Blockchain-based Economic Voting with Posterior Security from Lattices

Navid Abapour[1] Amir Goharshady[2] Catalin Dragan[1]

Mahdi Mahdavi[3]

[1]Surrey Centre for Cyber Security, University of Surrey

{n.abapour,c.dragan}@surrey.ac.uk

[2]Department of Computer Science, University of Oxford

amir.goharshady@cs.ox.ac.uk

[3]K-ryptography and Information Security for Open Networks,

Universitat Oberta de Catalunya

m_mahdavi@uoc.edu

September 2025

Abstract

Electronic voting has demonstrated that it streamlines the democratic process, making it more convenient for citizens and enhancing the accuracy and speed of election results in real-world scenarios in the US, Estonia, Switzerland, and many other countries. One major challenge for e-voting, especially online voting, is ensuring that voting and tallying devices behave honestly, particularly in cases involving monetary transactions. These are addressed by economic voting, where everything is on-chain; in essence, voters utilize smart contracts to conduct all voting stages. There are very few results on economic voting, and none post-quantum secure. The challenge comes from having the entire voting system run by smart contracts. In this work, we propose the first post-quantum economic voting scheme, which combines hybrid on- and off-chain operations, called the Post-Quantum Blind Vote (PQBV). The core idea is to utilize smart contracts that enable blind signatures during the voting process. We enhance our contribution by introducing a post-quantum blind signature with Posterior Security, as proposed by Yuen et al. (CCS 2025), which retroactively enhances the privacy of already generated signatures. This has a significant impact on PQBV, as it is able to satisfy formal cryptographic privacy definitions, including ballot privacy. Our efficiency analysis reveals competitive

performance compared to existing state-of-the-art post-quantum e-voting systems, such as Epoque (EuroS&P 2021), which is done without blockchain.

Keywords: Electronic Voting, Lattice-based Cryptography, Economic Voting, Blind Signature, Posterior Security

1 Introduction

Electronic voting is considered an outstanding step towards modern democracies, as it enhances accessibility, security, and efficiency in elections, driven by a growing interest among citizens in integrating technology into government affairs [1], as they represent a firmly established and expanding component of electoral infrastructure, demonstrably employed for a spectrum of elections—from country-wide polls to local contests—in numerous countries, including, but not limited to, the US, Estonia, India, Switzerland, France, and Australia [2]. In some real-world financial scenarios where an election takes place (e.g., an auction), the voting process happens off-chain, and then the transaction happens separately on-chain using cryptocurrency. This separation in the process is inefficient, as it requires additional time and energy. This challenge can be addressed by utilizing e-voting schemes based on smart contracts, which are referred to as Economic Voting[1] [3]. For example, in an online casino or an auction [4], it is more efficient to conduct part/or all of the procedure on-chain because the entire transaction, including the money and voting, will be processed directly on the same route, and provides better transparency on the money flow [4,5]. Additionally, it has been demonstrated that blockchain provides a reliable foundation for implementing a public bulletin board, offering a proactive approach to ensuring the integrity of published records by preventing retroactive modifications [6, 7].

However, not only are there a few promising results on blockchain-based voting, but also limitations occur as basing the voting system fully on-chain can fail to address core vulnerabilities like device compromise and network attacks [8] while introducing additional risks such as complexity, weak ballot secrecy, key management issues, scalability problems, and dependency on unproven technologies, and offer limited security benefits over simpler alternatives like traditional databases or paper ballots [9] and complicates governance and rapid vulnerability response. The existing economic approaches for voting are not cryptographically sufficient; for example, the Blind Vote lacks formal definitions of key terminology, such as secrecy and anonymity. Not only is this scheme not formalized, but it also lacks a formal proof of privacy and does not provide provable security guarantees [5]. It is based on classical computational assumptions, such as RSA, that, apart from its conventional vulnerabilities [10], is vulnerable to quantum attacks [11–13].

Despite the absence of post-quantum economic voting systems, from the post-quantum side, there has been substantial work on lattice-based non-economic e-

[1]Economic voting, where voting processes are integrated with financial transactions on a blockchain, offers advantages in scenarios requiring both democratic decision-making and monetary flows. For instance, in Decentralized Autonomous Organizations (DAOs), stakeholders vote on funding proposals using governance tokens, where the voting outcome directly triggers on-chain fund allocation (see `https://ethereum.org/en/dao/`).

voting [14–16]. However, it is not straightforward to integrate a smart contract into this approach, as it requires compatible assumptions and primitives. For example, Epoque [14] is a significant result in post-quantum non-economic voting; but, Epoque's utilization of multiple components, such as identity-based encryption with complex non-interactive zero-knowledge proofs, cannot be efficiently implemented or verified within the computational and gas constraints of current smart contract platforms.

In addition, as a recently presented concept, Posterior Security enables the enhancement of privacy properties for already-generated standard signatures without requiring access to the original signing keys [17], allowing applications to retroactively gain stronger anonymity and message hiding guarantees. This capability is particularly valuable in our blockchain voting context, as it permits the upgrading of deployed systems with enhanced privacy features while maintaining compatibility with existing signature infrastructure and enabling voters to obtain unlinkable ballots from standard blind signatures.

So, the problem is *how to achieve a posterior-secure voting scheme that can support economic operations and be securely used in the post-quantum era?*

Contributions. In this work, we use the potential of blockchain to design post-quantum economic voting with lattices, then we embed this variant into an efficient structure of Blind Vote [5], along with achieving ballot privacy. Our contributions include:

- presenting the first provable ballot-private voting system based on smart contract; a step towards bridging the gap between cryptographers and the cryptocurrency communities' idea on blockchain-based voting.
- constructing a Post-Quantum version of Blind Vote (PQBV) using lattices.
- demonstrating the first formalization and application of *Posterior Security* in blockchain, enabling anonymity and message hiding for an already generated standard signature, even by someone who has no access to the signing key.

Organization. Section 3 establishes the notation and building blocks necessary for our construction. Section 4 introduces the concept of posterior security and formalizes our blind signature scheme. Section 5 presents the complete construction of our post-quantum blockchain voting system PQBV, detailing the integration of functions with smart contracts. Section 6 presents a formal security proof that demonstrates our scheme achieves ballot privacy and unforgeability, while also analyzing its computational complexity and comparing our scheme's performance with Epoque [14]. Finally, Section 7 summarizes our contributions and discusses future research directions.

2 Related Works

To the best of our knowledge, no post-quantum economic voting scheme has been presented that has been proven to be ballot-private. Therefore, in this section, we highlight the need for such a scheme by examining the most similar schemes to ours, as presented from both cryptographic and cryptocurrency perspectives.

2.1 The Existing Lattice-based E-voting Schemes

To highlight the gap mentioned in the previous section, due to not considering economic operations and inheriting a high computational cost, these lattice-based schemes [14–16] cannot easily be transformed into variants with blockchain.

2.1.1 Epoque [14]

In this system, each voter V_i encodes their vote as $\mathbf{v}^i = (v^{i,j})_{j=1}^{n_{\mathsf{cand}}} \in \{0,1\}^{n_{\mathsf{cand}}}$ and secret shares each component among n_T trustees as $v^{i,j} = (v_1^{i,j}, ..., v_{n_T}^{i,j})$, then commits to each share using a homomorphic commitment scheme $c_k^{i,j} \leftarrow \mathsf{Com}(\mathrm{prm}_{\mathrm{com}}, v_k^{i,j}; r_k^{i,j})$ and encrypts the opening values under trustee T_k's IBE master public key using identity i: $e_k^i \leftarrow \mathsf{Enc}(\mathsf{prm}_k, i; (v_k^{i,j}, r_k^{i,j})_{j=1}^{n_{\mathsf{cand}}})$. During tallying, each trustee T_k uses master shortcut decryption to verify ballot validity and publishes $v_k^j \leftarrow \sum_{i=1}^{n_V} v_k^{i,j}$ and $r_k^j \leftarrow \sum_{i=1}^{n_V} r_k^{i,j}$ for each candidate j, with correctness verified via $\mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, v_k^j, c_k^j, r_k^j) = 1$ where $c_k^j \leftarrow \sum_{i=1}^{n_V} c_k^{i,j}$ using the homomorphic property. If a trustee claims a ballot is invalid, they must publish the voter's individual IBE secret key $\mathsf{msk}_k^i \leftarrow \mathsf{Extr}(\mathsf{prm}_k, \mathsf{msk}_k, i)$ to prove the claim.

2.1.2 Lattice-Based Electronic Voting from NTRU [15]

This schme works using NTRU encryption combined with threshold blind signatures, where each voter V_i encrypts their vote $v \in \mathcal{R}_p$ as $c = p(hs + e) + v \in \mathcal{R}_q$ using NTRU public key $h = g/f$ and encryption randomness $(s,e) \in \mathcal{S}_\nu^2$, then obtains a threshold blind signature σ from at least t signing authorities on the ciphertext c. The ballot $b = (c,\sigma)$ undergoes verifiable shuffling through ξ_1 mix servers using NTRU-based mix-nets with zero-knowledge proofs π_{Small} and π_{Shuf} to prove correct re-randomization and permutation, followed by threshold distributed decryption where each of ξ_2 decryption servers computes shares $ds_{i,j} = dk_j \cdot c_i + p \cdot E_{i,j}$ with noise drowning $E_{i,j} \leftarrow \mathcal{S}_{B_{\mathsf{Drown}}}$ and provides exact zero-knowledge proofs π_{Lin} and π_{Bnd} of correct computation and boundedness. Finally, the votes are recovered by combining decryption shares as $v_i = (\sum_{j \in [\xi_2]} ds_{i,j} \bmod q) \bmod p$.

2.1.3 Post-quantum E-voting Scheme from Ring-LWE [16]

The voting protocol operates in three phases: first, each voter V_i encrypts their vote $v \in \{0,1\}^*$ using a Ring-LWE-based encryption scheme to produce ciphertext $c = \mathsf{PK.Encrypt}(v, mk)$ where mk is the public election key. Second, the voter obtains a threshold blind signature $\sigma = \mathsf{BS.Sign}(\{A_j([sk]_j)\}_{j \in T}, V_i(pk, c))$ from at least t signing authorities on the ciphertext c, where the blind signature scheme is based on lattice trapdoors and provides perfect blindness. Finally, the ballot $b = (c,\sigma)$ is posted to the bulletin board, and after the election closes, at least t authorities reveal their decryption key shares $[ek]_j$ to reconstruct the full decryption key ek, allowing anyone to verify signatures and decrypt ballots to compute the election result $r = \{v_i : \mathsf{PK.Decrypt}(c_i, ek) = v_i \text{ for valid } (c_i, \sigma_i) \in BB\}$.

2.2 Economic E-voting Systems

These blockchain-based schemes [3, 5] lack long-term privacy and a post-quantum level of security, and their structures make it difficult to achieve posterior security with them. What makes it even harder is that putting lattice structures on-chain can consume a huge amount of gas on the smart contract, so a hybrid approach can be more effective, which we will present in Section 5 how we did it.

2.2.1 Tornado Vote [3]

This system combines Tornado Cash cryptocurrency mixing with zero-knowledge proofs, where voters deposit ERC-20 voting tokens into a vault with Pedersen hash $H_{Ped}(sect||k)$ (where *sect* is a random secret and k is a nullifier), then commit to their vote v by sending the first 20 bytes of $H_{SHA}(sec_c||v)$ along with $H_{Ped}(k)$ and a ZK proof of secret knowledge to achieve anonymity before voting. During the reveal phase, voters disclose v and sec_c through relayers to the smart contract, which verifies $H_{SHA}(sec_c||v)$ against stored commitments and transfers voting tokens to unowned addresses representing each vote choice, with nullifiers preventing double voting and final tallies determined by token balances at outcome addresses.

2.2.2 Blind Vote [5]

In this scheme, voters generate RSA key pairs (N_i,e_i,d_i) and obtain blind signatures from the administrator on their public key hash $h_i = \mathsf{hash}(N_i,e_i)$ through the protocol $h'_i = h_i \cdot r_i^e \bmod N \to s'_i = (h'_i)^d \bmod N \to s_i = s'_i \cdot r_i^{-1}$, then commit to their vote by submitting (N_i,e_i,s_i,c_i,sc_i) where $c_i = \mathsf{hash}(v_i,x_i)$ and $sc_i^{e_i} = c_i \bmod N_i$ proves knowledge of the vote commitment. During the reveal phase, voters disclose (c_i,v_i,x_i) through relays for contract verification of $\mathsf{hash}(v_i,x_i) = c_i$, enabling anonymous vote tallying while maintaining complete blockchain verifiability through the blind signature protocol that decouples voter identity from vote content.

3 Preliminaries

Let $\lambda \in \mathbb{R}$ denote the security parameter. Given positive integers $m,n \in \mathbb{N}$, we define $[m] := \{1,2,\ldots,m\}$ and $[n,m] := \{n,n+1,\ldots,m\}$. We denote by $\mathbb{Z}_m$ the quotient ring of integers modulo m, where elements are represented using the interval $[-m/2,m/2) \cap \mathbb{Z}$, and by $\mathbb{Z}_m^*$ its group of units. For any vector $\mathbf{x} \in \mathbb{R}^m$, the notation $\|\mathbf{x}\|$ represents the Euclidean norm $\|\mathbf{x}\|_2$, while $\|\mathbf{x}\|_\infty$ denotes the maximum norm.

3.1 Computational Assumptions

We recall a set of hard mathematical problems required by the security proof for our construction.

Randomized One-More Inhomogeneous Short Integer Solution (rOM-ISIS) [18]

Let $n,m,q \in \mathbb{N}$ and let $\beta > 0$. In the random oracle model, given $\mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}$ and access to a random oracle $\mathcal{H}: \{0,1\}^* \to \mathbb{Z}_q^n$, the $\mathsf{rOM\text{-}ISIS}(n,m,q,\beta)$ problem asks an adversary to output $\ell+1$ pairs $\{(\mathbf{z}_i,\mathbf{t}_i)\}_{i=0}^{\ell}$ such that $\mathbf{A}\mathbf{z}_i = \mathbf{t}_i \pmod q$, $\|\mathbf{z}_i\| \le \beta$, and $\mathbf{z}_i \neq \mathbf{0}$, after making at most ℓ queries to $\mathcal{H}$.

Module Short Integer Solution (MSIS) [19]

Let $n,k,m \in \mathbb{N}$, $q \in \mathbb{N}$, and let $R_q = \mathbb{Z}_q[X]/(X^n+1)$. Let $\beta > 0$. Define $\mathsf{MSIS}(n,k,m,q,\beta)$ as the problem of finding a non-zero vector $\mathbf{z} \in R_q^m$ such that $\mathbf{A}\mathbf{z} = \mathbf{0} \pmod q$ and $\|\mathbf{z}\|_\infty \le \beta$, where $\mathbf{A} \leftarrow\$ R_q^{k\times m}$.

Module Learning with Errors (MLWE) [19]

Let $n,k,m \in \mathbb{N}$, $q \in \mathbb{N}$, and let $R_q = \mathbb{Z}_q[X]/(X^n+1)$. Let χ be a distribution over R_q. Consider $D_0 := (\mathbf{A},\mathbf{A}^T\mathbf{s}+\mathbf{e})$ and $D_1 := (\mathbf{A},\mathbf{u})$, where $\mathbf{A} \leftarrow\$ R_q^{k\times m}$, $\mathbf{s} \leftarrow\$ R_q^k$, $\mathbf{e} \leftarrow\$ \chi^m$, $\mathbf{u} \leftarrow\$ R_q^m$. The $\mathsf{MLWE}(n,k,m,q,\chi)$ problem is to decide between the distributions D_0 and D_1.

3.2 Building Blocks

Non-Interactive Blind Signature ($\mathsf{NIBS_{rOM}}$) [20]

This signature, which is based on Randomized One-More Inhomogeneous Short Integer Solution, utilizes a CPA-secure public key encryption scheme[2] $\mathsf{PKE} = (\mathsf{PKE.KeyGen},\mathsf{PKE.Enc},\mathsf{PKE.Dec})$, and a non-interactive zero-knowledge proof $\mathsf{NIZK} = (\mathsf{NIZK.Setup},\mathsf{NIZK.Prove},\mathsf{NIZK.Verify})$ for linear relations over $\mathbb{Z}_q$, lattice trapdoors $\mathsf{bLT} = (\mathsf{bLT.TrapGen},\mathsf{bLT.SamplePre})$, and hash function $H: \{0,1\}^* \to \mathbb{Z}_q^n$.

Definition 1. *For parameters $n = \mathsf{poly}(\lambda)$, $m > n\log q + \lambda$, prime q, Gaussian parameter of $\varsigma = \Omega(m)$, and norm bound as $\beta = \varsigma\sqrt{m}$, the scheme $\mathsf{NIBS_{rOM}} = (\mathsf{Setup},\mathsf{KeyGen}_S,\mathsf{KeyGen}_R,\mathsf{Issue},\mathsf{Obtain},\mathsf{Verify})$ operates as:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: Outputs the public parameters* $\mathsf{pp} = (\mathbf{A},\mathbf{B},\mathsf{pke.pk},\mathsf{NIZK.crs})$ *where* $\mathbf{A},\mathbf{B} \xleftarrow{R} \mathbb{Z}_q^{n\times 2m}$, $(\mathsf{pke.pk},\cdot) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, *and the setup of* $\mathsf{NIZK.crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ *for language* $\mathcal{L}_1 = \{(\mathbf{C},\mathbf{A},\mathbf{B},\mathsf{pke.pk},\mathsf{ct},\mathbf{w},\delta) : \exists(\mathbf{x},\mathbf{y},\mathbf{z},r)$ *satisfying the relation}.*
- $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KeyGen}_S(\mathsf{pp})$*: Gives secret key* $\mathsf{sk} = \mathbf{T}_C$, $\mathsf{vk} = \mathbf{C}$ *where* $(\mathbf{T}_C,\mathbf{C}) \leftarrow \mathsf{bLT.TrapGen}(1^\lambda,n,2m,q)$.
- $(\mathsf{sk}_R,\mathsf{pk}_R) \leftarrow \mathsf{KeyGen}_R(\mathsf{pp})$*: Outputs* $\mathsf{sk}_R = (\mathbf{x},\delta)$, $\mathsf{pk}_R = \mathbf{A}\cdot\mathbf{x} + H(\delta)$ *where* $\mathbf{x} \leftarrow D_{\mathbb{Z}^{2m},\varsigma/m}$, $\delta \xleftarrow{R} \{0,1\}^\lambda$.

[2] The choice of IND-CPA security (rather than CCA) is sufficient because the PKE is only used to encrypt the witness for the ZKP proof, and the ZKP provides the necessary non-malleability guarantees.

– $(\mathsf{psig},\mathsf{nonce}) \leftarrow \mathsf{Issue}(\mathsf{sk},\mathsf{pk}_R)$: *For* $\mathbf{y} \xleftarrow{R} \{-1,+1\}^{2m}$, *outputs* $\mathsf{psig}=\mathbf{z}$, $\mathsf{nonce}=\mathbf{y}$ *where* $\mathbf{z} \leftarrow \mathsf{bLT.SamplePre}(\mathbf{C},\mathbf{T}_C,\mathsf{pk}_R-\mathbf{B}\cdot\mathbf{y},\varsigma)$.

– $(\mu,\sigma) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R,\mathsf{vk},\mathsf{psig},\mathsf{nonce})$: *Verifies validity conditions. If valid, outputs* $\mu = (\mathbf{w},\delta)$, $\sigma = (\pi,\mathsf{ct})$ *where* $\mathbf{w} = \mathbf{A}\cdot[\mathbf{x}_\perp^T \| \mathbf{z}_\perp^T]^T$, *and the chiphertext* $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk},\mathbf{x}\|\mathbf{y}\|\mathbf{z})$, *and* π *is the* NIZK *proof for instance* $(\mathbf{C},\mathbf{A},\mathbf{B},\mathsf{pke.pk},\mathsf{ct},\mathbf{w},\delta) \in \mathcal{L}_1$.

– $b \leftarrow \mathsf{Verify}(\mathsf{vk},\mu,\sigma)$: *This sub-procedure outputs the result of verification by using zero-knowledge* $\mathsf{NIZK.Verify}(\mathsf{NIZK.crs},(\mathbf{C},\mathbf{A},\mathbf{B},\mathsf{pke.pk},\mathsf{ct},\mathbf{w},\delta),\pi)$.

The scheme achieves one-more-unforgeability under the $\mathsf{rOM}-\mathsf{ISIS}_{q,n,2m,\varsigma,3\sqrt{2}\beta}$ and receiver blindness under ZKP zero-knowledge and PKE semantic security.

Polynomial Commitment (PC)

The scheme from [21], based on Module Short Integer Solution, enables a prover to commit to a polynomial $h(X) \in \mathbb{Z}_p[X]$ and later prove evaluations $h(x)=y$ with square-root-sized proofs. The construction utilizes a modified Ajtai commitment [22] $\mathsf{Com}=(\mathsf{Setup},\mathsf{Com},\mathsf{Open})$ with randomized encoding $\mathsf{R.Ecd}: \mathbb{Z}_p^{d/r} \times \mathbb{R}_{>0} \to R$ mapping large prime field elements to small-coefficient ring elements, where $R=\mathbb{Z}[X]/(X^d+1)$ and $p=b^r+1$.

Definition 2. *For polynomial degree bound* $N=nm$ *and security parameter* λ, *the polynomial commitment scheme* $\mathsf{PC}=(\mathsf{Setup},\mathsf{Com},\mathsf{Open},\mathsf{Eval},\mathsf{Verify})$ *consists of:*

– $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, N)$: *Outputs commitment key* $\mathsf{ck} = (A_0, A_1)$ *where* $A_0 \leftarrow \mathcal{U}(R_q^{\mu\times\ell})$, $A_1 = [A_1' | I_\mu] \in R_q^{\mu\times(\mu+\nu)}$ *with* $A_1' \leftarrow \mathcal{U}(R_q^{\mu\times\nu})$, *for* $n=d\ell/r$.

– $(\vec{h},\delta) \leftarrow \mathsf{Com}(\mathsf{ck},h(X))$: *For* $h(X)=\sum_{i=0}^{N-1} h_i X^i \in \mathbb{Z}_p[X]$, *outputs commitment* $\vec{h} = \vec{h}_0 \| \cdots \| \vec{h}_{m+1} \in R_q^{\mu(m+2)}$ *and opening* $\delta = (\vec{h},\vec{\eta})$ *where:*

  ∘ $\vec{h}_i \leftarrow \mathsf{R.Ecd}(\vec{h}_i; s_1)$, $\vec{\eta}_i \leftarrow D_{\mathbb{Z}^d}^{\mu+\nu,\sigma_1}$, $\vec{h}_i = A_0\vec{h}_i + A_1\vec{\eta}_i \pmod q$ *for* $i \le m$

  ∘ $\vec{h}_{m+1} \leftarrow \mathsf{R.Ecd}(\vec{h}_{m+1}; \sqrt{m+2}\cdot s_3)$, $\vec{\eta}_{m+1} \leftarrow D_{\mathbb{Z}^d}^{\mu+\nu,\sqrt{m+2}\cdot\sigma_3}$

– $b \leftarrow \mathsf{Open}(\mathsf{ck},\vec{h},h(X),\delta)$: *Outputs* 1 *iff* $\|2\vec{h}_i\|2\vec{\eta}_i\|_2 \le 2d\beta_{\mathsf{Open}}$ *for all* i *and polynomial reconstruction holds.*

– $(y,\rho) \leftarrow \mathsf{Eval}(x,\delta)$: *Computes* $\vec{e} = \sum_{i=0}^{m-1} \mathsf{Ecd}(x^{ni})\cdot\vec{h}_i + \mathsf{Ecd}(x)\cdot\vec{h}_m + \vec{h}_{m+1}$, *outputs* $y = \langle \mathsf{Dcd}(\vec{e}),(1,x,\ldots,x^{n-1})\rangle \pmod p$ *and proof* $\rho=(\vec{e},\vec{\varepsilon})$.

– $b \leftarrow \mathsf{Verify}(\mathsf{ck},\vec{h},x,y,\rho)$: *Accepts iff* $\|\vec{e}\|\vec{\varepsilon}\|_2 \le \beta_{\mathsf{Eval}}$ *and consistency checks pass.*

The scheme satisfies computational hiding *under* $\mathsf{MLWE}_{R,\nu,q,\sigma_1}$, binding *under* $\mathsf{MSIS}_{R,\mu,q,4\beta_{\mathsf{PC}}}$, *and* evaluation binding *under* $\mathsf{MSIS}_{R,\mu,q,2\beta_{\mathsf{Eval}}}$, *where* $\beta_{\mathsf{PC}} = \beta_{\mathsf{Eval}} + \frac{(b+1)(m+1)dr}{2}\cdot\beta_{\mathsf{Open}}$.

Blockchain Operations

We model the general blockchain as a system $\mathsf{Blockchain} = (\mathsf{Deploy}, \mathsf{Submit}, \mathsf{GetDecoys})$ with the following operations:

- $\mathsf{contract} \leftarrow \mathsf{Deploy}(\mathsf{code}, \mathsf{params})$: Deploys smart contract code with parameters to the blockchain, returning contract address.
- $\mathsf{tx} \leftarrow \mathsf{Submit}(\mathsf{contract.function}(\mathsf{args}), \mathsf{value})$: Submits a transaction calling contract function with arguments and optional payment value.
- $\mathsf{keys} \leftarrow \mathsf{GetDecoys}(\mathsf{contract}, k)$: Retrieves k dummy verification keys from the blockchain to create an anonymity set that will hide the actual signer's verification key during the posterior security conversion.

3.3 Ballot Privacy

Proving a voting scheme secure against ballot privacy (BPRIV) means that ballots do not leak information on votes. As formalized in Definition 3 by [23], it should be impossible, even for active adversaries who can submit arbitrary ballots.

Definition 3. *A voting scheme $\mathcal{V}$ has ballot privacy if there exists a simulator* Sim *such that no efficient adversary $\mathcal{A}$ can distinguish between the games* $\mathsf{Exp}^{\mathsf{BPRIV},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim},I}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{BPRIV},1}_{\mathcal{A},\mathcal{V},\mathsf{Sim},I}(\lambda)$ *defined in Figure1. That is, the expression*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{BPRIV},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim},I}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{BPRIV},1}_{\mathcal{A},\mathcal{V},\mathsf{Sim},I}(\lambda)=1\right]\right|$$

is negligible in λ, for any set of voters I.

An experiment is shown in Figure 1; in the experiments $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{V},I}$, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has access to the set of oracles $\mathcal{O} = \{Ocast, Ovote, Otally, Oboard\}$. The adversary is allowed to call the *Otally* oracle at most once. In this model, the adversary tries to distinguish between two worlds.

4 Intuition on Posterior Security

4.1 Syntax and Security Properties

Posterior Security addresses a gap by enabling stronger privacy features to be applied to standard signatures after they are generated, allowing applications like two-tier Central Bank Digital Currencies [17] to gain further post-hoc privacy guarantees without any changes to existing signing algorithms. From [17], in Definition 4, we formalize the concept of posterior security [3].

Definition 4. *Given a signature scheme* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, a* posterior security transformation $\Pi = (\mathsf{PS.Setup}, \mathsf{PS.Convert}, \mathsf{PS.CVerify})$ *enhances signatures with additional security properties post-generation, where*

[3] It provides two properties: posterior anonymity (via incognito signature) and message hiding (via concealed signature). For additional information, refer to Appendix A.2.

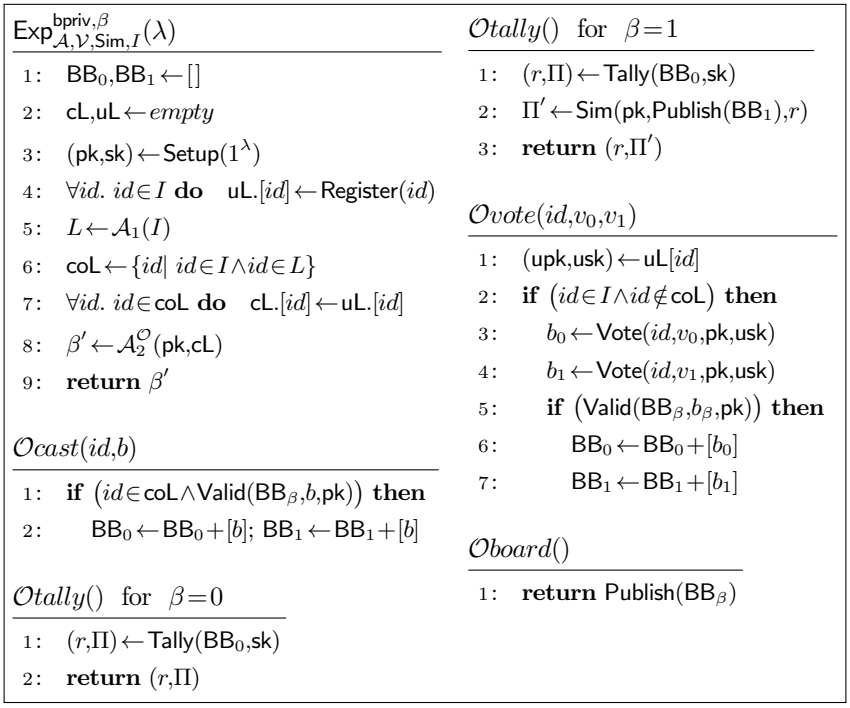

Figure 1: Ballot Privacy Experiment and Oracles [23]

- $\mathsf{PS.Setup}(1^\lambda,\mathsf{param}) \to \mathsf{param}_{\mathsf{PS}}$
- $\mathsf{PS.Convert}(\sigma,m,\mathsf{pk}^*,\mathsf{aux}) \to \sigma_{\mathsf{PS}} \cup \{\perp\}$ *outputs* $\perp \Leftrightarrow \mathsf{Verify}(\sigma,m,\mathsf{pk}^*) = 0$
- $\mathsf{PS.CVerify}(\sigma_{\mathsf{PS}},\mathsf{stmt},\mathsf{aux}) \to \{0,1\}$

have correctness: $\forall(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(),\sigma \leftarrow \mathsf{Sign}(m,\mathsf{sk})$*:*

$$\mathsf{PS.CVerify}(\mathsf{PS.Convert}(\sigma,m,\mathsf{pk},\mathsf{aux}),\mathsf{stmt},\mathsf{aux}) = 1.$$

Posterior Anonymity.

Let the Incognito Signature to be $\Pi_{\mathsf{IS}} = (\mathsf{IS.Setup},\ \mathsf{IS.Convert},\ \mathsf{IS.CVerify})$ *where* $\mathsf{aux} = \mathsf{pk} = \{\mathsf{pk}_1,...,\mathsf{pk}_n\}$, $\mathsf{pk}^* \in \mathsf{pk}$, *and* $\mathsf{stmt} = m$. *We define* $\mathsf{Exp}^{\mathsf{s-anon}}_{\mathcal{A},\Pi_{\mathsf{IS}}}(\lambda)$ *by:*

$$\begin{cases} (\mathsf{sk}_i,\mathsf{pk}_i)_{i\in[n]} \leftarrow \mathsf{KeyGen}(\omega_i);\mathsf{param}_I \leftarrow \mathsf{IS.Setup}() \\ (m^*,i_0,i_1,\mathsf{pk}^*,\sigma_0^*,\sigma_1^*) \leftarrow \mathcal{A}_1(\mathsf{param}_I,\{\mathsf{pk}_i,\omega_i\}_{i\in[n]}) \\ b \xleftarrow{\$} \{0,1\};\sigma_I^* \leftarrow \mathsf{IS.Convert}(\sigma_b^*,m^*,\mathsf{pk}_{i_b},\mathsf{pk}^*) \\ b' \leftarrow \mathcal{A}_2(\sigma_I^*);\textit{return } b' \stackrel{?}{=} b \end{cases}$$

where $\mathsf{pk}_{i_0},\mathsf{pk}_{i_1} \in \mathsf{pk}^*$ *and* $\sigma_j^* = \mathsf{Sign}(m^*,\mathsf{sk}_{i_j})$. *For the adversary's advantage in strong anonymity we have* $\mathsf{Adv}^{\mathsf{s-anon}}_{\mathcal{A}} := |\Pr[\mathsf{Exp}^{\mathsf{s-anon}} = 1] - 1/2| \leq \mathsf{negl}(\lambda)$.

Posterior Message Hiding.

Let the Concealed Signature to be $\Pi_{\mathsf{CS}} = (\mathsf{CS.Setup}, \mathsf{CS.Convert}, \mathsf{CS.CVerify}, \mathsf{CS.Decom})$ *where* $\mathsf{stmt} = \mathsf{pk}$. *We define* $\mathsf{Exp}^{\mathsf{s-hide}}_{\mathcal{A},\Pi_{\mathsf{CS}}}(\lambda)$ *as:*

$$\begin{cases} (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(); \mathsf{param}_{\mathsf{CS}} \leftarrow \mathsf{CS.Setup}() \\ (m_0^*, m_1^*, \sigma_0^*, \sigma_1^*) \leftarrow \mathcal{A}_1(\mathsf{param}_{\mathsf{CS}}, \mathsf{pk}, \mathsf{sk}) \\ b \xleftarrow{\$} \{0,1\}; (\sigma_{\mathsf{CS}}^*, \mathsf{aux}^*) \leftarrow \mathsf{CS.Convert}(\sigma_b^*, m_b^*, \mathsf{pk}) \\ b' \leftarrow \mathcal{A}_2(\sigma_{\mathsf{CS}}^*); \textit{return } b' \stackrel{?}{=} b \end{cases}$$

where $\sigma_j^* = \mathsf{Sign}(m_j^*, \mathsf{sk})$. *For the adversary's advantage in strong hiding we have:* $\mathsf{Adv}^{\mathsf{s-hide}}_{\mathcal{A}} \leq \mathsf{negl}(\lambda)$.

Unforgeability.

For $\mathsf{Exp}^{\mathsf{forge}}_{\mathcal{A}}(\lambda)$, *adversary* $\mathcal{A}$ *has oracle access to* $\mathcal{O}_{\mathsf{sign}}(m, \mathsf{pk}_i)$, $\mathcal{O}_{\mathsf{corr}}(\mathsf{pk}_i)$, *and* $\mathcal{O}_{\mathsf{conv}}(m, \mathsf{pk}', \mathsf{pk}_i)$. $\mathcal{A}$ *wins if it outputs* $(\mathsf{pk}^*, \sigma^*, m^*)$ *where the verification function* $\mathsf{PS.CVerify}(\sigma^*, \cdot, \mathsf{pk}^*) = 1$ *and* $\forall \mathsf{pk}_i \in \mathsf{pk}^*$*: no queries* (m^*, pk_i) *to* $\mathcal{O}_{\mathsf{sign}}$, pk_i *to* $\mathcal{O}_{\mathsf{corr}}$, *or* $(m^*, \mathsf{pk}^*, \mathsf{pk}_i)$ *to* $\mathcal{O}_{\mathsf{conv}}$, *then* $\Pr[\mathsf{Exp}^{\mathsf{forge}} = 1] \leq \mathsf{negl}(\lambda)$.

4.2 Posterior-Secure Blind Signature

We transform the lattice-based scheme $\mathsf{NIBS}_{\mathsf{rOM}}$ into a posterior-secure variant $\Pi_{\mathsf{PS-NIBS}}$ by augmenting it with posterior security transformations. Given $\mathsf{NIBS}_{\mathsf{rOM}}$ = (Setup, KeyGen_S, KeyGen_R, Issue, Obtain, Verify) remain unchanged (inherited from $\mathsf{NIBS}_{\mathsf{rOM}}$), we construct $\Pi_{\mathsf{PS-NIBS}}$ with additional algorithms, where first the base components get considered as a commitment scheme $\mathsf{COM} = (\mathsf{Com}, \mathsf{Decom})$ with binding and hiding properties, and an extended ZKP system for languages $\mathcal{L}_{\mathsf{anon}}$ and $\mathcal{L}_{\mathsf{hide}}$ defined in the Figure 2, Figure 6, and Figure 7.

Therefore, $\Pi_{\mathsf{PS-NIBS}}$ = (Setup, KeyGen_S, KeyGen_R, Issue, Obtain, Verify, PS.Setup, PS.Convert, PS.CVerify) where PS.Convert and PS.CVerify internally uses the IS/CS algorithms based on the chosen mode.

Theorem 1 investigates the posterior security of $\Pi_{\mathsf{PS-NIBS}}$, and establishes that our lattice-based construction maintains all essential security properties from the original blind signature scheme while adding the new posterior security capabilities.

Theorem 1. *The construction* $\Pi_{\mathsf{PS-NIBS}}$ *satisfies posterior security (Definition 4) under the following assumptions:*

1. *Posterior anonymity under* ZKP *zero-knowledge and* PKE *semantic security.*
2. *Strong posterior message hiding under* ZKP *zero-knowledge,* PKE *semantic security, and commitment hiding.*
3. *Unforgeability under* rOM-ISIS*, soundness of* ZKP*, and commitment binding.*

Proof. We prove each property separately. Given adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against $\mathsf{Exp}^{\mathsf{s-anon}}$, we construct a sequence of hybrids:

$\mathsf{PS.Setup}(1^\lambda,\mathsf{pp})$

1: $(\mathbf{A},\mathbf{B},\mathsf{pke.pk},\mathsf{nizk.crs}) \leftarrow \mathsf{pp}$
2: $\mathsf{nizk.crs}_{\mathsf{anon}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda,\mathcal{L}_{\mathsf{anon}})$
3: $\mathsf{nizk.crs}_{\mathsf{hide}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda,\mathcal{L}_{\mathsf{hide}})$
4: $\mathsf{pp}_{\mathsf{PS}} := (\mathsf{pp},\mathsf{nizk.crs}_{\mathsf{anon}},\mathsf{nizk.crs}_{\mathsf{hide}})$
5: **return** $\mathsf{pp}_{\mathsf{PS}}$

$\mathsf{PS.Convert}(\sigma,\mu,\mathsf{mode},\mathsf{aux})$

1: **if** $\mathsf{mode}=\mathsf{anon}$:
2: **return** $\mathsf{IS.Convert}(\sigma,\mu,\mathsf{aux})$
3: **if** $\mathsf{mode}=\mathsf{hide}$:
4: **return** $\mathsf{CS.Convert}(\sigma,\mu,\mathsf{aux})$

$\mathsf{PS.CVerify}(\sigma_{\mathsf{PS}},\mathsf{stmt},\mathsf{mode},\mathsf{aux})$

1: **if** $\mathsf{mode}=\mathsf{anon}$:
2: **return** $\mathsf{IS.CVerify}(\sigma_{\mathsf{PS}},\mathsf{stmt},\mathsf{aux})$
3: **if** $\mathsf{mode}=\mathsf{hide}$:
4: **return** $\mathsf{CS.CVerify}(\sigma_{\mathsf{PS}},\mathsf{stmt},\mathsf{aux})$

Figure 2: Posterior Security Algorithms

- Hyb_0: Real experiment $\mathsf{Exp}^{\mathsf{s-anon}}$ with $b=0$
- Hyb_1: Replace π_I with simulated proof of ZKP
- Hyb_2: Substitute ct_I with encryption of $0^{|\cdot|}$
- Hyb_3: As Hyb_2 but with $b=1$

By zero-knowledge property of ZKP we have $|\Pr[\mathsf{Hyb}_0=1]-\Pr[\mathsf{Hyb}_1=1]| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK}}(\lambda)$ and by semantic security property of the PKE scheme we have $|\Pr[\mathsf{Hyb}_1=1]-\Pr[\mathsf{Hyb}_2=1]| \leq \mathsf{Adv}^{\mathsf{sem}}_{\mathsf{PKE}}(\lambda)$, since $\mathcal{A}_2$ receives no information about i_b. By symmetry, $\Pr[\mathsf{Hyb}_2=1]=\Pr[\mathsf{Hyb}_3=1]=1/2$. Therefore, $\mathsf{Adv}^{\mathsf{s-anon}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK}}(\lambda) + \mathsf{Adv}^{\mathsf{sem}}_{\mathsf{PKE}}(\lambda)$. Hence, $\Pi_{\mathsf{PS-NIBS}}$ provides strong posterior anonymity.

Now, given adversary $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ against $\mathsf{Exp}^{\mathsf{s-hide}}$:

- Hyb_0: Real experiment with $b=0$
- Hyb_1: Replace π_{CS} with simulated proof
- Hyb_2: Swap com with commitment to $0^{|\mu|}$
- Hyb_3: Substitute $\mathsf{ct}_{\mathsf{CS}}$ with encryption of $0^{|\cdot|}$
- Hyb_4: As Hyb_3 but with $b=1$

Indistinguishability follows from ZKP zero-knowledge (Hyb_0 to Hyb_1), commitment hiding (Hyb_1 to Hyb_2), and PKE semantic security (Hyb_2 to Hyb_3), so, strong posterior message hiding is getting achieved by $\Pi_{\mathsf{PS-NIBS}}$.

To check unforgeability, given forger $\mathcal{F}$ outputting $(\mathsf{pk}^*,\sigma^*,\mu^*)$ where $\sigma^*=(\pi^*,\mathsf{ct}^*)$, by ZKP soundness, if $\mathsf{PS.CVerify}(\sigma^*,\cdot,\mathsf{pk}^*)=1$, then with overwhelming probability:

– For incognito: $\exists i \in [n]$ s.t. $\mathbf{C}_i \in \mathsf{pk}^*$ and valid (σ_i, μ_i) exists

– For concealed: $\exists$ valid (σ, μ) for the committed message

If $\mathcal{F}$ never queried for these values, we construct the rOM-ISIS solver: Extract witness (i^*, σ^*, μ^*) from π^* (by NIZK knowledge soundness), then parse $\mu^* = (\mathbf{w}^*, \delta^*)$, $\sigma^* = (\pi', \mathsf{ct}')$, eventally, if $\mathcal{F}$ never obtained signature on μ^* from $\mathbf{C}_{i^*}$, this breaks one-more-unforgeability of $\mathsf{NIBS}_{\mathsf{rOM}}$. Therefore, $\Pr[\mathsf{Exp}^{\mathsf{forge}} = 1] \leq \mathsf{Adv}^{\mathsf{omuf}}_{\mathsf{NIBS}}(\lambda) + \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIZK}}(\lambda) + \mathsf{Adv}^{\mathsf{bind}}_{\mathsf{COM}}(\lambda)$. □

5 Post-quantum Blind Vote

We present Post-quantum Blind Vote PQBV in Figure 3, a blockchain-based voting scheme that achieves voter privacy through polynomial commitments and posterior-secure blind signatures.

The election authority EA deploys the PQBV smart contract on the blockchain with the following parameters: PS-NIBS public parameters $\mathsf{pp}_{\mathsf{NIBS}}$, verification key vk_S for blind signature validation, polynomial commitment key $\mathsf{ck}_{\mathsf{PC}}$, and economic parameters including maximum voters $n_{\max}$, registration fee f, relay reward ρ, deposit δ, and time boundaries $\{t_i\}_{i=1}^{6}$. The protocol operates in distinct phases: registration, commitment, reveal, and tallying.

In this scheme, votes are encoded as polynomials rather than encrypted. Each voter encodes their vote $v \in \{0,1\}$ as a degree-1 polynomial $h(X) = v \cdot X^0 + r \cdot X^1$ where r is a random blinding factor. Privacy is achieved through: (1) polynomial commitments that hide the vote, (2) posterior-secure blind signatures that prevent linking voters to ballots, (3) decoy sets that provide anonymity among multiple verification keys, and (4) the commit-reveal mechanism where voters first commit to their vote polynomials and later reveal only evaluations at a blockchain-generated challenge point.

The setup algorithm generates PS-NIBS parameters and signer keypair $(\mathsf{sk}_S, \mathsf{vk}_S)$, initializes polynomial commitment with key $\mathsf{ck}_{\mathsf{PC}}$, deploys the smart contract with economic parameters, and outputs public key $\mathsf{pk} = (\mathsf{pp}_{\mathsf{NIBS}}, \mathsf{ck}_{\mathsf{PC}}, \mathsf{vk}_S)$ and secret key $\mathsf{sk} = \mathsf{sk}_S$.

The registration sub-procedure generates receiver keypair $(\mathsf{sk}_R, \mathsf{pk}_R)$ for the blind signature protocol, submits $h_{\mathsf{reg}} = H(\mathsf{id} \| \mathsf{pk}_R)$ to the blockchain with fee f, and outputs credential $c = (\mathsf{sk}_R, \mathsf{pk}_R)$.

The voting function obtains blind signature (μ, σ) through off-chain interaction with authority, encodes vote as polynomial $h(X) = v \cdot X^0 + r \cdot X^1$, creates commitment $(\mathsf{com}_v, \delta_v) \leftarrow \mathsf{PC.Com}(\mathsf{ck}_{\mathsf{PC}}, h(X))$, forms message $\mu_{\mathsf{com}} = (\mathsf{com}_v, H(\mathsf{id}))$, retrieves $k-1$ decoy verification keys from blockchain, converts signature to posterior-secure form $(\sigma_{\mathsf{PS}}, \mathsf{aux}) \leftarrow \Pi_{\mathsf{PS\text{-}NIBS}}.\mathsf{PS.Convert}(\sigma, \mu_{\mathsf{com}}, \mathsf{vk}_S, \{\mathsf{vk}_i\}_{i=1}^{k})$, commits $h_{\mathsf{com}} = H(\sigma_{\mathsf{PS}})$ on-chain receiving reward ρ, waits for challenge $x = H(\mathsf{block.number} \| \text{"challenge"})$, evaluates the polynomial $(y, \rho) \leftarrow \mathsf{PC.Eval}(x, \delta_v)$ where $y = h(x) = v + r \cdot x$, and reveals $(\sigma_{\mathsf{PS}}, \mathsf{aux}, \mu_{\mathsf{com}}, y, \rho)$ on-chain receiving another reward ρ.

Tallying collects all revealed evaluations $\mathcal{Y} = \{y_i\}$ and computes $r = \sum_i y_i \bmod p = \sum_i (v_i + r_i \cdot x) = \sum_i v_i + x \cdot \sum_i r_i$, which reveals the vote sum plus a randomness term dependent on challenge x. The authority generates a zero-knowledge proof

Π demonstrating correct aggregation locally (off-chain) and then submits it to the blockchain contract and publishes (r,Π).

Verification confirms all commitments have corresponding reveals, validates each reveal's posterior-secure signature, decommitment, and polynomial evaluation at challenge point x, ensuring protocol integrity without revealing individual votes.

Off-chain operations include all NIBS operations except final verification, polynomial creation and evaluation preparation, and signature aggregation for batch submission. As Figure 5 shows, it takes 32 bytes per vote (just the value), and 32 bytes per commitment in comparison with Blind Vote [5], taking $\sim$1 KB. Also, PQBV does not need storage of full signatures or proofs. Multiple voters can share one transaction, batch verification of NIBS signatures, and amortized gas costs across voters are other optimizations of PQBV compared to Blind Vote. The challenge has been simplified as well by providing a deterministic challenge from the block hash, and no need to store individual randomness.

6 Privacy and Efficiency Analysis

6.1 Ballot Privacy of PQBV

Having established the construction of our posterior-secure blind signature scheme $\Pi_{\mathsf{PS-NIBS}}$ and its integration into the PQBV voting scheme, we now turn to the formal privacy analysis; proceeding with Theorem 2 for showing PQBV satisfies the ballot privacy [23] definition (Definition 3).

Theorem 2. *Let $\Pi_{\mathsf{PS\text{-}NIBS}}$ be a posterior-secure non-interactive blind signature scheme (Definition 1) that satisfies unforgeability (Theorem 1), let* PC *be a polynomial commitment scheme (Definition 2) that satisfies hiding and binding properties, and let H be a random oracle. Then the* PQBV *voting scheme achieves ballot privacy BPRIV according to Definition 3.*

Before proving Theorem 2, we construct the simulator for PQBV as Sim in the Definition 5 in the following, and then prove ballot privacy through a sequence of hybrid games by Lemmas 3, 4, 5, 6.

Definition 5. *The simulator $\mathsf{Sim}(\mathsf{pk},\mathsf{BB}_1,r)$ takes as input the public key $\mathsf{pk} = (\mathsf{pp}_{\mathsf{NIBS}},\mathsf{ck}_{\mathsf{PC}},\mathsf{vk}_S)$, the bulletin board BB_1 from world $\beta = 1$, and the tally result r, and operates as follows: Extract all revealed polynomial evaluations $\mathcal{Y}_1 = \{y_i^{(1)}\}_{i=1}^n$ from $\mathsf{BB}_1.\mathsf{reveals}$, then calculate adjustment $\Delta = r - \sum_{i=1}^n y_i^{(1)} \bmod p$, and generate simulated proof $\Pi' \leftarrow \mathsf{ZK.Sim}(\mathsf{pk},r,\mathcal{Y}_1,\Delta)$. Finally return Π' as output.*

Proof of Theorem 2. Let $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$ be any polynomial-time adversary attacking the ballot privacy of PQBV.

- G_0: The real ballot privacy experiment $\mathsf{Exp}^{\mathsf{BPRIV},0}_{\mathcal{A},\mathsf{PQBV},\mathsf{Sim},I}(\lambda)$.
- G_1: Same as G_0, but polynomial commitments for honest voters are replaced with commitments to random degree-1 polynomials.

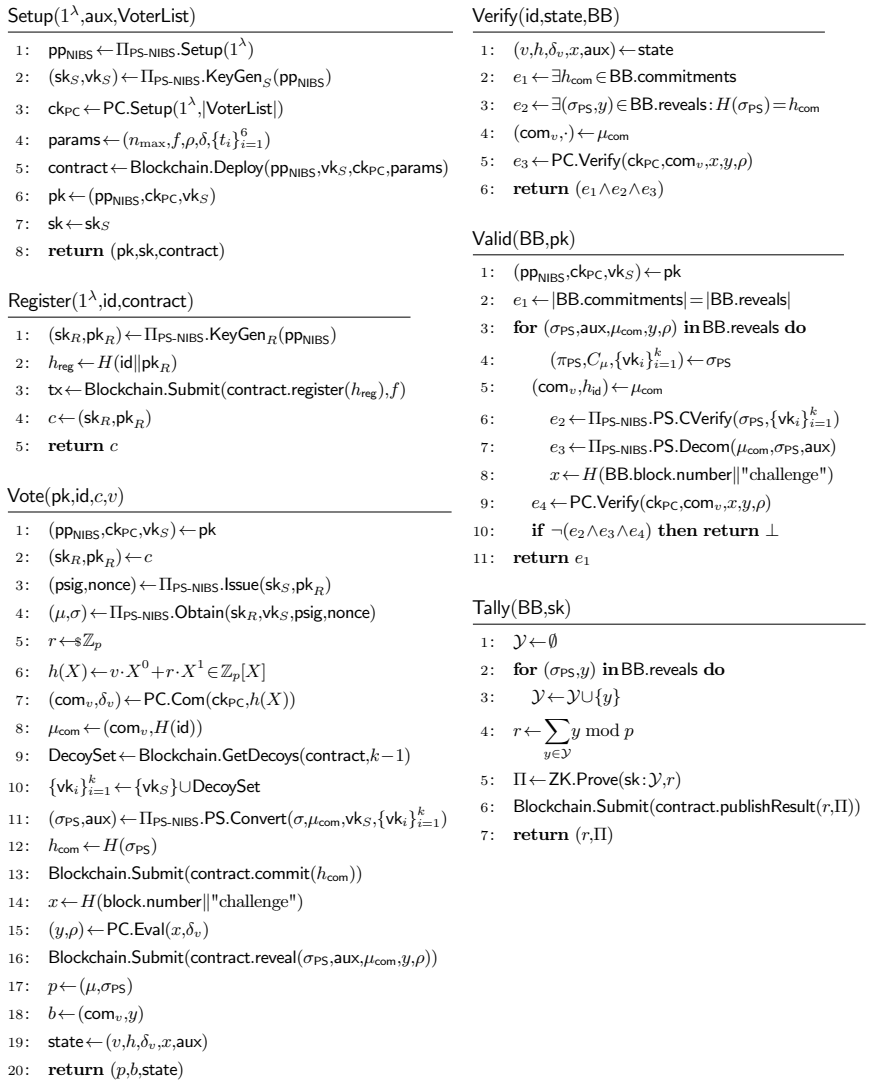

Figure 3: PQBV with $\Pi_{\mathsf{PS-NIBS}} = (\mathsf{Setup},\mathsf{KeyGen}_S,\mathsf{KeyGen}_R,\mathsf{Issue},\mathsf{Obtain},\mathsf{Verify},\mathsf{PS.Convert},\mathsf{PS.CVerify},\mathsf{PS.Decom})$, $\mathsf{PC} = (\mathsf{Setup},\mathsf{Com},\mathsf{Open},\mathsf{Eval},\mathsf{Verify})$, and $\mathsf{Blockchain} = (\mathsf{Deploy},\mathsf{Submit},\mathsf{GetDecoys})$.

- G_2: Same as G_1, but posterior-secure blind signatures for honest voters are replaced with simulated unlinkable signatures.
- G_3: Same as G_2, but polynomial evaluations are replaced with uniformly random values from $\mathbb{Z}_p$.
- G_4: The simulated ballot privacy experiment $\mathsf{Exp}^{\mathsf{BPRIV},1}_{\mathcal{A},\mathsf{PQBV},\mathsf{Sim},I}(\lambda)$.

Lemma 3. $|\Pr[G_0 = 1] - \Pr[G_1 = 1]| \leq \mathsf{negl}(\lambda)$

Proof of Lemma 3. We construct a reduction $\mathcal{B}_1$ that uses any distinguisher between G_0 and G_1 to break the hiding property of the polynomial commitment scheme PC. Given commitment hiding challenger with public parameters $\mathsf{ck}_{\mathsf{PC}}$, $\mathcal{B}_1$ proceeds as follows:

For each honest voter $i \in I$, $\mathcal{B}_1$ defines two polynomials $h_0^{(i)}(X) = v_i^{(0)} \cdot X^0 + r_i \cdot X^1$(real vote polynomial) and $h_1^{(i)}(X) = s_i \cdot X^0 + t_i \cdot X^1$(random polynomial), where $v_i^{(0)}$ is the vote in world $\beta = 0$, $r_i \leftarrow\!\$\, \mathbb{Z}_p$ is the blinding factor, and $s_i, t_i \leftarrow\!\$\, \mathbb{Z}_p$ are uniformly random. $\mathcal{B}_1$ sends $(h_0^{(i)}, h_1^{(i)})$ to the hiding challenger and receives commitment com_i^*. $\mathcal{B}_1$ embeds com_i^* as the polynomial commitment for voter i in the ballot privacy experiment. $\mathcal{B}_1$ simulates all other protocol components honestly, including the posterior-secure blind signature generation and the reveal phase. When $\mathcal{A}_2$ outputs a bit β', $\mathcal{B}_1$ outputs β'.

If the challenger commits to $h_0^{(i)}$, we have game G_0. If the challenger commits to $h_1^{(i)}$, we have game G_1. Therefore $|\Pr[\mathcal{B}_1 = 1 | \beta = 0] - \Pr[\mathcal{B}_1 = 1 | \beta = 1]| = |\Pr[G_0 = 1] - \Pr[G_1 = 1]|$; by the hiding property of PC, this advantage is negligible. □

Lemma 4. $|\Pr[G_1 = 1] - \Pr[G_2 = 1]| \leq \mathsf{negl}(\lambda)$

Proof of Lemma 4. We construct a reduction $\mathcal{B}_2$ that uses any distinguisher between G_1 and G_2 to break the unlinkability property of $\Pi_{\mathsf{PS\text{-}NIBS}}$. The unlinkability provides $\mathcal{B}_2$ with public parameters $\mathsf{pp}_{\mathsf{NIBS}}$ and signer verification key vk_S, access to a signing oracle that can produce blind signatures, and a challenge consisting of signatures that are either real or unlinkable. $\mathcal{B}_2$ proceeds as follows:

For each honest voter i, $\mathcal{B}_2$ generates receiver keypair $(\mathsf{sk}_R^{(i)}, \mathsf{pk}_R^{(i)}) \leftarrow \Pi_{\mathsf{PS\text{-}NIBS}}.\mathsf{KeyGen}_R(\mathsf{pp}_{\mathsf{NIBS}})$. $\mathcal{B}_2$ queries the signing oracle to obtain blind signatures (μ_i, σ_i) for messages $\mu_{\mathsf{com}}^{(i)} = (\mathsf{com}_i, H(\mathsf{id}_i))$. For the posterior-secure conversion, $\mathcal{B}_2$ generates decoy verification keys $\{\mathsf{vk}_j\}_{j=2}^k$ and sets $\{\mathsf{vk}_j\}_{j=1}^k = \{\mathsf{vk}_S\} \cup \{\mathsf{vk}_j\}_{j=2}^k$. $\mathcal{B}_2$ computes the posterior-secure signatures:

$$(\sigma_{\mathsf{PS}}^{(i)}, \mathsf{aux}^{(i)}) \leftarrow \Pi_{\mathsf{PS\text{-}NIBS}}.\mathsf{PS.Convert}(\sigma_i, \mu_{\mathsf{com}}^{(i)}, \mathsf{vk}_S, \{\mathsf{vk}_j\}_{j=1}^k)$$

The key insight is that the posterior-secure conversion with $k-1$ decoy keys makes it computationally infeasible to link $\sigma_{\mathsf{PS}}^{(i)}$ to voter i, even given sk_S. $\mathcal{B}_2$ embeds the challenge signatures in the ballot privacy experiment and outputs whatever $\mathcal{A}_2$ outputs.

The unlinkability property ensures that:

$$|\Pr[\mathcal{B}_2 = 1 | \text{real signatures}] - \Pr[\mathcal{B}_2 = 1 | \text{unlinkable signatures}]| \leq \mathsf{negl}(\lambda)$$

This directly translates to the bound in the lemma statement. □

Lemma 5. $|\Pr[G_2 = 1] - \Pr[G_3 = 1]| \leq \mathsf{negl}(\lambda)$

Proof of Lemma 5. For each honest voter i, the revealed value in G_2 is $y_i = h_i(x) = v_i + r_i \cdot x$, where $h_i(X) = v_i \cdot X^0 + r_i \cdot X^1$ is a random degree-1 polynomial (by the modification in G_1), $r_i \leftarrow\!\$ \mathbb{Z}_p$, and $x = H(\mathsf{block.number} \| \text{"challenge"})$. In this case, there are

some key observations: I) The challenge point x is chosen *after* the commitment phase via the random oracle H, making it unpredictable during commitment. II) The blinding factor r_i is chosen uniformly at random and independently for each voter. III) The term $r_i \cdot x$ is uniformly distributed in $\mathbb{Z}_p$ since r_i is uniform and x is independent of r_i.

Therefore, for any fixed vote value v_i, the distribution of $y_i = v_i + r_i \cdot x$ is uniform over $\mathbb{Z}_p$. In G_3, we replace each y_i with a uniformly random value $y_i' \leftarrow\$ \mathbb{Z}_p$. Since both distributions are uniform over $\mathbb{Z}_p$, they are statistically identical $|\Pr[G_2=1]-\Pr[G_3=1]|=0$. □

Lemma 6. $|\Pr[G_3=1]-\Pr[G_4=1]| \leq \mathsf{negl}(\lambda)$

Proof of Lemma 6. This follows from the zero-knowledge property of the aggregation proof system. In G_3, the tally oracle computes $r=\sum_{i=1}^{n} y_i \bmod p (y_i \sim \mathcal{U}(a,b))$ and $\Pi \leftarrow \mathsf{ZK.Prove}(\mathsf{sk}: \text{correct aggregation of } \{y_i\})$.

In G_4, the tally oracle uses the simulator for r = same result as in G_3 as $Pi' \leftarrow \mathsf{Sim}(\mathsf{pk}, \mathsf{BB}_1, r)$. The simulator Sim works by computing the difference $\Delta = r - \sum_j y_j^{(1)} \bmod p$ where $\{y_j^{(1)}\}$ are the values from world $\beta=1$, and using the zero-knowledge simulator: $\Pi' \leftarrow \mathsf{ZK.Sim}(\mathsf{pk}, r, \{y_j^{(1)}\}, \Delta)$. By the zero-knowledge property:

$$\{\Pi : \Pi \leftarrow \mathsf{ZK.Prove}(\mathsf{sk}: \text{statement})\} \approx_c \{\Pi' : \Pi' \leftarrow \mathsf{ZK.Sim}(\mathsf{pk}, \text{statement})\}$$

Therefore, $|\Pr[G_3=1]-\Pr[G_4=1]| \leq \mathsf{negl}(\lambda)$. □

Combining all Lemmas 3, 4, 5, 6 we have the final bound:

$$\begin{aligned}
&\left|\Pr[\mathsf{Exp}^{\mathsf{BPRIV},0}_{\mathcal{A},\mathsf{PQBV},\mathsf{Sim},I}(\lambda)=1]-\Pr[\mathsf{Exp}^{\mathsf{BPRIV},1}_{\mathcal{A},\mathsf{PQBV},\mathsf{Sim},I}(\lambda)=1]\right| \\
&= |\Pr[G_0=1]-\Pr[G_4=1]| \\
&\leq \sum_{i=0}^{3} |\Pr[G_i=1]-\Pr[G_{i+1}=1]| \leq 4 \cdot \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)
\end{aligned}$$

Therefore, PQBV achieves ballot privacy BPRIV according to Definition 3. □

6.2 Efficiency Analysis of PQBV

In [5], it has been demonstrated that Blind Vote outperforms other known economical blockchain-based e-voting schemes, including Boardroom Voting [24], Tornado Vote [3], and Metamask [25]. Here, we examine the post-quantum variant of Blind Vote, and among other schemes, Epoque [14] is the most similar to the structure of PQBV. As a result, PQBV achieves better efficiency than Epoque in the tally phase due to eliminating homomorphic decryption, while maintaining comparable or better performance in other phases.

In the Setup phase of PQBV, the PS-NIBS signer key generation needs $O(m^2 \cdot \mathsf{poly}(\lambda))$ for lattice trapdoor generation, polynomial commitment setup $O(N \cdot \mathsf{poly}(\lambda))$ where N is the degree bound. Then, in the Voting phase PS-NIBS Issue/Obtain takes $O(m^2 \cdot \mathsf{poly}(\lambda))$, polynomial commitment for degree-1 polynomial $O(\mathsf{poly}(\lambda))$,

	Setup Phase	Vote Phase	Tally Phase	Verification Phase
PQBV (Ours)	$O(m^2 \cdot \mathsf{poly}(\lambda) + N \cdot \mathsf{poly}(\lambda))$	$O(k \cdot m^2 \cdot \mathsf{poly}(\lambda))$	$O(n \cdot \mathsf{poly}(\lambda))$	$O(n \cdot k \cdot m^2 \cdot \mathsf{poly}(\lambda))$
Epoque [14]	$O(n \cdot \mathbb{t} \cdot m^3 \cdot \log^2 q)$	$O(\mathbb{v} \cdot \mathbb{c} \cdot \mathbb{t} \cdot m \cdot n \cdot \log^2 q)$	$O(\mathbb{v} \cdot \mathbb{t} \cdot m \cdot n \cdot \log^2 q)$	$O(\mathbb{v} + \mathbb{c} \cdot \mathbb{t})$

Table 1: Cumulative time complexity of schemes in each phase, assuming $\mathbb{c}$, $\mathbb{t}$, and $\mathbb{v}$ be the number of candidates, trustees, and voters, respectively, in Epoque.

PS.Convert with k decoys needs $O(k \cdot m^2 \cdot \mathsf{poly}(\lambda))$, on-chain commit $O(1)$, PC.Eval takes $O(\mathsf{poly}(\lambda))$, and on-chain reveal with PS.CVerify and PS.Decom verification $O(k \cdot m^2 \cdot \mathsf{poly}(\lambda))$. In Tally, collecting n revealed vote values takes $O(n)$, aggregating by simple summation $O(n)$, generating proof for correct aggregation $O(n \cdot \mathsf{poly}(\lambda))$, and publishing result $O(1)$. Finally, in global verification, it verifies all n reveals with PS.CVerify at cost $O(k \cdot m^2 \cdot \mathsf{poly}(\lambda))$ each.

7 Conclusion

In this work, we took a step toward bridging the gap between cryptographic security requirements and the cryptocurrency community's aspirations for blockchain-based voting by presenting PQBV, the first provably secure post-quantum blockchain voting system, along with addressing quantum threats through lattice-based primitives. The integration of posterior-secure blind signatures enables enhanced privacy guarantees, allowing for both voter anonymity and message hiding capabilities that can be applied retroactively to already-generated signatures. Through our analysis, we showed that PQBV offers promising efficiency compared to other schemes, such as Epoque, while maintaining formal security proofs for ballot privacy. As future directions, we aim for the realistic implementation of PQBV for demonstrating feasibility and discussing practical deployment.

8 Acknowledgment

N. Abapour was funded by the Computer Science Research Centre (Grant No. AB8031) and FEPS (Grant No. TB8071), both from UniOfSurrey. C. C. Drăgan is partially supported by TrustVote – EPSRC grant EP/Y020529/1, AP4L - EPSRC grant EP/W032473/1, CONNECT - Horizon Europe Guarantee 10043730 and EU Horizon grants 101069688, REWIRE - Horizon Europe Guarantee 10043743 and EU Horizon grants 101070627. M. Mahdavi was supported by the Spanish Ministry of Science and Innovation through the PID2021-125962OB-C31 "SECURING" project, along with the ARTEMISA International Chair of Cybersecurity (C057/23) and the DANGER Strategic Project of Cybersecurity (C062/23), both funded by the Spanish National Institute of Cybersecurity through the European Union — NextGenerationEU and the Recovery, Transformation, and Resilience Plan.

A Appendix

A.1 On-chain Operations of Smart Contract

According to Figure 5 and Figure 4, the smart contract operations enforce timing constraints, verify posterior-secure signatures using $\Pi_{\mathsf{PS\text{-}NIBS}}.\mathsf{PS.CVerify}$, validate decommitments via $\Pi_{\mathsf{PS\text{-}NIBS}}.\mathsf{PS.Decom}$, and confirm polynomial evaluations through $\mathsf{PC.Verify}$.

A.2 Concealed and Incognito Signatures

As Figure 6 shows, the incognito signature transformation enhances blind signatures with posterior anonymity by encrypting the signer's identity with signature components. A verifier receiving (π_I, ct_I) can confirm that some authorized signer from VK produced a valid signature without learning the specific signer. This leverages encryption, semantic security, and NIZK zero-knowledge to keep the encrypted identity computationally hidden, while π_I proves the ciphertext contains a valid signature from an authorized party.

```
pragma solidity ^0.8.19;
contract PosteriorSignatureCommitment {
    // commitment structure for posterior signatures
    struct PosteriorCommitment {
        bytes32 sigHash;       // H(\ sigma_PS )
        uint256 timestamp;
        bool revealed;
    }
    // commitment storage
    mapping(bytes32 => PosteriorCommitment) public commitments;
    // protocol parameters
    uint256 public t_commit_end;
    uint256 public relay_reward;
    // events
    event CommitmentSubmitted(bytes32 indexed sigHash);
    // batch commit posterior signatures with relay incentive
    function batchCommitPS(bytes[] memory signatures_PS) external payable {
        require(block.timestamp <= t_commit_end, "Commitment period ended");
        for (uint i = 0; i < signatures_PS.length; i++) {
            bytes32 h = keccak256(signatures_PS[i]);
            require(commitments[h].timestamp == 0, "Duplicate commitment");
            commitments[h] = PosteriorCommitment({
                sigHash: h,
                timestamp: block.timestamp,
                revealed: false
            });
            emit CommitmentSubmitted(h);
        }
        // single relay payment for batch submission
        payable(msg.sender).transfer(relay_reward * signatures_PS.length);
    }
}
```

Figure 4: Signature Commitment

```
pragma solidity ^0.8.19;
contract PosteriorSignatureReveal {
    // reveal function for posterior signatures with vote extraction
    function revealPS(
        bytes memory sigma_PS,
        bytes memory aux,
        uint256 com_vote,
        bytes32 voter_hash,
        uint256 y,
        bytes memory rho
    ) external {
        require(block.timestamp >= t_reveal_start, "Reveal not started");
        bytes32 h = keccak256(sigma_PS);
        PosteriorCommitment storage pc = commitments[h];
        require(pc.timestamp > 0 && !pc.revealed, "Invalid commitment");
        // Parse \ sigma_PS components
        (bytes memory pi_PS, bytes32 C_mu, address[] memory decoys) = parsePosteriorSig(sigma_PS);
        // Verify posterior - secure signature
        require(PS_NIBS.PS_CVerify(sigma_PS, decoys), "Invalid PS sig");
        // Decommit message
        bytes memory mu_com = abi.encode(com_vote, voter_hash);
        require(PS_NIBS.PS_Decom(mu_com, sigma_PS, aux), "Decommit failed");
        // Derive and verify challenge
        uint256 x = uint256(keccak256(abi.encode(block.number, "challenge")));
        require(PC.verify(ck, com_vote, x, y, rho), "Invalid eval");
        // Mark as revealed and store vote
        pc.revealed = true;
        votes.push(y);
        emit VoteRevealed(h, y);
    }
}
```

Figure 5: Posterior Signature Reveal

$\mathsf{IS.Convert}(\sigma,\mu,\mathsf{VK})$

$(\pi,\mathsf{ct}) \leftarrow \sigma$
$(\mathbf{w},\delta) \leftarrow \mu$
$\{\mathbf{C}_1,...,\mathbf{C}_n\} \leftarrow \mathsf{VK}$
$i^* \leftarrow \{i \in [n] : \mathsf{Verify}(\mathbf{C}_i,\mu,\sigma) = 1\}$
$\mathsf{ct}_I \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}, i^* \| \sigma \| \mu)$
$\mathcal{L}_{\mathsf{anon}} := \{(\mathsf{VK},\mathsf{ct}_I) : \exists (i,\sigma,\mu)$ s.t.
  $\mathsf{Verify}(\mathbf{C}_i,\mu,\sigma) = 1 \wedge \mathsf{ct}_I = \mathsf{PKE.Enc}(\mathsf{pke.pk}, i \| \sigma \| \mu)\}$
$\pi_I \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}_{\mathsf{anon}}, (\mathsf{VK},\mathsf{ct}_I), (i^*,\sigma,\mu))$
$\sigma_I := (\pi_I,\mathsf{ct}_I)$
return σ_I

$\mathsf{IS.CVerify}(\sigma_I,\mu',\mathsf{VK})$

$(\pi_I,\mathsf{ct}_I) \leftarrow \sigma_I$
$(\mathbf{w}',\delta') \leftarrow \mu'$
if $\mathbf{w}' \neq \mathbf{w} \vee \delta' \neq \delta$:
  return 0
return $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}_{\mathsf{anon}}, (\mathsf{VK},\mathsf{ct}_I), \pi_I)$

Figure 6: Incognito Signature (Posterior Anonymity)

Unlike the incognito variant that hides the signer's identity, concealed signature transformation (Figure 7) focuses on hiding the signed message content while maintaining verifiability. The commitment com serves as a public binding to the hidden message, and the zero-knowledge proof π_{CS} demonstrates that the encrypted data contains a valid signature on the committed message. The decommitment $\mathsf{CS.Decom}$ allows authorized parties to reveal the message.

The scheme $\Pi_{\mathsf{PS-NIBS}}$ is a single unified construction that supports both posterior security properties through the mode parameter. The party can generate standard blind signatures using original algorithms, convert them to incognito signatures using $\mathsf{PS.Convert}(\cdot,\cdot,\mathsf{anon},\cdot)$, and convert them to concealed signatures using $\mathsf{PS.Convert}(\cdot,\cdot,\mathsf{hide},\cdot)$. This modular design allows the same base signature to achieve different posterior security properties as needed. We investigate the security of $\Pi_{\mathsf{PS-NIBS}}$ in Section 6.

$\mathsf{CS.Convert}(\sigma,\mu,\mathbf{C})$

1: $(\pi,\mathsf{ct}) \leftarrow \sigma$
2: $(\mathbf{w},\delta) \leftarrow \mu$
3: $(\mathsf{com},r) \leftarrow \mathsf{Com}(\mu)$
4: $\mathsf{ct}_{\mathsf{CS}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk},\sigma\|\mu\|r)$
5: $\mathcal{L}_{\mathsf{hide}} \coloneqq \{(\mathbf{C},\mathsf{com},\mathsf{ct}_{\mathsf{CS}}) : \exists(\sigma,\mu,r)$ s.t.
$\mathsf{Verify}(\mathbf{C},\mu,\sigma)=1 \wedge \mathsf{com}=\mathsf{Com}(\mu;r)\}$
6: $\pi_{\mathsf{CS}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}_{\mathsf{hide}},(\mathbf{C},\mathsf{com},\mathsf{ct}_{\mathsf{CS}}),(\sigma,\mu,r))$
7: $\sigma_{\mathsf{CS}} \coloneqq (\pi_{\mathsf{CS}},\mathsf{ct}_{\mathsf{CS}},\mathsf{com})$
8: $\mathsf{aux}_{\mathsf{CS}} \coloneqq (\mu,r)$
9: **return** $(\sigma_{\mathsf{CS}},\mathsf{aux}_{\mathsf{CS}})$

$\mathsf{CS.CVerify}(\sigma_{\mathsf{CS}},\mathbf{C})$

1: $(\pi_{\mathsf{CS}},\mathsf{ct}_{\mathsf{CS}},\mathsf{com}) \leftarrow \sigma_{\mathsf{CS}}$
2: **return** $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}_{\mathsf{hide}},(\mathbf{C},\mathsf{com},\mathsf{ct}_{\mathsf{CS}}),\pi_{\mathsf{CS}})$

$\mathsf{CS.Decom}(\mu',\sigma_{\mathsf{CS}},\mathsf{aux}_{\mathsf{CS}})$

1: $(\mu,r) \leftarrow \mathsf{aux}_{\mathsf{CS}}$
2: $(\pi_{\mathsf{CS}},\mathsf{ct}_{\mathsf{CS}},\mathsf{com}) \leftarrow \sigma_{\mathsf{CS}}$
3: **return** $(\mu'=\mu) \wedge (\mathsf{Decom}(\mathsf{com},\mu,r)=1)$

Figure 7: Concealed Signature (Posterior Message Hiding)

References

[1] Josh Benaloh, Matthew Bernhard, J. Alex Halderman, Ronald L. Rivest, Peter Y. A. Ryan, Philip B. Stark, Vanessa Teague, Poorvi L. Vora, and Dan S. Wallach. Public evidence from secret ballots. *CoRR*, abs/1707.08619, 2017.

[2] Rihab H Sahib and Eman S. Al-Shamery. A review on distributed blockchain technology for e-voting systems. *Journal of Physics: Conference Series*, 1804(1):012050, feb 2021.

[3] Robert Muth and Florian Tschorsch. Tornado vote: Anonymous blockchain-based voting. In *ICBC*, pages 1–9, 2023.

[4] Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. Cicada: A framework for private non-interactive on-chain auctions and voting.

[5] Amir Kafshdar Goharshady and Zhaorun Lin. Blind vote: Economical and secret blockchain-based voting. In *IEEE Blockchain*, pages 46–53, 2024.

[6] Patrick Mccorry, Maryam Mehrnezhad, Ehsan Toreini, Siamak F. Shahandashti, and Feng Hao. On secure e-voting over blockchain. *Digital Threats*, 2(4), October 2021.

[7] *Securing the Vote: Protecting American Democracy*. National Academies Press, August 2018.

[8] MIT News. Mit researchers identify security vulnerabilities in voting app. `https://news.mit.edu/2020/voting-voatz-app-hack-issues-0213`, 2020. Accessed: 2025-01-26.

[9] U.S. Vote Foundation. The blockchain threat to democracy. `https://www.usvotefoundation.org/blockchain-threat-to-democracy`. Accessed: 2025-01-26.

[10] Mahdi Mahdavi, Navid Abapour, and Zahra Ahmadian. Trustworthy approaches to RSA: Efficient exploitation strategies based on common modulus. Cryptology ePrint Archive, Paper 2024/1903, 2024.

[11] Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. Transition to post-quantum cryptography standards. NIST Internal Report 8547, National Institute of Standards and Technology, November 2024. Initial Public Draft.

[12] Craig Gidney and Sophie Schmieg. Tracking the cost of quantum factoring. Google Security Blog, May 2025.

[13] Craig Gidney. How to factor 2048 bit rsa integers with less than a million noisy qubits, 2025.

[14] Xavier Boyen, Thomas Haines, and Johannes Müller. Epoque: Practical end-to-end verifiable post-quantum-secure e-voting. In *EuroS&P*, pages 272–291, 2021.

[15] Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. More efficient lattice-based electronic voting from NTRU. *IACR Communications in Cryptology*, 1(4), 2025.

[16] Guillaume Kaim, Sébastien Canard, Adeline Roux-Langlois, and Jacques Traoré. Post-quantum Online Voting Scheme. In *FC 2021 - Financial Cryptography and Data Security. International Workshops*, volume Lecture Notes in Computer Science, pages 290–305, Virtual event, France, March 2021.

[17] Tsz Hon Yuen, Ying-Teng Chen, Shimin Pan, Jiangshan Yu, and Joseph K. Liu. Posterior security: Anonymity and message hiding of standard signatures. Cryptology ePrint Archive, Paper 2025/855, 2025.

[18] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In *CCS*, CCS '22, page 39–53, New York, NY, USA, 2022. Association for Computing Machinery.

[19] Angèle Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75:565–599, 2015.

[20] Foteini Baldimtsi, Justin Cheng, Rachit Goyal, and Ayush Yadav. Non-interactive blind signatures: Post-quantum and stronger security. In Kai-Min Chung and Yuichi Sasaki, editors, *ASIACRYPT*, volume 15485 of *Lecture Notes in Computer Science*. Springer, Singapore, 2025.

[21] Intak Hwang, Jinyeong Seo, and Yongsoo Song. Concretely efficient lattice-based polynomial commitment from standard assumptions. In *CRYPTO*, page 414–448, Berlin, Heidelberg, 2024. Springer-Verlag.

[22] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, STOC '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.

[23] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *S&P*, pages 499–516, 2015.

[24] Patrick McCorry, Siamak F. Shahandashti, and Fang Hao. A smart contract for boardroom voting with maximum voter privacy. In Aggelos Kiayias, editor, *FC*, volume 10322 of *Lecture Notes in Computer Science*. Springer, Cham, 2017.

[25] Deni Pramulia and Bayu Anggorojati. Implementation and evaluation of blockchain based e-voting system with ethereum and metamask. In *ICIMCIS*, pages 18–23, 2020.