023*, volume 14346 of *LNCS*. Springer, Cham, 2024.
18. David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy*, pages 499–516, 2015.
19. S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 12 pp.–42, 2006.
20. IACR. IACR News Item: 2025 Election Update. https://www.iacr.org/news/item/27138, November 2025. Accessed: 2025-12-23.
21. J. Alex Halderman and Edward W. Felten. Secure, auditable, and usable voting with cryptography. In *Proceedings of the 11th Conference on USENIX Security Symposium*, SSYM '02, pages 255–270. USENIX Association, 2002.
22. Ben Adida. Helios: web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, page 335348, USA, 2008. USENIX Association.
23. Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: A simple private and verifiable electronic voting system. In Joshua Guttman, Carl Landwehr, José Meseguer, and Duško Pavlovic, editors, *Foundations of Security, Protocols, and Equational Reasoning*, volume 11565 of *LNCS*. Springer, Cham, 2019.
24. Xavier Boyen, Thomas Haines, and Johannes Müller. Epoque: Practical end-to-end verifiable post-quantum-secure e-voting. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 272–291, 2021.
25. Sevdenur Baloglu, Sergiu Bursuc, Sjouke Mauw, and Jun Pang. Election verifiability revisited: Automated security proofs and attacks on helios and belenios. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pages 1–15, 2021.
26. Veronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *2011 IEEE 24th Computer Security Foundations Symposium*, pages 297–311, 2011.
27. Véronique Cortier and Pierrick Gaudry. Known caveats of belenios 3.1. Technical report, CNRS, Loria, April 2025. Version 3.1.
28. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 103–118, Berlin, Heidelberg, 1997. Springer.
29. ElectionGuard - Official — electionguard.vote. https://www.electionguard.vote/spec/. [Accessed 20-07-2024].
30. ElectionGuard - Official — electionguard.vote. https://github.com/microsoft/electionguard/releases/download/v2.1/EG_Spec_2_1.pdf. [Accessed 23-12-2024].
31. Pranav Baskar. Cryptographers held an election. they can't decrypt the results. *The New York Times*, November 2025. Accessed: 2025-12-23.
32. BBC News. Cryptology firm cancels elections after losing encryption key. *BBC News*, November 2025. Accessed: 2025-12-23.
33. Eike Hauck, Eike Kiltz, Johannes Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology CRYPTO 2020*, volume 12171 of *LNCS*. Springer, Cham, 2020.
34. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: New insights from a case study. In *2011 IEEE Symposium on Security and Privacy*, pages 538–553, 2011.

35. Josh Benaloh. Verifiable secret-ballot elections. 1987.
36. Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 16141625. ACM, 2016.
37. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, page 526535. ACM, 2010.
38. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Fast and compact subversion-resistant cryptosystems. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–276. Springer, 2010.
39. Carsten Baum, Ivan B. Damgård, Shengli Lu, K. Omkar, and Patrick Peer. Efficient lattice-based zero-knowledge proofs with improved concrete security. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, volume 10991 of *LNCS*, pages 3–33. Springer, 2018.
40. Vadim Lyubashevsky. Lattice-based zero-knowledge proofs for integer relations. In Shai Halevi, editor, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 92–110. Springer, 2012.
41. Damien Stehlé and Ron Steinfeld. Faster and more practical lattice-based cryptography. In Pil Joong Lee, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 237–261. Springer, 2011.
42. Diego F. Aranha, Dan Boneh, Rosario Gennaro, and Valerio Zappalà. Post-quantum lattice-based shuffle proofs and their applications. In Phong Q. Nguyen and Serge Vaudenay, editors, *Public-Key Cryptography – PKC 2021*, volume 12711 of *LNCS*, pages 297–326. Springer International Publishing, 2021.
43. Jonathan Bootle, Vadim Lyubashevsky, Cecilia Naya-Plasencia, and Tim Seiler. Exact zero-knowledge proofs for lattice-based cryptography. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2021*, volume 12826 of *LNCS*, pages 234–263. Springer International Publishing, 2021.
44. Sofiane Bouaziz, Maximilian Ermann, Eike Kiltz, Aline Labrabor, and Johannes Lehmkuhl. Efficient lattice-based blind signatures: Optimizations and implementation. IACR ePrint Archive, 2020. Cryptology ePrint Archive, Paper 2020/653.
45. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206. ACM, 2008.
46. Akshay Jain, Eike Kiltz, Chris Peikert, and Stefano Tessaro. New kinds of trapdoor functions from hard lattice problems. In Rei Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 692–710. Springer, 2012.
47. Divyansh et al. Aggarwal. Crystals-kyber: A quantum-resistant kem. *NIST Post-Quantum Cryptography Standardization*, 2023. Submission to the NIST Post-Quantum Cryptography Standardization Project.
48. Zvika Brakerski, Yuval Fan, and Frederik Vercauteren. Fully homomorphic encryption with a leveled structure. In *Proceedings of the 5th International Conference on Pairing-Based Cryptography*, volume 7281 of *LNCS*, pages 149–166. Springer, 2012.
49. Karim Baghery. *Π*: A unified framework for computational verifiable secret sharing. In Tibor Jager and Jia Pan, editors, *Public-Key Cryptography PKC 2025*, volume 15677 of *LNCS*. Springer, Cham, 2025.
50. Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security ESORICS 2013*, volume 8134 of *LNCS*. Springer, 2013.

51. Navid Abapour, Amir Kafshdar Goharshady, Constantin Catalin Dragan, and Mahdi Mahdavi. Blockchain-based economic voting with posterior security from lattices. In Carlos Cid and Naoto Yanai, editors, *Advances in Information and Computer Security (IWSEC 2025)*, volume 16208 of *LNCS*. Springer, Singapore, 2026.
52. Pavithra Sekar, J. Idhikash, Parth Agrawal, and J. R. Aarthy. Quantum-resistant electronic voting system using lattice cryptography techniques. In Maxim Bakaev, Radomir Bolgov, Anna Chizhik, Andrei Chugunov, Valeriia Demareva, Yury Kabanov, Roberto Pereira, R. Elakkiya, and Wei Zhang, editors, *Internet and Modern Society (IMS 2025)*, volume 2672 of *Communications in Computer and Information Science*, pages 257–270. Springer, Cham, 2026.
53. S. P. Mahanayak, B. Nikhita, and S. Bilgaiyan. Enhancing e-voting security with quantum-resistant encryption: A blockchain-based approach utilizing elliptic curve diffie–hellman and decentralized storage. *SN Computer Science*, 4(5):642, 2023.
54. Arome Junior Gabriel, Boniface Kayode Alese, Adebayo Olusola Adetunmbi, Olumide Sunday Adewale, and Oluwafemi Abimbola Sarumi. Post-quantum crystography system for secure electronic voting. *Open Computer Science*, 9(1):292–298, 2019.
55. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. A homomorphic LWE based e-voting scheme. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography – PQCrypto 2016*, volume 9606 of *Lecture Notes in Computer Science*, pages 245–265. Springer, Cham, 2016.
56. Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 15651581. Association for Computing Machinery, 2017.
57. Sunoo Park, Michael Specter, Neha Narula, and Ronald L. Rivest. Going from bad to worse: from internet voting to blockchain voting. *Journal of Cybersecurity*, 7(1):tyaa025, 2021.
58. Ralph C. Merkle. Secure communications over insecure channels. In Ralph C. Merkle, editor, *Proceedings of the Workshop on Computer Science*, pages 235–248. Stanford University, 1979.
59. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. In Gilles Brassard, editor, *Advances in Cryptology – EUROCRYPT '87*, volume 304 of *LNCS*, pages 149–165. Springer, 1988.
60. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010.
61. Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, volume 14929 of *LNCS*. Springer, Cham, 2024.
62. Xiayu Xiang, Shaoming Duan, Hezhong Pan, Peiyi Han, Jiahao Cao, and Chuanyi Liu. From one-hot encoding to privacy-preserving synthetic electronic health records embedding. In *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, CIAT 2020, page 407413. ACM, 2021.
63. Julian R. Ullmann. Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism. *ACM J. Exp. Algorithmics*, 15, February 2011.
64. Ronald L. Rivest and Michael L. Dertouzos. On data banks and privacy homomorphisms. 1978.
65. Véronique Cortier and Joseph Lallemand. Voting: You can't have privacy without individual verifiability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 5366. ACM, 2018.

66. Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In *Advances in Cryptology CRYPTO 2024*, page 352386. Springer, 2024.
67. Brent Waters and David J. Wu. Silent threshold cryptography from pairings: Expressive policies in the plain model. Cryptology ePrint Archive, Paper 2025/1547, 2025.
68. Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. Silent threshold encryption with one-shot adaptive security. Cryptology ePrint Archive, Paper 2025/1384, 2025.
69. Jan Bormet and et al. BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. Cryptology ePrint Archive, Paper 2025/1419, 2025.