Practical Post-Quantum Secure Publicly Verifiable Secret Sharing and Applications

Aniket Kate
Mysten Labs / Purdue University
West Lafayette, USA
aniket@purdue.edu

Pratyay Mukherjee
Hashgraph
Kolkata, USA
pratyay85@gmail.com

Hamza Saleem
Independent Researcher
Los Angeles, USA
hamzasaleemzpr@gmail.com

Pratik Sarkar
Obsidus Labs
Bangalore, India
iampratiksarkar@gmail.com

Rohit Sinha
Hashgraph
Ohio, USA
sinharo@gmail.com

ABSTRACT

We present a new framework for constructing practically efficient publicly verifiable secret sharing (PVSS) with *non-interactive dealers*, in that the dealer may go offline after sending a single message, and is not involved in the share verification process. We use identity-based encryption (IBE) and commitments as the main ingredients and avoid expensive zero-knowledge proofs. By instantiating them with post-quantum secure schemes, a lattice-based IBE and a hash-based commitment, we obtain our first construction: a post-quantum secure PVSS with non-interactive dealers that outperforms the prior lattice-based practical construction by Gentry et al. [Eurocrypt 2022].

However, to enable the aggregation of PVSS transcripts (which facilitates many additional applications such as secure voting), we propose our second construction by replacing hash-based commitments with Pedersen's homomorphic commitments. While this does not achieve full-fledged post-quantum security (as Pedersen's scheme is not quantum safe), it still provides privacy against a post-quantum adversary. We prove the security of this construction in a new model, which we call long-lasting security. This model guarantees that the protocol is secure in the present (pre-quantum era) while maintaining privacy in the long term (post-quantum era). This new model is of independent interest for constructing efficient schemes that are resilient to the harvest-now-decrypt-later line of attacks. In this model, we propose a blockchain-compatible secure voting scheme using our PVSS scheme.

Our PVSS schemes demonstrate practical efficiency: our post-quantum PVSS shares a secret among 1024 receivers in 667 ms and verifies the dealing in < 5 ms, and communicates 4.4MB, overall whereas the state of the art [Gentry et al., Eurocrypt 2022] communicates 251MB and requires 2060s for dealing the secret.

KEYWORDS

Publicly Verifiable Secret Sharing, Post-Quantum Cryptography, Identity-Based Encryption, Long-Lasting Security

1 INTRODUCTION

Publicly verifiable secret sharing (PVSS) schemes [37] enable a dealer to share a secret s among n receivers (typically using Shamir's secret sharing [62]) in a publicly verifiable manner, such that anyone, even outside the system, can verify that the sharing has been done correctly. Any $t + 1$ ($t < n$) receivers can reconstruct the secret, while it stays hidden from any (possibly colluding) subset of t parties. PVSS has found numerous applications in the security space, including distributed key generation [13], threshold cryptosystems [34], secure voting [61], private auctions [65], and secure randomness on blockchain [50]. In particular, the public verifiability feature has emerged as one of the most crucial requirements for blockchain-based systems as it allows any potential user to ascertain the system's integrity.

In PVSS applications such as secure voting or secure auctions on blockchain, the protocol transcript is stored forever on the blockchain. In such cases, post-quantum security is crucial due to the ease of executing *harvest-now-decrypt-later* attacks [18, 19, 35][1] on the transcript. For example, votes cast now privately, if they get leaked in a decade with a plausible quantum attack, that could have significant repercussions for voters – such a possibility poses a pragmatic resistance for large-scale adaptability of any such applications today.

Moreover, for many client-server applications, including those that can be built over blockchains, such as ZK-as-a-service [39], MPC-as-a-service [17, 64, 66], private statistics [9] etc., it is important to keep the dealer/client non-interactive – the dealer posts/sends its message, and then can go offline. In particular, in order to produce a (publicly verifiable) correctness proof, the dealer should not be involved at all. We say that such PVSS has a non-interactive dealer.

With growing applications in the blockchain space, a few PVSS schemes has been proposed recently [27, 31, 44, 47, 49]. However, [4, 54] [2] are two proposals that are both *post-quantum secure* and have a *non-interactive dealer* follow a standard paradigm [3]: (i) given individual public keys for every receiver, the dealer encrypts their corresponding share with them plus also commits to the actual secret s; (ii) along with that it sends a non-interactive zero knowledge (NIZK) proof of correctness attesting to the fact that the encryptions indeed contains a t out of n Shamir shairng of the committed

[1] In this attack, the adversary collects/harvests the encrypted data available presently. Once quantum computers become practical, it can then break the encryption security to extract the plaintexts. Since all data on blockchains is essentially always available, one can easily harvest a massive quantity of data from blockchains to prepare for such attacks.

[2] [41] achieves post-quantum privacy and not full post-quantum security due to reliance on Discrete-Log based Bulletproofs [23].

[3] The same paradigm is followed by many works including [37, 55, 61], albeit in the classical/pre-quantum setting.

s. Clearly, as long as all the ingredients, namely, encryption, commitments, and NIZK are post-quantum secure, the scheme satisfies post-quantum security. Furthermore, the presence of NIZK ensures that dealer's message is already publicly verifiable without any additional step, thereby achieving the dealer's non-interactivity feature. Unfortunately, the final scheme incurs substantial overhead due to the use of post-quantum secure (lattice-based) NIZK proofs. This motivates us to ask the following question:

Can we construct a practically efficient post-quantum secure PVSS with a non-interactive dealer without expensive NIZKs?

We answer the question in the affirmative by proposing a new approach to obtaining PVSS with non-interactive dealers.

1.1 Our Contributions

A new approach to PVSS with non-interactive dealer. We use a post-quantum secure identity-based encryption scheme (IBE) as the core tool. Each receiver-i publishes an individual master IBE public key mpk_i for which they store a master secret key msk_i. The dealer posts a commitment to each receiver's share $[\![s]\!]_i$ as $\mathsf{Com}(s_i)$, and encrypts each receiver's share $[\![s]\!]_i$ (and opening randomness) under index i and the dealer's identity id_d using mpk_i and then goes offline. The receivers check that the posted commitments commit to a degree t sharing of a polynomial. Then the i-th receiver, once decrypts the share and opening randomness, checks with the commitment $\mathsf{Com}([\![s]\!]_i)$ whether the share it obtained is correct – if not, then the receiver uses msk_i to extract the dealer's decryption key $\mathsf{sk}_{i,\mathsf{id}_d}$ – using this, anyone (even someone who is neither the dealer nor a receiver) can decrypt the ciphertext and can verify whether the receiver's claim is correct by matching with the commitment. Firstly, this clearly does not involve a NIZK proof; and secondly, the IBE enables the dealer to remain non-interactive, as the receiver can extract the dealer's decryption key locally using her master secret key msk_i.

Practically efficient PQ secure PVSS with non-interactive dealer. In our first construction, we use a lattice-based IBE [7, 36] and hash-based commitments to obtain overall post-quantum security – this yields a post-quantum secure PVSS with a non-interactive dealer which is around *an order of magnitude* faster over state–of-the-art lattice based PVSS (viz. Gentry, Halevi and Lyubashevsky [41]).[4] Considering 1024 receivers, our PVSS protocol takes 667 ms to deal the secret and < 5 ms to verify the shares, whereas [41] takes roughly 34 minutes to perform the same task. Our PVSS communicates 4.4MB to perform the dealing, whereas [41] communicates 251MB.

Long-lasting security notion. The transcripts in our PVSS protocol are not aggregatable (as defined in [14]) due to the use of hash-based commitments, which are not homomorphic. As a result, we cannot use our hash-based PVSS protocol for applications like secure voting, DKG or secure data aggregation as they require aggregatable protocol transcripts. To obtain such aggregatable PVSS protocols, we relax the security notion. Specifically, we introduce the notion of *long-lasting security*, which provides full classical computational security during the execution and provides post-quantum privacy – this notion, while weaker than full-fledged post-quantum security, suffices to protect against harvest-now-decrypt-later attacks [18]. Long-lasting security provides soundness guarantees against classical adversaries but not against quantum adversaries, since long-lasting secure protocols exploit elliptic curve cryptography(along with post-quantum cryptography) for efficiency. The privacy is post-quantum secure since the primitives relying on elliptic curve cryptography provide perfect privacy, whereas the post-quantum secure primitives provide post-quantum privacy. We believe long-lasting security is a more realistic model compared to everlasting security (which requires information-theoretic privacy) in the post-quantum world as it allows employing post-quantum secure encryptions for secure data transmission which is not possible [11] in everlasting security model. For example, in our secure voting application, while the integrity/validity of votes is guaranteed at present, when quantum computers are not available, the votes would remain private even in a post-quantum world due to the usage of lattice-based encryptions and perfectly hiding commitments and NIZKs.

We note that the notion of long-lasting security can be used to capture current elliptic curve+lattice-based protocols (like Zama's FHEVM that uses Plonk proofs+TFHE) that are computationally sound against current classical adversaries but provide secrecy against future quantum adversaries. The long-lasting security model is of independent interest.

Practical long-lasting secure PVSS with non-interactive dealer. In this long-lasting setting, we propose our second PVSS construction by replacing the hash-based commitments with homomorphic Pedersen commitments (and adequately changing the verification procedure). Sharing a secret among 1024 receivers requires 908 ms, verifying a share requires 2.20 s, and the entire dealing requires 4.4MB communication.[5]

Secure Voting on Blockchain with long-lasting security. Using our aggregatable PVSS, we build a practical, secure voting protocol over blockchains in a client-server model. The entire protocol is publicly verifiable by any external party and provides long-lasting vote privacy, i.e. the vote of the client remains *private* even against a future quantum adversary. Conducting a poll with 128 servers and four poll options: client-side voting takes 0.11 s, vote verification takes 21 ms, and 100K votes are aggregated and decrypted in 67 ms - offering a 40× speedup over prior lattice-based voting schemes [22, 33].

We prove the security of all our protocols assuming an honest majority and synchronous communication channels. Our security proofs are in the classical random oracle model.

Other Instantiations: We note that the recent work of [6] constructs an isogeny-based IBE scheme. Even though their IBE construction is not practical, still it yields a candidate instantiation of our compiler from isogenies instead of lattice.

1.2 Related Works

We compare with the relevant works as follows.

[4] [41] does not achieve post-quantum security as mentioned in Table 1.

[5] Our notion of long-lasting security is inspired by the existing notion of everlasting security [67], in that the privacy is unconditional as opposed to post-quantum. Nevertheless, to instantiate any everlasting secure protocol [10] one needs to rely on secure channels, which must rely on computational assumptions, effectively rendering the overall protocol to our setting, if post-quantum secure channels are used. Our protocols do not rely on secure channels, but use authenticated bulletin boards or the blockchain or a broadcast channel.

Protocols	Non-Interactive Dealer	Long-lasting Security	Post-Quantum Security	Assumptions	Communication Model
Discrete-Log-based+Multisignatures [40, 55]	×	×	×	Discrete Log+Multisignatures	Broadcast + P2P
Discrete-Log-based [29, 37, 61]	✓	×	×	Discrete Log	Broadcast + P2P
Class-Group-based [27, 47]	✓	×	×	q-Hard subgroup membership	$\mathcal{F}_{\text{BC}}$/Broadcast
Signature-based [31]	×	✓	×	Perfectly hiding PCS+Signatures+ PQ-PKE (LWE/RLWE)	Broadcast + P2P
Bivariate Polynomial-based+ Multisignatures [5, 8]	✓	✓	×	Perfectly hiding PCS+Multisignatures+ PQ-PKE (LWE/RLWE)	Broadcast + P2P
Lattice-based [4, 54]	✓	✓	✓	LWE/Ring-LWE	Broadcast + P2P
Lattice+DL-based [41]	✓	✓	×	LWE/Ring-LWE+Discrete Log	Broadcast + P2P
Hash-based [12, 15]	×	✓	✓	RO + PQ-PKE (LWE/RLWE)	Broadcast + P2P
Ours (Hash-based)	✓	✓	✓	RO + PQ-IBE (LWE/RLWE)	$\mathcal{F}_{\text{BC}}$/Broadcast
Ours (Commitment-based)	✓	✓	×[†]	AHC + PQ-IBE (LWE/RLWE)	$\mathcal{F}_{\text{BC}}$/Broadcast

Note: PQ-PKE: Post-quantum Secure Public Key encryption, RO: Random Oracle, AHC: Perfectly Hiding Additively Homomorphic Commitment, P2P: Secure point-to-point channel

[†] Our Commitment-based PVSS protocol has aggregatable transcripts that enabling different applications. The hash-based PVSS lacks that property.

Table 1: Comparison of Publicly Verifiable VSS Schemes.

1.2.1 Comparison of PQ-secure PVSS Protocols. We compare our PQ-secure and long-lasting secure PVSS protocols with existing PVSS protocols from the literature. A qualitative comparison can be found in Table. 1.

Discrete Log-based protocols: In traditional discrete log-based protocols [29, 37, 40, 55, 61], the dealer shares a secret using Shamir's scheme, publishing commitments and encrypted shares. Some protocols [40, 55] allow receivers to complain about inconsistent shares, requiring dealer interaction for resolution. These schemes are not publicly verifiable unless receivers add an extra multisignature round. The works of [29, 37, 61] make the dealer non-interactive by publishing NIZK proofs that each encrypted share matches the secret polynomial. While verifiable, these protocols are not post-quantum secure due to reliance on DL, and replacing DL-based components with lattice-based ones is impractical due to the high cost of lattice-based NIZK proofs, preventing efficient post-quantum or long-lasting security.

Class-group-based protocols: The works of [27, 47] construct non-interactive PVSS using class group encryptions, with NIZK proofs ensuring correct encryption. Receivers verify the proofs and decrypt their shares. However, class group encryptions are not known to be post-quantum secure, and the NIZKs are customized for class group encryptions; hence, long-lasting secure protocols are hard to construct.

Lattice-based VSS protocols: The recent work of [4] constructed a post-quantum non-interactive PVSS from vector commitments that satisfy linear opening, functional hiding, and NIZK proof of smallness properties, and an encryption scheme that supports efficient proofs of correct decryption. Concurrently, the work of [54] constructed a non-interactive PVSS from an encryption and a NIZK proof system for gap language. However, the practicality of these works is unclear as they lack a concrete implementation and both of them rely on expensive NIZK proofs over lattices. The work of Gentry et al. [41] designed a PVSS scheme that uses an LWE-based encryption scheme and uses DL-based Bulletproofs [23] to prove that the ciphertext is well-formed. Using Bulletproofs improved efficiency over the pure lattice-based solutions, but it failed to provide practical efficiency, as it takes around 34 minutes for the dealer to deal a single secret among 1024 receivers, whereas our work performs the dealing and verification in less than a second. Moreover, due to reliance on Bulletproofs, the PVSS of [41] achieves long-lasting security and not post-quantum security.

Hash-based protocols: The work of [12] introduced an efficient hash-based VSS where the dealer non-interactively distributes shares using hash-based distributed ZK proofs. However, during share verification, receivers can file complaints, requiring the dealer to reveal corresponding shares—making the protocol interactive and its transcript not publicly verifiable. A later improvement by [15] proposed a general framework for computational VSS in the honest-majority setting using Shamir secret sharing, random oracles, and non-homomorphic commitments. Yet, both [12] and the post-quantum secure variants of [15] require dealer interaction in the verification phase. In contrast, our post-quantum secure PVSS removes this interaction by leveraging IBE and thereby enables many new applications.

Bivariate-Polynomial Based: The works of [5, 8, 51] construct asynchronous VSS using bivariate polynomials, where the dealer commits to a random low-degree polynomial and distributes partial evaluations with proofs to nodes, who then collaborate to reconstruct shares. These protocols avoid PKI but require $O(n^2)$ exponentiations and communication, and rely on q-SDH commitments, making them insecure against quantum adversaries. Long-lasting security and public verifiability are possible from them using perfectly hiding commitments, lattice-based encryption for channels, and multisignature rounds, though at the cost of higher dealer computation.

Signature-based VSS protocols: The recent work by [31] constructs an asynchronous dual threshold VSS protocol using Polynomial commitments [48] and signatures where the dealer shares the secret among n receivers and waits for signed confirmations from the receivers. Once the dealer receives confirmations from $t+1$ receivers it broadcasts the shares of the rest of the $n-t-1$ receivers and the $t+1$ signatures. This enables efficient reconstruction later on. However, their dealer is interactive in nature. We observe that this VSS protocol can be made long-lasting secure (albeit with an interactive dealer) if the underlying Polynomial commitment is instantiated using the Pedersen variant of KZG commitment, as it provides computational binding and perfect hiding, and the

point-to-point channels are replaced by a PQ-secure public key encryption.

1.2.2 Comparison between Everlasting and Long-lasting Security Models. Unruh [67] proposed the notion of everlasting security that requires the protocol to be secure against classical PPT adversaries during protocol execution and secure against computationally unbounded adversaries after protocol execution. They constructed quantum-UC secure communication protocols using trusted signature cards and quantum channels. Following this work, there has been constructed everlasting secure MPC protocols [10] in the honest majority setting, assuming secure channels. Then [46] constructed certified everlasting secure functional encryption assuming quantum channels. There have been secure voting protocols [45, 57] with everlasting privacy as well. However, all these protocols require quantum channels for secure communication. To implement a secure channel in the everlasting security model, one needs to have quantum-secure channels. An alternate solution was proposed by [35] where a secure channel is assumed between one designated party, called the hypervisor, and the protocol participants. We provide a detailed comparison in Appendix. A.

However, the current practical feasibility of quantum channels for regular cryptographic protocols is unclear, and the assumption of a secure channel between the parties and the hypervisor is also not realistic in nature. Hence, everlasting security is difficult to implement in practice. So, we relaxed the notion of everlasting security to long-lasting security where the post-execution security is only required to hold against post-quantum adversaries instead of information-theoretic adversaries. This is a meaningful relaxation as it allows us to build provably secure PVSS (with non-interactive dealer) and voting protocols that can be readily instantiated using post-quantum secure assumptions and highly optimized DL-based cryptography. As an application, we provide a long-lasting secure voting protocol.

1.2.3 Comparison of Secure Voting Protocols with Post-Quantum Vote Privacy. We compare our voting protocol with other voting protocols that provide post-quantum or everlasting privacy.

Post-Quantum Secure Voting: The state-of-the-art lattice-based voting scheme by [33] enforces the voting client to vote by secret sharing its vote, committing to it and encrypting the shares under each server's public key. The client also proves that the committed shares sum up to 0 or 1. Then each server decrypts its share and computes a valid NIZK proof that the committed share has been tallied correctly. This way, a client delegates the proof computation to the voting servers. Considering 4 servers, each of their server runs for 41 min to conduct a poll with two choices and participation from 11K voters. Meanwhile, our polling smart contract+server protocol performs the same task in a minute, yielding a 40× improvement. We note that this would further increase with the increase in the number of poll choices and the number of servers, as our cost grows sublinearly. The Epoque protocol [22] has similar performance as [33] since they use the same underlying NIZK proofs, and hence our improvement over Epoque would be similar.

Everlasting Secure Voting: There are also several everlasting secure voting protocols [45, 57] based on different properties that a voting scheme can satisfy. We do not delve into that for the sake of brevity. However, our long-lasting secure protocol provides similar performance to the state-of-the-art protocol of [57].

1.3 Technical Overview

A popular line of PVSS protocols with non-interactive dealers follow the commit–encrypt–prove paradigm: the dealer commits to a polynomial $f(x)$ with secret $s := f(0)$, encrypts each share $[\![s]\!]_i := f(i)$ under the receivers' public keys, and provides a NIZK proof that each encrypted $f(i)$ is an evaluation of the committed polynomial $f(x)$. Classical DL-based approaches [29, 37, 40, 55, 61] yield highly efficient schemes using group homomorphisms but are vulnerable to quantum attacks via Shor's algorithm [63]. Lattice-based PVSS schemes [4, 54] in this paradigm offer post-quantum security but rely on costly lattice-based NIZKs, making them computationally expensive.

Another line of work [5, 8, 51] explores dealing the secret using a bivariate polynomial so that the receivers error-correct the shares without dealer interaction. However, this approach incurs quadratic communication overhead for the dealer.

We explore a third paradigm [60] where the dealer neither incurs the computation overhead of an NIZK nor does it incur the quadratic communication overhead of sharing a bivariate polynomial. We observe that in the commit-encrypt-prove paradigm the NIZK proof can be broken down into two parts:

(i) The commitment to secret s and the commitment to the shares $[\![s]\!]_i$ lie on a committed t-degree polynomial, and
(ii) The committed shares are indeed the same as the encrypted values under each receiver's public key.

The first statement can be performed quite efficiently, e.g. using a non-interactive threshold ZK proofs [12] if the commitment is non-homomorphic, or using a SCRAPE test [26] if the commitment is additively homomorphic. The second statement is expensive to prove because the NIZK uses the encryption and commitment in a non-black-box fashion. Therefore, one key idea in our approach is to defer this verification to a later phase, which does not involve the dealer, and yet achieves public verifiability.

One-Time Public Key Encryption Approach. The high-level intuition is that the dealer encrypts the share and the decommitment and posts these encryptions and commitments on a public bulletin board or a blockchain, denoted by $\mathcal{F}_{\mathsf{BC}}$. The receivers read the commitments on $\mathcal{F}_{\mathsf{BC}}$ and check that the committed shares indeed lie on a t-degree polynomial. Then, each i-th receiver decrypts its share (and opening randomness) from the i-th encryption. If the decrypted values is not a correct opening to the i-th committed share, then the i-th receiver simply reveals their secret key on $\mathcal{F}_{\mathsf{BC}}$. Anyone can publicly decrypt the dealer-posted i-th encryption using the this decryption key, and verify the validity of the i-th receiver's claim. This avoids using NIZK for step-(ii): encrypted share and the committed share is the same.

Non-Interactive Key Exchange Approach. Leaking the receiver's long-term secret key is non-scalable, requiring fresh public-key setup for each incident and preventing key reuse across applications. Prior work like Eth-DKG [60] uses Diffie-Hellman key exchange: the dealer and each receiver maintain DH public keys, enabling pairwise secret key establishment for encrypting shares and opening randomness. For complaints, receivers reveal the pairwise secret

and provide efficient Chaum-Pedersen [28] NIZK proofs of correctness. Public complaint verification involves checking the NIZK proof, decrypting the share/randomness, and validating against the commitment. This approach is limited to discrete-log settings due to cheap NIZK proofs. Extension to lattice-based key exchange [38] is prohibitively expensive: efficient NIZK proofs for the required relations are unknown, and millisecond-level lattice-based decryption proofs remain unavailable.

Our Protocol. We circumvent the need for a lattice-based NIZK by replacing the public key encryption with an IBE [7, 36] – we use a lattice-based construction to obtain post-quantum security. In a one-time setup phase, each i-th receiver publishes an individual master IBE public key mpk_i for which they store a master secret key msk_i. The dealer commits to the shares and the secret, and encrypts the openings to the i-th shares under each mpk_i and the dealer's identity id_d. The dealer posts these commitments and encryptions, and then goes offline. The receivers check that the posted commitments commit to a degree t sharing of a polynomial. Then, each i-th receiver checks that the i-th encrypted values is indeed a valid decommitment to the i-th commitment – if not, then the receiver uses msk_i to extract dealer's decryption key $\mathsf{sk}_{i,\mathsf{id}_d}$ and post it on $\mathcal{F}_{\mathsf{BC}}$. It takes < 5ms to generate identity secret key. Anyone (even someone who is neither the dealer nor a receiver) can decrypt the i-th ciphertext using $\mathsf{sk}_{i,\mathsf{id}_d}$ and verify whether the receiver's claim is correct or not by matching the decrypted values with the commitment – this ensures public verifiability. Decrypting the IBE ciphertext takes $< 1ms$. This allows our dealer to deal a secret among 1000 receivers in < 1 sec while maintaining crucial properties such as public verifiability and non-interactive dealers. Concretely,

(Informal) Security Argument. Clearly, the only way a dealer deals a bad share without getting detected is if it breaks the binding of the commitment scheme; otherwise, it always gets caught. Meanwhile, the dealer's shares remain private due to the hiding of the commitment and the CPA-security of the encryption. We note that a malicious receiver cannot frame an honest dealer as corrupt by producing a bad secret key $\mathsf{sk}'_{i,\mathsf{id}_d}$ s.t. a valid ciphertext under $(\mathsf{mpk}_i, \mathsf{id}_d)$ decrypts to something else. This follows from the newly defined key verifiability property of the IBE scheme, which enforces that an adversarial receiver cannot produce a master public key mpk_i, s.t. it can later release a decryption key$\mathsf{sk}'_{i,\mathsf{id}_d}$ which verifies correctly, but leads to an incorrect decryption.

Concrete Instantiations. To complete our protocol, we need a PVSS scheme with a non-interactive dealer that allows verifying that the committed shares correspond to a t-degree sharing of a polynomial. Our post-quantum secure instantiation consists of the hash-based VSS protocol of [12]. In their PVSS, once the dealing is done, the dealer uses a distributed ZK proof (DZK) to prove that the committed shares lie on a t-degree polynomial, and in case, there is a complaint by a receiver the dealer broadcasts the opening to the receiver's share. We note that it suffices for us that the committed shares are partially verified as the IBE allows a receiver to prove later on that the dealer dealt bad shares, while removing the dealer interaction during the share verification phase. The DZK proofs are instantiated using hashes, modeled as a random oracle.

For our long-lasting secure instantiation, we use a variant of Pedersen PVSS [55] where the dealer commits to the secret and the shares using (homomorphic) Pedersen commitments. Due to homomorphism, the degree-check (part-(i) of the proof) is possible using a so-called SCRAPE test (described in Appendix. B.1). The rest of the protocol stays unaltered: the receiver outputs the identity secret key in case the encrypted share does not match the committed share. Finally, Pedersen's unconditional hiding along with the post-quantum CPA-security of the IBE scheme yield long-lasting security by ensuring the post-quantum privacy of the secret.

2 PRELIMINARIES

Notations. We use $\mathbb{N}$ to denote a set of positive integers, and $[n]$ to denote the set $\{1, \ldots, n\}$ for any $n \in \mathbb{N}$. We denote the security parameter by $\lambda \in \mathbb{N}$. The symbol $\perp$ denotes an undefined value, or invalidity. $\mathsf{negl}(\lambda)$ is a negligble function in security parameter λ. A set $X = \{x_1, \ldots, x_n\}$ is denoted as $x_{[n]}$ or $\{x_i\}_{i\in[n]}$. For any subset $S \subset [n]$, x_S or $\{x_i\}_{i\in S}$ denotes the subset of X containing all x_i's such that $i \in S$. A ordered tuple $(x_1, \ldots, x_n)$ is denoted by vector notation $\vec{x}_{[n]}$ or $(x_i)_{i\in[n]}$. Similarly for any subset S of $[n]$, $\vec{x}_S$ and $(x_i)_{i\in S}$ are defined. Every algorithm takes the computational security parameter λ and statistical security parameter μ as inputs. We use $y := D(x)$ to denote a deterministic computation and $y := x$ for assignment. Randomized computations are denoted as $y \leftarrow R(x)$ by running a randomized algorithm $R(\cdot)$ on input x. We discuss the SCRAPE test from [26] in Appendix. B.1. We denote indistinguishability against a PPT quantum adversary using $\overset{\mathsf{pq}}{\approx}$ and indistinguishability against a PPT classical adversary using $\overset{\mathsf{c}}{\approx}$ respectively.

Communication Channels. We assume that all the parties have read and write access to a public bulletin board or a blockchain [32], denoted as $\mathcal{F}_{\mathsf{BC}}$. For secure voting we additionally need smart contracts which can be formally modeled using [43]. For point-to-point communication we assume secure, authenticated channels between the parties.

Commitment Schemes. We denote a perfectly hiding and computationally binding commitment Com to message $m \in \mathcal{M}$ under randomness $r \in \mathcal{R}$ as $\langle m; r\rangle := \mathsf{Com}(m; r)$. We assume that Com is additively homomorphic, i.e. $\langle m_1; r_1\rangle \cdot \langle m_2; r_2\rangle = \langle m_1 + m_2; r_1 + r_2\rangle$ for all $m_1, m_2 \in \mathcal{M}$ and all $r_1, r_2 \in \mathcal{R}$. For concrete instantiation, we assume Com is Pedersen Commitment [55].

2.1 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme [20] consists of the following probabilistic polynomial-time algorithms. We also define an additional algorithm VerKey to verify a secret key $\mathsf{sk}_{\mathsf{ID}}$ corresponding to an identity ID.

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: Outputs a master public key mpk and a master secret key msk.
- $\mathsf{Extract}(\mathsf{msk}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$: Outputs the private key $\mathsf{sk}_{\mathsf{ID}}$ corresponding to identity ID.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, m) \to c$: Encrypts the message m as ciphertext c under master public key mpk and identity ID.
- $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) \to m$: Decrypts c to obtain message m.

- $\mathsf{VerKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) \to \{0,1\}$: Outputs 1 if $\mathsf{sk}_{\mathsf{ID}}$ is a valid secret key corresponding to ID, else outputs 0.

The IBE scheme needs to satisfy correctness and IND-ID-CPA Security [20]. In addition, we need a new property called key verifiability that allows checking whether a generated secret key $\mathsf{sk}_{\mathsf{ID}}$ is valid or not w.r.t. mpk and ID. We define it as.

Key Verifiability: An IBE scheme is said to satisfy key verifiability if for any security parameter λ, any identity ID, any message m, and any stateful PPT adversary $\mathcal{A}$, the following probability is negligible in λ.

$$\Pr\left[\begin{array}{c} m' := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) \notin \{m, \perp\}; \\ \mathsf{VerKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) = 1 \end{array} \middle| \begin{array}{c} \mathsf{mpk} \leftarrow \mathcal{A}(1^\lambda); \\ c \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, m); \\ \mathsf{sk}_{\mathsf{ID}} \leftarrow \mathcal{A}(c) \end{array}\right]$$

Informally, except with negligible probability, an adversary can not produce a master public key, such that it can later release a decryption key which verifies correctly, but leads to an incorrect decryption to a valid message $m' \neq m$. The Boneh-Franklin IBE scheme [20] and the lattice-based IBE scheme [36] satisfy key verifiability.

Post Quantum Secure IBE: Finally, we say that an IBE scheme is post-quantum secure if all of the above properties hold against an adversary with quantum capabilities.

We use a slightly modified version of the lattice-based IBE scheme of [36] for our protocol. It is based on the GPV scheme [42] over NTRU lattices, and it is post-quantum secure under the Ring-LWE [52] assumption. In the security proofs of our PVSS protocols, we also argue the adaptive security of the schemes. For that, we need the IBE encryptions to be non-committing [25], i.e., a simulated ciphertext can be later opened consistently as an encryption of any message. We note that the IBE in [36] is non-committing in the programmable random oracle model. We refer to Appendix. B.3 for more details.

2.2 Shamir's Secret Sharing Scheme

We recall the definition of an (n, t) Shamir secret sharing scheme [62] as follows:

Definition 2.1 (Shamir's Secret Sharing Scheme)**.** Let $\mathbb{F}$ be a finite field. It consists of two algorithms:

- $\mathsf{Share}(s, n, t) \to \{[\![s]\!]_i\}_{i\in[n]}$: Shares a secret s using a random t-degree polynomial to generate n shares $\{[\![s]\!]_i\}_{i\in[n]}$.
- $\mathsf{Recon}(z, \{i_j, [\![s]\!]_{i_j}\}_{j=1}^{t+1}) \to f(z)$: Given any $t+1$ shares, it outputs evaluation value $f(z)$ at point z.
- $\mathsf{ReconPoly}(\{i_j, [\![s]\!]_{i_j}\}_{j=1}^{t+1}) \to f(x)$: Given any $t+1$ shares, it outputs the polynomial $f(x)$.

Shamir sharing scheme is also defined over an additively homomorphic commitment scheme defined over message space $\mathcal{M} = \mathbb{F}$ and randomness space $\mathcal{R} = \mathbb{F}$. It consists of the following additional algorithms:

- $\mathsf{ShareCom}(n, t) \to (s, r, \{[\![s]\!]_i, [\![r]\!]_i, \langle [\![s]\!]_i; [\![r]\!]_i\rangle\}_{i\in[n]})$: Shares a random secret s and a randomness r using two t-degree polynomials to generate n shares $\{[\![s]\!]_i\}_{i\in[n]}, \{[\![r]\!]_i\}_{i\in[n]}$ and commitments to those shares as $\{\langle [\![s]\!]_i; [\![r]\!]_i\rangle\}_{i\in[n]}$.
- $\mathsf{ShareCom}(s, n, t) \to (r, \{[\![s]\!]_i, [\![r]\!]_i, \langle [\![s]\!]_i; [\![r]\!]_i\rangle\}_{i\in[n]})$: Shares secret s and a randomness r using two t-degree polynomials to generate n shares $\{[\![s]\!]_i\}_{i\in[n]}, \{[\![r]\!]_i\}_{i\in[n]}$ and commitments to those shares as $\{\langle [\![s]\!]_i; [\![r]\!]_i\rangle\}_{i\in[n]}$.
- $\mathsf{ReconCom}(z, \{i_j, \langle [\![s]\!]_{i_j}; [\![r]\!]_{i_j}\rangle\}_{j=1}^{t+1}) \to \langle s; r\rangle$: Given any $t+1$ shares, it reconstructs the committed secret $\langle s; r\rangle$.

The above scheme satisfies correctness and perfect secrecy. Full details in Appendix. B.4.

3 PUBLICLY VERIFIABLE SECRET SHARING WITH NON-INTERACTIVE DEALER

A (n, t) *Verifiable Secret Sharing (VSS)* [12, 31] consists of the following PPT algorithms:

(1) $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]})$: Given the security parameter λ as input, the public parameters are generated, and each party generates its own public and secret key pair. The public parameters and the public keys are shared among all the parties and they are implicit inputs to all the following algorithms.
(2) $\mathsf{Share}(s, n, t) \to (\{[\![s]\!]_i, e_i\}_{i\in[n]}, \pi)$: Given the number of parties n, corruption threshold t, and the secret s, the dealer computes shares $[\![s]\!]_i$ for the i-th party and the corresponding encryption under pk_i as e_i. The dealer also computes a proof π of correct sharing. The dealer outputs $(\{[\![s]\!]_i, e_i\}_{i\in[n]}, \pi)$.
(3) $\mathsf{Decrypt}(e_i, \mathsf{sk}_i) \to ([\![s]\!]_i, \mathsf{st}_i)$: Given the encrypted share e_i and secret key sk_i compute the decrypted share as $[\![s]\!]_i$ and a publicly verifiable decryption state st_i ($\in \{0,1\}^*$).
(4) $\mathsf{PubVer}(\mathsf{st}_i, e_i, \pi, n, t) \to \{0,1\}$: Given the public decryption state st_i any party can verify that the encrypted share e_i is invalid or not, corresponding to the proof π. If the verification succeeds, then PubVer outputs 1, else 0.
(5) $\mathsf{Recon}(\{i_j, [\![s]\!]_{i_j}\}_{j\in[t+1]}, \pi, n, t) \to s/\mathcal{L}$: Given any $t+1$ shares, the algorithm either reconstructs the secret s or outputs a list $\mathcal{L}$ of malformed shares.

Public Verifiability: A VSS is publicly verifiable if the transcript of the VSS protocol can be publicly verified by any external party to check that the dealing was valid/invalid. Given the decryption state st_i any external party can publicly verify, via PubVer algorithm, whether the encrypted share e_i is valid or not, given the proof π.

Definition 3.1 (Computationally Secure PVSS). A publicly verifiable VSS scheme (PVSS) is computationally secure if the dealer and receiver and all adversaries are computationally bounded algorithms (PPT) and the scheme satisfies the following properties:

- *Correctness:* Any set of $t+1$ or more valid shares correctly reconstructs the secret s.
- *Termination:* All honest nodes will eventually terminate the Share, Decrypt and PubVer phase.
- *Reconstructability:* If a dealing verifies successfully, then, given any set of $\geq t+1$ shares, the Recon algorithm either reconstructs a unique s or identifies a non-empty list $\mathcal{L} \neq \emptyset$ of malformed shares even if the dealer is adversarial.
- *Computational Secrecy:* Assuming an honest dealer dealing secret s, then any set of $\leq t$ shares, $\leq t$ secret keys, n states $\{\mathsf{st}_i\}_{i\in[n]}$, n encrypted shares and π hides s against an adversary, i.e. for any two secrets s, s' following holds:

$$\{s, \{i_j, [\![s]\!]_{i_j}, \mathsf{sk}_i\}_{j\in[t]}, \{e_i, \mathsf{pk}_i\}_{i\in[n]}, \pi\} \overset{c}{\approx}$$

$$\{s', \{i_j, [\![s]\!]_{i_j}, \mathsf{sk}_i\}_{j\in[t]}, \{e_i, \mathsf{pk}_i\}_{i\in[n]}, \pi\},$$
where the $\{i_j\}_{j\in[t]}$ indices are chosen by the adversary.

Post-Quantum Secure PVSS: We say that a PVSS scheme achieves post-quantum security if all the properties hold even against quantum PPT adversaries.
Long-lasting Secure PVSS: We say that a PVSS scheme achieves long-lasting security if all properties except computational secrecy holds against classical PPT adversaries, but the secrecy holds against quantum PPT adversaries.
Non-interactive Dealers: A PVSS is said to have non-interactive dealers if the dealer non-interactively performs the sharing by locally producing its message and sending it over and then each receiver can verify that their share $[\![s]\!]_i$, obtained via Decrypt, is valid without further interaction with the dealer.

4 A NEW SECURITY MODEL: LONG-LASTING SECURITY

A protocol is said to be long-lasting secure if it is secure against a PPT classical adversary *during* the protocol execution. And after the protocol terminates, the protocol transcript would remain secure (does not leak any information about honest parties' inputs) against a future PPT quantum adversary. We formally define the notion of long-lasting security following the simulation-based real-ideal world paradigm of [24] in Appendix. C, Definition. C.1. We note that our PVSS protocol satisfies game-based definition and so we do not need the simulation-based definition for this paper. However, we believe this model can be interesting independently for building schemes that are to be deployed today, but require long-term privacy, and can be used to model current elliptic-curve+lattice based protocols (like Zama's FHE-based protocols) that are perfectly hiding and hence provide secrecy in the long run against quantum adversaries.

5 OUR POST-QUANTUM SECURE PVSS SCHEME

We build our PVSS scheme on top of the distributed ZK proof and hash-based PVSS of [12]. In the PVSS protocol of [12], the dealer shares its secret s and commits to it and its shares using (hash-based) non-homomorphic commitments. Then, the dealer computes a distributed ZK proof (DZK) using a Schnorr-like proof on committed polynomials to prove that the sharing lies on a t-degree polynomial. The dealer sends the shares and the proofs to each receiver. Upon receiving its share and part of the ZK proof, each receiver verifies its local share, and raises a complaint if its ZK proof does not verify. In case of a complaint, then the dealer leaks the receiver's share, and anyone can verify the legitimacy of the claim. Their PVSS protocol requires the dealer to interact to resolve the receivers' complaints. The overview can be found in Section 1.3.

In our PVSS, the dealer encrypts the receiver's state under the receiver's master public key and the dealer's identity. To raise a complaint, the receiver produces the identity secret key, and anyone can decrypt the receiver's state(encrypted by the dealer) and verify the legitimacy of the claim. This removes the dealer interaction from the PVSS protocol.

We present the formal protocol details below. We assume that (KeyGen, Extract, Enc, Dec) is a post-quantum secure IBE scheme, H is a random oracle, and (Share, Recon, ReconPoly) is a Shamir secret sharing scheme. Security is summarized as follows.

$\underline{\mathsf{Setup}(1^\lambda)}$:

- For $i \in [n]$, each i-th receiver runs $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$ to obtain master public-secret keys.
- All the n master public keys - $(\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n)$ are posted on the bulletin board $\mathcal{F}_{\mathsf{BC}}$.

$\underline{\mathsf{Share}(s \in \mathbb{F}, n, t)}$: The dealer performs the following:

(1) *Dealing:* Generates shares of s using t-degree polynomial $f(x)$ s.t. $f(0) := s$. The dealer also shares a commitment randomness r.

$\{[\![s]\!]_i\}_{i\in[n]} := \mathsf{Share}(s, n, t), \{[\![r]\!]_i\}_{i\in[n]} := \mathsf{Share}(n, t)$

Commit the share as $\mathsf{csh}_i := \mathrm{H}([\![s]\!]_i, [\![r]\!]_i)$ for $i \in [n]$.

(2) *First message of Distributed-ZK (DZK):* Sample a random t-degree padding polynomial $B(x)$ with coefficients in the field $\mathbb{F}$. For each $i \in [n]$: sample randomness $q_i \leftarrow \{0,1\}^\lambda$. Compute commitments to $B(i)$ as
$$\mathsf{cpt}_i := \mathrm{H}(B(i), q_i) \text{ for } i \in [n].$$
(3) *Challenge of DZK:* Set $\mathsf{ch} := \mathrm{H}(\{\mathsf{csh}_i, \mathsf{cpt}_i\}_{i\in[n]})$.
(4) *Opening of DZK:* Set $\mathsf{c} := W(x) := B(x) - \mathsf{ch} \cdot f(x)$.
(5) *Encrypting Receivers' States:* For $i \in [n]$, encrypt the i-th receiver's state under $\mathsf{ID}_i := (\mathsf{Dealer}, i)$ as:
$$\mathsf{E}_i = \mathsf{Enc}(\mathsf{mpk}_i, \mathsf{ID}_i, ([\![s]\!]_i, [\![r]\!]_i, q_i)).$$
(6) Set PVSS proof $\pi := (\mathsf{c}, \{\mathsf{csh}_i, \mathsf{cpt}_i\}_{i\in[n]})$. Posts encryptions $\{\mathsf{E}_i\}_{i\in[n]}$ and proof π on $\mathcal{F}_{\mathsf{BC}}$.

$\underline{\mathsf{Decrypt}(\mathsf{E}_i, \mathsf{msk}_i)}$: Each i-th receiver runs the following:

(1) Decrypts its shares and randomness values as $([\![s]\!]_i, [\![r]\!]_i, q_i) := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}_i}, \mathsf{E}_i)$ where $\mathsf{sk}_{\mathsf{ID}_i} := \mathsf{Extract}(\mathsf{msk}_i, (\mathsf{Dealer}, i))$.
(2) Set $\mathsf{st}_i := (\textsc{Complaint}, i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}_i})$ if
- The committed share does not decommit correctly, i.e. $\mathrm{H}([\![s]\!]_i, [\![r]\!]_i) \neq \mathsf{csh}_i$, OR
- The DZK failed to verify for i-th receiver, i.e. $\mathrm{H}(W(i)+\mathsf{ch}\cdot[\![s]\!]_i, q_i) \neq \mathsf{cpt}_i$ where $\mathsf{ch} := \mathrm{H}(\{\mathsf{csh}_i, \mathsf{cpt}_i\}_{i\in[n]})$.

Otherwise, set $\mathsf{st}_i := \emptyset$.
(3) Post st_i on $\mathcal{F}_{\mathsf{BC}}$, and output $([\![s]\!]_i, [\![r]\!]_i, q_i)$ and st_i.

$\underline{\mathsf{PubVer}(\mathsf{st}_i, \mathsf{E}_i, \pi, n, t)}$: Any party can run this as follows:

(1) Parse PVSS proof $(\mathsf{c}, \{\mathsf{csh}_i, \mathsf{cpt}_i\}_{i\in[n]}) := \pi$.
(2) If $W(x) := \mathsf{c}$ is not a t-degree polynomial then output 0 and discard the dealings.
(3) If there are no complaints, i.e. $\mathsf{st}_i == \emptyset$ for all $i \in [n]$, then output 1. Else, continue to the next step.
(4) For all complaints of the form $(\textsc{Complaint}, i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}_i})$ that are posted on $\mathcal{F}_{\mathsf{BC}}$: Reject the complaint if
$$(\mathsf{VerKey}(\mathsf{mpk}_i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}_i}) == 0) \vee (\mathrm{H}([\![s]\!]_i, [\![r]\!]_i) == \mathsf{csh}_i \;\&\; \mathrm{H}(W(i) + \mathsf{ch} \cdot [\![s]\!]_i, q_i) == \mathsf{cpt}_i),$$

where $(\llbracket s \rrbracket_i, \llbracket r \rrbracket_i, q_i) := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}_i}, \mathsf{E}_i)$ and $\mathsf{ch} := \mathrm{H}(\{\mathsf{csh}_i, \mathsf{cpt}_i\}_{i\in[n]})$. Output 0 if the complaint is valid.

If $W(x)$ is an t-degree polynomial and none of the raised complaints were valid, then output 1.

$\underline{\mathsf{Recon}}(\{i_j, \llbracket s \rrbracket_{i_j}, \llbracket r \rrbracket_{i_j}, q_{i_j}\}_{j\in[t+1]}, \pi, n, t)$:

(1) Initialize list $\mathcal{L} := \emptyset$ for storing malformed shares.
(2) *Output Share Verification:* Given any share $\{i_j, \llbracket s \rrbracket_{i_j}, \llbracket r \rrbracket_{i_j}, q_{i_j}\}_{j\in[t+1]}$ if $\mathsf{cpt}_{i_j} \neq \mathrm{H}(W(i_j) + \mathsf{ch} \cdot \llbracket s \rrbracket_{i_j}, q_{i_j}) \vee \mathsf{csh}_{i_j} \neq \mathrm{H}(\llbracket s \rrbracket_{i_j}, \llbracket r \rrbracket_{i_j})$, where $\mathsf{ch} := \mathrm{H}(\{\mathsf{csh}_i, \mathsf{cpt}_i\}_{i\in[n]})$, then consider it as invalid share and add i_jth share to discard list as $\mathcal{L} := \mathcal{L} \cup \{i_j\}$.
(3) Given $t+1$ valid shares that pass the above check, reconstruct $f(x)$ and $r(x)$ polynomials as

$$f(x) := \mathsf{ReconPoly}(\{i_j, \llbracket s \rrbracket_{i_j}\}_{j\in[t+1]}),$$
$$r(x) := \mathsf{ReconPoly}(\{i_j, \llbracket r \rrbracket_{i_j}\}_{j\in[t+1]}).$$

Output $s := f(0)$ as the secret, $f(x)$ as the secret polynomial and $r(x)$ as the randomness polynomial corresponding to the commitments $\mathsf{csh}_i = \mathrm{H}(f(i), r(i))$ for $i \in [n]$.

Theorem 5.1. *Assuming that* (KeyGen, Extract, Enc, Dec) *is a post-quantum secure IND-ID-CPA-Secure IBE scheme with key verifiability, H is a random oracle and* (Share, Recon, ReconPoly) *is a Shamir secret sharing scheme (Sec.2.2), then the above protocol is a post-quantum secure PVSS (Sec. 3.1) with non-interactive dealer and n receivers against a static adversary corrupting up to $t < \frac{n}{2}$ parties in the classical random oracle and $\mathcal{F}_{BC}$ model with synchronous communication between parties, yielding $\frac{Q_H}{|\mathbb{F}|} + \epsilon^{\mathsf{Com}}$ soundness error over a field $\mathbb{F} \geq 2^{128}$), Q_H is the number of random oracle queries by the adversary and ϵ^{Com} is the advantage of the* Com *adversary. Moreover, for security against adaptive corruption (instead of static corruption), we require non-committing property from the IBE scheme and programmability from H.*

We prove that the above PVSS protocol is post-quantum secure by proving Thm. 5.1 in Appendix. D.1, and we also briefly discuss adaptive security of our PVSS protocol. We note that our protocol is secure up to $t < n$ corruptions, but for liveness we need $t < \frac{n}{2}$.

6 LONG-LASTING SECURE PVSS CONSTRUCTION AND POLLING

Our post-quantum secure PVSS protocol did not provide an aggregatable transcript since the commitments csh and cpt are instantiated using hash-based PQ-secure commitments. This limits its usage in different aggregation protocols like secure voting, statistics computation, and so on, where we need these commitments to be aggregated. One could consider replacing the hashes with an additively homomorphic commitment, but that would require $2n$ commitments in proof π. We instead reduce that to n commitments using a more optimized PVSS protocol that has aggregatable transcripts. For that, we relax the security to long-lasting, as it suffices for our long-lasting secure voting protocol.

6.1 Long-lasting Secure PVSS

Our long-lasting PVSS protocol uses a computationally binding and perfectly hiding additively homomorphic commitment scheme Com, instantiated using Pedersen Commitments. In addition, it uses the IBE scheme and a Shamir sharing scheme - (Share, Recon, ShareCom, ReconCom), defined (in Sec. 2.2) over commitment Com. Our protocol uses a SCRAPE test [26] over committed shares to publicly check that the underlying committed polynomial is t-degree. The previous PVSS protocol did this using the distributed ZK proof. We provide the details of the SCRAPE test (denoted as $\mathcal{F}_{\mathsf{SCRAPE}}$) in Appendix. B.1 for completeness. The high-level overview of our PVSS appears in Sec. 1.3, and the protocol details follow.

$\underline{\mathsf{Setup}}(1^\lambda)$:

- For $i \in [n]$, each i-th receiver runs $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$ to obtain master public-secret keys.
- All the n master public keys - $(\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n)$ are posted on the bulletin board $\mathcal{F}_{\mathsf{BC}}$.

$\underline{\mathsf{Share}}(s, n, t)$: The dealer performs the following:

(1) *Dealing:* Generates committed shares of secret s using randomness r.

$$(s, r, \{\llbracket s \rrbracket_i, \llbracket r \rrbracket_i, \langle \llbracket s \rrbracket_i; \llbracket r \rrbracket_i \rangle\}_{i\in[n]}) := \mathsf{ShareCom}(s, n, t)$$

For $i \in [n]$, denote the committed shares as

$$\mathsf{csh}_i := \langle \llbracket s \rrbracket_i, \llbracket r \rrbracket_i \rangle.$$

Denote the committed secret as $\mathsf{c} := \langle s; r \rangle$.
(2) *Encrypting Receivers' States:* For $i \in [n]$, encrypt i-th receiver's share under $\mathsf{ID}_i := (\mathsf{Dealer}, i)$ as E_i:

$$\mathsf{E}_i = \mathsf{Enc}(\mathsf{mpk}_i, \mathsf{ID}_i, (\llbracket s \rrbracket_i, \llbracket r \rrbracket_i)).$$

(3) Sets PVSS proof $\pi := \{\mathsf{c}, \{\mathsf{csh}_i\}_{i\in[n]}\}$.
(4) The dealer posts encryptions $\{\mathsf{E}_i\}_{i\in[n]}$ and proof π on $\mathcal{F}_{\mathsf{BC}}$.

$\underline{\mathsf{Decrypt}}(\mathsf{E}_i, \mathsf{msk}_i)$: Each i-th receiver runs the following:

(1) Decrypt shares as $(\llbracket s \rrbracket_i, \llbracket r \rrbracket_i) := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}_i}, \mathsf{E}_i)$ where $\mathsf{sk}_{\mathsf{ID}_i} := \mathsf{Extract}(\mathsf{msk}_i, (\mathsf{Dealer}, i))$.
(2) Set $\mathsf{st}_i := (\textsc{Complaint}, i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}_i})$ if $\mathsf{Com}(\llbracket s \rrbracket_i; \llbracket r \rrbracket_i) \neq \mathsf{csh}_i$. Otherwise, set $\mathsf{st}_i := \emptyset$.
(3) Post st_i on $\mathcal{F}_{\mathsf{BC}}$, and output $(\llbracket s \rrbracket_i, \llbracket r \rrbracket_i)$ and st_i.

$\underline{\mathsf{PubVer}}(\mathsf{st}_i, \mathsf{E}_i, \pi, n, t)$: Any party can run this as:

(1) Parse PVSS proof as $\{\mathsf{c}, \{\mathsf{csh}_i\}_{i\in[n]}\} := \pi$.
(2) If $\mathcal{F}_{\mathsf{SCRAPE}}((n, t), \mathsf{c}, \mathsf{csh}_1, \ldots, \mathsf{csh}_n) == 0$ then output 0 and discard the dealings.
(3) If there are no complaints, i.e. $\mathsf{st}_i == \emptyset$ for all $i \in [n]$, then output 1. Else, continue to the next step.
(4) For all complaints (posted on $\mathcal{F}_{\mathsf{BC}}$) of the form $(i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}_i})$: Reject the complaint if

$$(\mathsf{VerKey}(\mathsf{mpk}_i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}_i}) == 0) \vee$$
$$(\mathsf{Com}(\llbracket s \rrbracket_i; \llbracket r \rrbracket_i) == \mathsf{csh}_i),$$

where $(\llbracket s \rrbracket_i, \llbracket r \rrbracket_i) := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}_i}, \mathsf{E}_i)$. If the complaint is valid, then output 0.

If the SCRAPE test passed and none of the raised complaints were valid then output 1.

$\mathsf{Recon}(\{i_j, [\![s]\!]_{i_j}, [\![r]\!]_{i_j}\}_{j\in[t+1]}, \pi, n, t)$:

(1) Initialize list $\mathcal{L} := \emptyset$ for storing malformed shares.
(2) *Output Share Verification:* Given any $t+1$ shares of the form $\{i_j, [\![s]\!]_{i_j}, [\![r]\!]_{i_j}\}_{j\in[t+1]}$ if $\mathsf{csh}_{i_j} \neq \mathsf{Com}([\![s]\!]_{i_j}; [\![r]\!]_{i_j})$ then consider it as invalid share and add i_jth share to discard list as $\mathcal{L} := \mathcal{L} \cup \{i_j\}$.
(3) Given $t+1$ valid shares that pass the above check, reconstruct the secret s and randomness r as

$$s := \mathsf{Recon}(0, \{i_j, [\![s]\!]_{i_j}\}_{j\in[t+1]}),$$

$$r := \mathsf{Recon}(0, \{i_j, [\![r]\!]_{i_j}\}_{j\in[t+1]}).$$

Output s as the secret and r as the randomness corresponding to commitment $\mathsf{c} = \mathsf{Com}(s; r)$.

Theorem 6.1. *Assuming that* Com *is a computationally binding and perfectly hiding additively homomorphic commitment scheme,* (KeyGen, Extract, Enc, Dec) *is a post quantum secure IND-ID-CPA-Secure IBE scheme with key verifiability and* (Share, Recon, ShareCom, ReconCom) *is a Shamir secret sharing scheme (defined in Sec. 2.2) over commitment* Com*, then the above protocol is a long-lasting secure PVSS (Def. 3.1) with non-interactive dealer and n receivers against a static adversary corrupting up to $t < \frac{n}{2}$ parties in the classical random oracle and $\mathcal{F}_{BC}$ model with synchronous communication between parties. Moreover, for security against adaptive corruption (instead of static corruption), we require non-committing property from the IBE scheme and programmability from H.*

We prove that the above PVSS protocol is long-lasting secure by proving Thm. 6.1 in Appendix. D.2, and we also briefly discuss adaptive security of our PVSS protocol. We note that our protocol is secure up to $t < n$ corruptions, but for liveness we need $t < \frac{n}{2}$.

6.2 Long-lasting Secure Polling

We present our polling protocol in this section. For polling, we use our long-lasting PVSS protocol based on additive homomorphic commitments since the PVSS transcript is aggregatable and it allows us to generate additional proofs over the commitments very efficiently without sacrificing computational performance. The hash-based PVSS protocol does not allow aggregation of the commitment c, and so it cannot be used in polling since c will consist of the committed votes and needs to satisfy additive homomorphism. We assume that Com is a computationally binding and perfectly hiding additively homomorphic commitment scheme, and (Share, Recon, ShareCom, ReconCom) is a Shamir secret sharing scheme defined over Com. We assume that all the parties have read and write access to $\mathcal{F}_{BC}$. Our polling protocol consists of the following phases.

Client-Side Setup: Each client C is identified by its public verification key vk and possesses a corresponding secret key used to sign transactions. vk is posted on $\mathcal{F}_{BC}$.

Server-Side Setup: There are n servers - $(\mathsf{S}_1, \ldots, \mathsf{S}_n)$, out of which $t < n$ of them can be corrupt. Each server S_i has an IBE master public key and master secret pair $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{Setup}(1^\lambda)$. All the n master public keys are posted on $\mathcal{F}_{BC}$ as $\{\mathsf{S}_i, \mathsf{mpk}_i\}_{i\in[n]}$.

Setting up a Poll: To set up a poll with identifier pid, the organizer specifies the question, options (assume four for simplicity), and an expiry timestamp. The pid and expiry are posted on the smart contract, and once expiry is reached, servers compute the results.

Voting by a Client: Each poll client is allowed to vote for a single option in an active poll. Each poll is uniquely identified by a poll identifier pid. Each client is identified by verification key vk, and each server is uniquely identified by S_i. Each identity $\mathsf{ID}_i := (\mathsf{vk}, \mathsf{pid}, \mathsf{S}_i)$ uniquely identifies the client's vote dealing for a server S_i in a poll, identified as pid. To vote in a poll the client performs the following:

(1) Denote the vote vector as $(\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3, \mathsf{v}_4)$ where $\mathsf{v}_j := 1$ if the client votes for jth option and $\mathsf{v}_{j'} := 0$ for all $j' \in [4] \setminus j$. For e.g., if the client votes for the second option then $(\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3, \mathsf{v}_4) := (0, 1, 0, 0)$.
(2) Compute the committed server shares of each value v_j, for $j \in [4]$, as:
   (a) The client generates committed shares of v_j as $(\mathsf{v}_j, \mathsf{r}_j, \{[\![\mathsf{v}_j]\!]_i, [\![\mathsf{r}_j]\!]_i, \langle [\![\mathsf{v}_j]\!]_i; [\![\mathsf{r}_j]\!]_i\rangle\}_{i\in[n]}) := \mathsf{ShareCom}(\mathsf{v}_j, n, t)$.
   (b) Denote the committed value as $\mathsf{c}_j := \langle \mathsf{v}_j, \mathsf{r}_j\rangle := \langle [\![\mathsf{v}_j]\!]_0, [\![\mathsf{r}_j]\!]_0\rangle$.
   (c) Denote the committed shares as $\mathsf{csh}_{j,i} := \langle [\![\mathsf{v}_j]\!]_i; [\![\mathsf{r}_j]\!]_i\rangle$ for each server S_i.
(3) For $i \in [n]$, encrypt the i-th server's shares under $\mathsf{ID}_i := (\mathsf{vk}, \mathsf{pid}, \mathsf{S}_i)$ as $\mathsf{E}_i := \mathsf{E}_i = \mathsf{Enc}(\mathsf{mpk}_i, \mathsf{ID}_i, \mathsf{msg}_i)$, where $\mathsf{msg}_i := ([\![\mathsf{v}_1]\!]_i, [\![\mathsf{r}_1]\!]_i, [\![\mathsf{v}_2]\!]_i, [\![\mathsf{r}_2]\!]_i, [\![\mathsf{v}_3]\!]_i, [\![\mathsf{r}_3]\!]_i, [\![\mathsf{v}_4]\!]_i, [\![\mathsf{r}_4]\!]_i)$.
(4) Posts $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_4, \{\mathsf{csh}_{1,i}, \mathsf{csh}_{2,i}, \mathsf{csh}_{3,i}, \mathsf{csh}_{4,i}, \mathsf{E}_i\}_{i\in[n]})$ on $\mathcal{F}_{BC}$.
(5) Proves that it has voted for a single option by proving that each committed vote is a bit, and they sum up to 1. The client computes the NIZK proof π_{vote} (Sec. E.1) as

$$\pi_{\mathsf{vote}} = \mathsf{Prove}_{\mathsf{vote}}((\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_4, \mathsf{vk}, \mathsf{pid}), (\{\mathsf{v}_j, \mathsf{r}_j\}_{j\in[4]})),$$

$\mathsf{Prove}_{\mathsf{vote}}$ proves that $\forall j \in [4], (\mathsf{c}_j = \langle \mathsf{v}_j; \mathsf{r}_j\rangle) \;\&\; (\mathsf{c}_j = \mathsf{Com}(0) \vee \mathsf{c}_j = \mathsf{Com}(1)) \;\&\; \mathsf{c}_1 + \mathsf{c}_2 + \mathsf{c}_3 + \mathsf{c}_4 = \mathsf{Com}(1)$.
(6) Posts the following on $\mathcal{F}_{BC}$ as a transaction and signs it using its secret key sk.
   (a) Verification key vk and poll identifier pid.
   (b) Commitments to the vote values as $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_4)$,
   (c) Commitments to the servers' shares to the vote values as $\{\mathsf{csh}_{1,i}, \mathsf{csh}_{2,i}, \mathsf{csh}_{3,i}, \mathsf{csh}_{4,i}\}_{i\in[n]}$.
   (d) Proof π_{vote} of correct voting.
   (e) IBE encryptions to the servers' shares as $\{\mathsf{E}_i\}_{i\in[n]}$.

Note: The client's verification key vk and pid values are included in the proving algorithm for the NIZKs so that the clients encrypted votes cannot be replicated by another client and the vote is bound to (vk, pid).

Vote Verification by Smart Contract: The smart contract for pid verifies each client vote as follows:

- Verify that the client has not previously voted in pid by checking that there does exist a vote that starts with (vk, pid, ...).
- Verify that the poll is active, by checking that the expiry timestamp of pid has not been reached.
- Verify that π_{vote} is valid (Sec. E.1) by running $\mathsf{Verify}_{\mathsf{vote}}((\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_4, \mathsf{vk}, \mathsf{pid}), \pi_{\mathsf{vote}}) == 1$.

If any of the above verification algorithms return 0, then mark this vote as invalid. Else, append the client's transaction on $\mathcal{F}_{\mathsf{BC}}$.

After a client vote is posted on the $\mathcal{F}_{\mathsf{BC}}$, the servers proceed to validate it and store the validated vote locally.

Client's Vote Validation: Each server S_i reads the client transaction, identified by $(\mathsf{vk}, \mathsf{pid}, \ldots)$, from $\mathcal{F}_{\mathsf{BC}}$ and then performs the following checks:

(1) If any of the SCRAPE tests (Appendix. B.1) fail then discard the client, i.e. the following holds - $\exists j \in [4]$ s.t.
$$\mathcal{F}_{\mathsf{SCRAPE}}((n,t), \mathsf{c}_j, \mathsf{csh}_{j,1}, \ldots, \mathsf{csh}_{j,n}) = 0.$$
(2) Extract the decryption key corresponding to $\mathsf{ID} := (\mathsf{vk}, \mathsf{pid}, \mathsf{S}_i)$ as $\mathsf{sk}_{\mathsf{ID}} := \mathsf{Extract}(\mathsf{msk}_i, \mathsf{ID})$.
(3) Decrypt the server's shares as $\mathsf{msg}_i := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{E}_i)$ from E_i and parse $([\![\mathsf{v}_1]\!]_i, [\![\mathsf{r}_1]\!]_i, [\![\mathsf{v}_2]\!]_i, [\![\mathsf{r}_2]\!]_i, [\![\mathsf{v}_3]\!]_i, [\![\mathsf{r}_3]\!]_i, [\![\mathsf{v}_4]\!]_i, [\![\mathsf{r}_4]\!]_i) := \mathsf{msg}_i$.
(4) *Complaint Phase:* If any of the shares $[\![\mathsf{v}_j]\!]_i, [\![\mathsf{r}_j]\!]_i$ (for $j \in [4]$) fail to match with commitment $\mathsf{csh}_{j,i}$, i.e. $\mathsf{csh}_{j,i} \neq \mathsf{Com}([\![\mathsf{v}_j]\!]_i; [\![\mathsf{r}_j]\!]_i)$ for some $j \in [4]$, then post a complaint on $\mathcal{F}_{\mathsf{BC}}$ as $(\mathsf{Complaint}, \mathsf{vk}, \mathsf{pid}, \mathsf{S}_i, j, \mathsf{sk}_{\mathsf{ID}})$ where $\mathsf{ID} := (\mathsf{vk}, \mathsf{pid}, \mathsf{S}_i)$.
(5) *Resolution Phase:* If the client passed the above checks for S_i then S_i resolves the complaints raised against the client by other servers on $\mathcal{F}_{\mathsf{BC}}$ as follows:
   (a) Parse a complaint by S_k as $(\mathsf{Complaint}, \mathsf{vk}, \mathsf{pid}, \mathsf{S}_k, j, \mathsf{sk}_{\mathsf{ID}})$ where $\mathsf{ID} := (\mathsf{vk}, \mathsf{pid}, \mathsf{S}_k)$.
   (b) Read $(\mathsf{E}_k, \mathsf{csh}_{j,k})$ from $\mathcal{F}_{\mathsf{BC}}$ and decrypt the server's shares as $\mathsf{msg}_k := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{E}_k)$ from E_k and parse $([\![\mathsf{v}_1]\!]_k, [\![\mathsf{r}_1]\!]_k, [\![\mathsf{v}_2]\!]_k, [\![\mathsf{r}_2]\!]_k, [\![\mathsf{v}_3]\!]_k, [\![\mathsf{r}_3]\!]_k, [\![\mathsf{v}_4]\!]_k, [\![\mathsf{r}_4]\!]_k) := \mathsf{msg}_k$.
   (c) Verify the server provided secret key $\mathsf{sk}_{\mathsf{ID}}$ for ID. If $\mathsf{VerKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) = 0$, then discard S_k.
   (d) Discard the client if $\mathsf{csh}_{j,k} \neq \mathsf{Com}([\![\mathsf{v}_j]\!]_k; [\![\mathsf{r}_j]\!]_k)$, else discard S_k.
(6) If the client passes the above checks then consider the vote as server-validated.

Once the expiry of a poll is reached, the voting phase has ended. The servers then proceed to perform vote aggregation and output computation.

Vote Aggregation For all the valid clients, each server S_i sums up their own shares as

$$[\![\mathsf{out}_j]\!]_i := \sum_{\text{Valid clients}} [\![\mathsf{v}_j]\!]_i, [\![\mathsf{rout}_j]\!]_i := \sum_{\text{Valid clients}} [\![\mathsf{r}_j]\!]_i \text{ for } j \in [4]$$

Compute the commitments to the output as

$$\mathsf{cout}_j := \prod_{\text{Validated clients}} \mathsf{c}_j \text{ for } j \in [4],$$

where c_j was the client's vote for option j.

Output Computation: The servers reconstruct output as:

(1) Each server S_i posts its share of the output - $(\{[\![\mathsf{out}_j]\!]_i, [\![\mathsf{rout}_j]\!]_i\}_{j\in[4]})$ on $\mathcal{F}_{\mathsf{BC}}$.
(2) Upon receiving shares from $t+1$ servers in set S perform the following check for each $j \in [4]$:
$$\mathsf{cout}_j == \mathsf{Com}(\mathsf{out}_j; \mathsf{rout}_j), \text{ where}$$
$$\mathsf{out}_j := \mathsf{Recon}(0, \{S_k, [\![\mathsf{out}_j]\!]_k\}_{k\in S}), \text{ and}$$
$$\mathsf{rout}_j := \mathsf{Recon}(0, \{S_k, [\![\mathsf{rout}_j]\!]_k\}_{k\in S}).$$
If the check passes then proceed to next step. Otherwise, all the servers computes the commitments to the share (of the output) of every other server as:
$$\mathsf{coutsh}_{j,i} := \prod_{\text{Validated clients}} \mathsf{csh}_{j,i} \text{ for } j \in [4], i \in [n].$$
Let $B := \emptyset$ denote the set of all the undiscarded servers. Add $B := B \cup \{i\}$ if the share of S_i matches with its commitment for all $j \in [4]$, i.e.
$$\mathsf{Com}([\![\mathsf{out}_j]\!]_i; [\![\mathsf{rout}_j]\!]_i) == \mathsf{coutsh}_{j,i}.$$
The servers wait (until $|B| = t+1$) for shares from other servers and repeat the above step. Once $|B| = t+1$ valid shares are identified, reconstruct the outputs $(\mathsf{out}_1, \mathsf{out}_2, \mathsf{out}_3, \mathsf{out}_4)$ as:
$$\mathsf{out}_j := \mathsf{Recon}(0, \{B_k, [\![\mathsf{out}_j]\!]_k\}_{k\in[B]}) \text{ for } j \in [4],$$
Compute the opening randomnesses as
$$\mathsf{rout}_j := \mathsf{Recon}(0, \{B_k, [\![\mathsf{rout}_j]\!]_k\}_{k\in[B]}) \text{ for } j \in [4].$$
(3) The servers post $(\mathsf{pid}, \{\mathsf{out}_j, \mathsf{rout}_j, \mathsf{cout}_j\}_{j\in[4]})$ as the output on $\mathcal{F}_{\mathsf{BC}}$.

Private Reconstruction. In the case of private reconstruction towards a specific party P, the servers send their shares to party P. Then party P locally discards the servers whose shares do not match with the commitments. Finally P reconstructs the output by using the undiscarded server shares, and matches it against the publicly computed commitments.

For the sake of brevity, we present the technical details of our NIZK protocols in Section E. Next, we discuss the security properties guaranteed by our above protocol.

(1) **Voter Privacy:** The individual votes cast by a client always remain private against a quantum adversary. The vote values are hidden in due to IND-ID-CPA security of the IBE encryption scheme Enc, the perfectly hiding property of Com and perfect zero-knowledge property of the NIZKs.
(2) **Censorship Resistance:** A client's vote is guaranteed to be included and verifiable via $\mathcal{F}_{\mathsf{BC}}$. Votes can be censored if a bad server proves a mismatch between an encrypted share and the commitment, which occurs with negligible probability due to IBE's key verifiability.
(3) **Public Verifiability:** All commitments and NIZKs are posted on $\mathcal{F}_{\mathsf{BC}}$, allowing any external party to compute output commitments and verify openings and proofs, ensuring public verifiability of the voting process.
(4) **Accountability:** The protocol records all actions on $\mathcal{F}_{\mathsf{BC}}$, enabling publicly detecting and identifying any misbehaving parties.

7 EVALUATION

We implement the post-quantum and long-lasting PVSS schemes, outlined in Sections 5 and 6.1 respectively. Moreover, we use the long-lasting PVSS scheme to build a voting application, as detailed in Section 6.2. Our implementation can be found in [59].

Setup: All protocols are implemented in Rust. Elliptic-curve operations (commitments, proofs, and group arithmetic) use curve25519-dalek on Ristretto255 [1]. For identity-based encryption (IBE), we use the construction of Ducas et al. [36], which works over NTRU

Servers	Long-lasting PVSS (sec. 6.1)			Post-quantum PVSS (sec. 5)			Gentry et al. [41]		
	Dealer	Receiver	Size	Dealer	Receiver	Size	Dealer	Receiver	Size
16	14.12 ms	5.43 ms	68.8 KB	10.47 ms	3.08 ms	69.3 KB	36.57 s	1.12 s	2868 KB
32	28.14 ms	8.09 ms	137.5 KB	20.06 ms	3.14 ms	138.5 KB	53.26 s	1.13 s	5804 KB
64	56.36 ms	15.50 ms	275 KB	40.60 ms	3.07 ms	277 KB	100.8 s	1.13 s	11.61 MB
128	112.60 ms	38.52 ms	550 KB	80.89 ms	3.38 ms	554 KB	196.0 s	1.19 s	23.85 MB
256	225.23 ms	125.69 ms	1100 KB	163.75 ms	3.10 ms	1108 KB	406.6 s	1.20 s	50.0 MB
512	451.92 ms	521.05 ms	2200 KB	326.43 ms	3.14 ms	2216 KB	882.36 s	1.19 s	109.19 MB
1024	907.80 ms	2.20 s	4400 KB	666.90 ms	3.14 ms	4432 KB	2060.48 s	1.25 s	251.5 MB

Table 2: Measurements for Long-lasting and Post-quantum PVSS schemes, in comparison to Gentry et al. [41]

lattices, and its open-source implementation [2][6]. We set $N = 1024$ and $q = 2^{29}$. For our modified $q = 2^{29}$ instantiation, the current `lattice-estimator` analysis gives an estimated ciphertext-recovery cost of approximately $2^{110.1}$ operations; see Appendix B.3. The concrete IBE instantiation provides approximately 110-bit classical security against the modeled ciphertext-recovery attacks; while the PVSS has a soundness of $1 - 2^{-128}$. The on-chain vote-verification logic is implemented as an Aptos Move smart contract [3]. All experiments were run on a MacBook Pro with an Apple M4 Max Pro chip and 64 GB RAM using a *single computational thread*. Each reported data point is the mean over 10 independent runs.

7.1 PVSS Experimental Results

Along with the two PVSS schemes from this paper, we also experiment with the lattice-based non-interactive VSS scheme by Gentry, Halevi and Lyubashevsky [41]. We compare all three schemes on the following dimensions: dealer running time, receiver running time, and communication cost. The dealer time includes the time for a dealing client to generate encrypted shares and the proof for all servers. The communication cost is the size of the payload that the dealer *must* publish/broadcast and the pairwise communication. The receiver verification time includes the time to derive the IBE decryption key, decrypt the IBE-encrypted share, and verify the proof. The empirical measurements in Table 2 clearly demonstrate that interaction between the receivers/verifiers can avoid expensive NIZKs and achieve massive improvement in running time.

Runtime and Memory Usage Comparison. In our post-quantum secure PVSS, the identity-specific secret key derivation accounts for 90-95% of the running time, while the other computations – deserialization, IBE encryptions/decryptions, polynomial evaluation – are relatively inexpensive. In contrast, our long-lasting PVSS computes a SCRAPE test during verification, whose running time grows with the server count. The SCRAPE degree-test requires Multi-Scalar Multiplication and that forms a dominant component of the running time. Hence, applications that do not need the homomorphic property can benefit from the post-quantum PVSS, which has a much smaller verification time (apart from being fully post-quantum secure). Meanwhile, the runtime of [41] is dominated by their encryptions, as it requires 1.2 s to compute an encryption. Usage of Bulletproofs, instead of lattice-based NIZK, improves their proving runtime yet incurs a significant overhead (1000×) compared to our post-quantum PVSS. Similar improvements were observed for our PVSS protocols over [41] in terms of memory consumption (Table. 3), which is significant for their scheme.

Servers	Long-lasting PVSS		Post-quantum PVSS		[41]
	Dealer	Receiver	Dealer	Receiver	Overall
16	0.75 MB	64.38 MB	0.75 MB	64.38 MB	-
32	1.51 MB	64.63 MB	1.51 MB	64.63 MB	-
64	3.01 MB	65.13 MB	3.01 MB	65.13 MB	-
128	6.03 MB	66.14 MB	6.03 MB	66.14 MB	2.26 GB
256	12.05 MB	68.15 MB	12.05 MB	68.15 MB	2.73 GB
512	24.11 MB	72.18 MB	24.10 MB	72.17 MB	3.74 GB
1024	48.21 MB	80.24 MB	48.20 MB	80.22 MB	5.28 GB

Table 3: Memory usage for Long-lasting and Post-quantum PVSS schemes, in comparison to Gentry et al. [41]

Communication Comparison. The lattice-based IBE ciphertexts (4352 bytes each) account for most of the communication size in each dealing, in both of our PVSS schemes. The post-quantum PVSS has slightly higher communication cost due to the use of hash-based commitments which grow with the number of servers. In contrast, [41] incurs at least an order of magnitude overhead in communication due to the lattice-based PVW encryptions [56] used in their dealing.

7.2 Polling Experimental Results

The various components of the voting application are depicted in Sec. 6.2. The voters use a command-line interface (CLI) app to submit their vote on-chain[7], which includes the IBE-encrypted Shamir shares, commitments, and NIZK proof. Next, the servers decrypt and validate the votes, and perform any complaints as mandated by the PVSS protocol. Finally, the servers aggregate the votes and compute the outcome, as described in Section 6.2.

To evaluate scalability for polling application, we systematically varied the number of poll choices $\in \{4, 8, 16\}$, and the number of servers in the committee $n \in \{16, 32, 64, 128\}$. We report costs separately for each role (i) *Client* - Time to produce an encrypted vote and the accompanying proofs, (ii) *Smart Contract* - Time for the Aptos Move contract to verify a submitted vote, and (iii) *Server*

[6]The library[2] is written in C++, so we expose it to Rust via FFI bindings.

[7]Since on-chain storage can be expensive, the system can be easily modified to use some data availability solution.

Poll Choices	Vote Size (KB) # of Servers 16	32	64	128	Smart Contract Cost (ms)
4	74.8	148.8	296.8	592.8	0.452
8	81.4	161.4	321.4	641.4	0.806
16	94.6	186.6	370.6	738.6	1.514

Table 4: Vote sizes for different numbers of poll choices and servers, and the corresponding MOVE smart contract verification time to verify an encrypted vote. The verification time is independent of the number of servers.

- Time for Client vote validation, Votes aggregation, and Output computation.

Client Costs. Figure 1 plots the total time for a poll participant to generate an encrypted vote, including zero-knowledge proofs of correct encryption. Client-side vote encryption and zero-knowledge proof generation increase with the number of choices and servers. For 128 servers, it takes 109 ms and 183 ms to vote for 4 and 16 poll choices, respectively, showing that client computation dominates latency but remains practical.

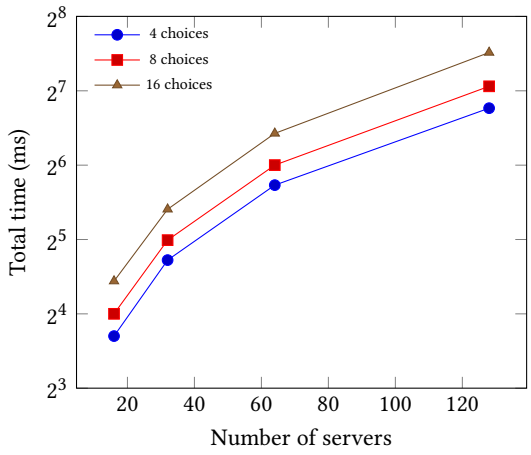
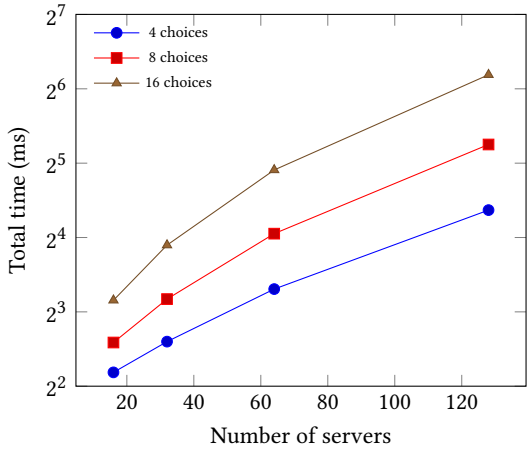


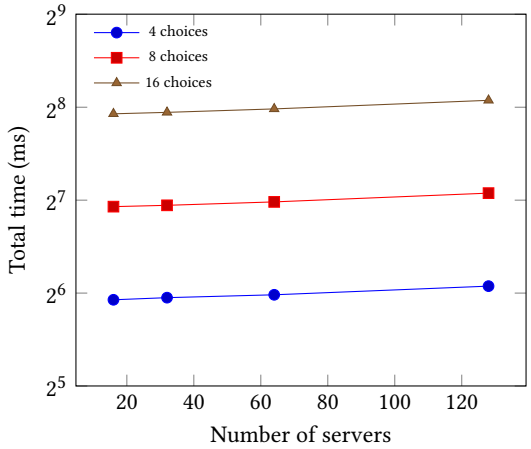
Figure 1: Computation time (ms) required for a poll participant to generate an encrypted vote using our private polling protocol, for varying numbers of poll choices.

Smart Contract Costs. Smart contract verification (Table 4) of encrypted votes scales with the number of choices but not servers, taking under 1.6 ms even for 16 choices, making the on-chain overhead negligible compared to client and server computations.

Server Costs. Each server in the committee validates all client votes and computes the final output. This process involves two phases: (i) *client's vote validation* performed independently for each vote, and (ii) *vote aggregation and output computation* done once collectively over all validated votes. Figure 2a shows that server validation of a single vote scales linearly with poll choices and committee size, remaining under 75 ms even for 128 servers. This is the most expensive protocol step, but it is performed locally by each server whenever a client submits a vote. Figure 2b shows that aggregating 100,000 votes and computing the final output scales



(a) Per-vote validation.



(b) Aggregation and output reconstruction (100,000 votes).

Figure 2: Server computation in our polling protocol with varying number of poll choices. (a) Per-vote validation performed independently by each server. (b) Committee-wide aggregation and output computation across all validated votes.

mainly with poll choices, increasing only slightly with more servers, and remains under 300 ms even for 128 servers and 16 choices.

Discussion. Our evaluation shows scalable, secure voting with post-quantum vote privacy, low client overhead, a lightweight smart contract, and efficient server-side computation - making it practical for large scale deployment.

8 CONCLUSION

As PVSS gains attention for its use in decentralized ecosystems like blockchains, it is important to study PVSS compatible with such settings – we began by examining PVSS with non-interactive dealers. Since the lack of post-quantum security hinders adoption of applications like private polling, we focus on post-quantum secure (or at least post-quantum private) PVSS. Our construction improves upon the state-of-the-art post-quantum PVSS results by an order of magnitude. We also introduce the security model of long-lasting

security that provides full classical computational security during execution and provides post-quantum privacy in the future. This model accurately captures protocols that are elliptic curve+lattice-based (e.g., Zama's FHEVM) but is secure against "harvest now, decrypt later" attacks.

ACKNOWLEDGEMENT

We thank the anonymous reviewers of PQQS'26 for their insightful feedback and constructive suggestions that helped to significantly improve the paper. The full version can be found in [**?**].

REFERENCES

[1] curve25519-dalek library. https://crates.io/crates/curve25519-dalek.

[2] Lattice-ibe library. https://github.com/tprest/Lattice-IBE.

[3] Move - a web3 language and runtime. https://aptos.dev/en/network/blockchain/move.

[4] Behzad Abdolmaleki, John Clark, Mohammad Foroutani, Shahram Khazaei, and Sajjad Nasirzadeh. A generic framework for practical lattice-based non-interactive publicly verifiable secret sharing. *IACR Cryptol. ePrint Arch.*, page 901, 2025.

[5] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 39–70. Springer, 2023.

[6] Shweta Agrawal, Andrea Basso, and Sikhar Patranabis. IBE and PIR from isogenies. Cryptology ePrint Archive, Paper 2026/1457, 2026.

[7] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010.

[8] Nicolas Alhaddad, Mayank Varia, and Ziling Yang. Haven++: Batched and packed dual-threshold asynchronous complete secret sharing with applications. *IACR Commun. Cryptol.*, 1(4):3, 2024.

[9] Ghada Almashaqbeh and Ravital Solomon. Sok: Privacy-preserving computing in the blockchain era. In *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*, pages 124–139. IEEE, 2022.

[10] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Round-optimal honest-majority MPC in minicrypt and with everlasting security - (extended abstract). In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 103–120. Springer, 2022.

[11] Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. Practical everlasting privacy. In David A. Basin and John C. Mitchell, editors, *Principles of Security and Trust - Second International Conference, POST 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7796 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2013.

[12] Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. VSS from distributed ZK proofs and applications. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part I*, volume 14438 of *Lecture Notes in Computer Science*, pages 405–440. Springer, 2023.

[13] Renas Bacho and Alireza Kavousi. Sok: Dlog-based distributed key generation. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, pages 614–632. IEEE, 2025.

[14] Renas Bacho and Julian Loss. Adaptively secure (aggregatable) PVSS and application to distributed randomness beacons. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 1791–1804. ACM, 2023.

[15] Karim Baghery. Π: A unified framework for computational verifiable secret sharing. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12-15, 2025, Proceedings, Part IV*, volume 15677 of *Lecture Notes in Computer Science*, pages 110–142. Springer, 2025.

[16] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

[17] Assi Barak, Martin Hirt, Lior Koskas, and Yehuda Lindell. An end-to-end system for large scale P2P mpc-as-a-service and low-bandwidth MPC for weak participants. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 695–712. ACM, 2018.

[18] Benjamin Benčina, Benjamin Dowling, Varun Maram, and Keita Xagawa. Post-quantum cryptographic analysis of SSH. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, pages 595–613. IEEE, 2025.

[19] Denis Berger, Mouad Lemoudden, and William J. Buchanan. Post-quantum migration of the tor application. *J. Cybersecur. Priv.*, 5(2):13, 2025.

[20] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[21] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. https://toc.cryptobook.us/book.pdf, 2023. (Version 0.6).

[22] Xavier Boyen, Thomas Haines, and Johannes Müller. Epoque: Practical end-to-end verifiable post-quantum-secure e-voting. In *IEEE European Symposium on Security and Privacy, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*, pages 272–291. IEEE, 2021.

[23] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018.

[24] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, 2000.

[25] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 1999.

[26] Ignacio Cascudo and Bernardo David. SCRAPE: scalable randomness attested by public entities. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings*, volume 10355 of *Lecture Notes in Computer Science*, pages 537–556. Springer, 2017.

[27] Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part V*, volume 14655 of *Lecture Notes in Computer Science*, pages 216–248. Springer, 2024.

[28] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[29] Yi-Hsiu Chen and Yehuda Lindell. Feldman's verifiable secret sharing for a dishonest majority. *IACR Commun. Cryptol.*, 1(1):16, 2024.

[30] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[31] Sourav Das, Zhuolun Xiang, Alin Tomescu, Alexander Spiegelman, Benny Pinkas, and Ling Ren. Verifiable secret sharing simplified. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, pages 633–651. IEEE, 2025.

[32] Bernardo David, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 683–696. ACM, 2022.

[33] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1565–1581. ACM, 2017.

[34] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

[35] Yevgeniy Dodis, Jiaxin Guan, Peter Hall, and Alison Lin. HELP: everlasting privacy through server-aided randomness. *IACR Commun. Cryptol.*, 1(4):6, 2024.

[36] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2014.

[37] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 427–437. IEEE Computer Society, 1987.

[38] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. SWOOSH: efficient lattice-based non-interactive key exchange. In Davide Balzarotti and Wenyuan Xu, editors, *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.

[39] Sanjam Garg, Aarushi Goel, Abhishek Jain, Guru-Vamsi Policharla, and Sruthi Sekar. zksaas: Zero-knowledge snarks as a service. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 4427–4444. USENIX Association, 2023.

[40] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, 2007.

[41] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 458–487. Springer, 2022.

[42] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008.

[43] Jacob Gorman, Lucjan Hanzlik, Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Pratik Sarkar, and Sri Aravinda Krishnan Thyagarajan. Vraas: Verifiable randomness as a service on blockchains. *IACR Cryptol. ePrint Arch.*, page 957, 2024. Accepted in 38th IEEE Computer Security Foundations Symposium.

[44] Jens Groth. Non-interactive distributed key generation and key resharing. *IACR Cryptol. ePrint Arch.*, page 339, 2021.

[45] Thomas Haines, Rafieh Mosaheb, Johannes Müller, and Ivan Pryvalov. Sok: Secure e-voting with everlasting privacy. *Proc. Priv. Enhancing Technol.*, 2023(1):279–293, 2023.

[46] Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa. Certified everlasting secure collusion-resistant functional encryption, and more. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part III*, volume 14653 of *Lecture Notes in Computer Science*, pages 434–456. Springer, 2024.

[47] Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive VSS using class groups and application to DKG. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 4286–4300. ACM, 2024.

[48] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010.

[49] Jonathan Katz. Round-optimal, fully secure distributed key generation. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 285–316. Springer, 2024.

[50] Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. Sok: Public randomness. In *9th IEEE European Symposium on Security and Privacy, EuroS&P 2024, Vienna, Austria, July 8-12, 2024*, pages 216–234. IEEE, 2024.

[51] Eleftherios Kokoris-Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1751–1767. ACM, 2020.

[52] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

[53] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[54] Pham Nhat Minh, Khoa Nguyen, Willy Susilo, and Khuong Nguyen-An. Publicly verifiable secret sharing: Generic constructions and lattice-based instantiations in the standard model. *CoRR*, abs/2504.14381, 2025.

[55] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[56] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[57] David Pointcheval. Efficient universally-verifiable electronic voting with everlasting privacy. In Clemente Galdi and Duong Hieu Phan, editors, *Security and Cryptography for Networks - 14th International Conference, SCN 2024, Amalfi, Italy, September 11-13, 2024, Proceedings, Part I*, volume 14973 of *Lecture Notes in Computer Science*, pages 323–344. Springer, 2024.

[58] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. IACR Cryptology ePrint Archive, Report 2014/254, 2014.

[59] Hamza Saleem and Rohit Sinha. Practical post-quantum secure publicly verifiable secret sharing and applications. https://github.com/hamzahsaleem/post-quant-pvss.

[60] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar R. Weippl. ETHDKG: distributed key generation with ethereum smart contracts. *IACR Cryptol. ePrint Arch.*, page 985, 2019.

[61] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 148–164. Springer, 1999.

[62] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[63] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[64] Daniel Filipe Nunes Silva. Exploring encrypted on-chain markets with orca, 2025. https://www.arcium.com/articles/exploring-encrypted-on-chain-markets-with-orca.

[65] Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, 1996.

[66] Partisia Blockchain Team. Demeter — mpc-as-a-service, 2022. https://partisiablockchain.com/demeter-mpc-as-a-service/.

[67] Dominique Unruh. Everlasting multi-party computation. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 380–397. Springer, 2013.

A ADDITIONAL RELATED WORKS

We compare the everlasting security notion with long-lasting security in this section. Then, we briefly compare our voting scheme with other relevant voting protocols.

A.1 Comparison between Everlasting and Long-lasting Security Models

Everlasting Model: Unruh [67] proposed the notion of everlasting security that requires the protocol to be secure against classical PPT adversaries during protocol execution and secure against computationally unbounded adversaries after protocol execution. They constructed quantum-UC secure communication protocols using trusted signature cards and quantum channels. Following this work,

there has been constructed everlasting secure MPC protocols [10] in the honest majority setting, assuming secure channels. Then [46] constructed certified everlasting secure functional encryption assuming quantum channels. There have been secure voting protocols [45, 57] with everlasting privacy as well. However, all these protocols require quantum channels for secure communication. To implement a secure channel in the everlasting security model, one needs to have quantum secure channels. An alternate solution was proposed by [35] where a secure channel is assumed between one designated party, called the hypervisor, and the protocol participants.

However, the current practical feasibility of quantum channels for regular cryptographic protocols is unclear, and the assumption of a secure channel between the parties and the hypervisor is also not realistic in nature. Hence, everlasting security is difficult to implement in practice. So, we relaxed the notion of everlasting security to long-lasting security where the post-execution security is only required to hold against post-quantum adversaries instead of information-theoretic adversaries. This is a meaningful relaxation as it allows us to build provably secure PVSS (with non-interactive dealer) and voting protocols that can be readily instantiated using post-quantum secure assumptions and highly optimized DL-based cryptography. As an application, we provide a long-lasting secure voting protocol.

A.2 Comparison of Secure Voting Protocols with Post-Quantum Vote Privacy

We compare our voting protocol with other voting protocols that provide post-quantum or everlasting privacy.

Post-Quantum Secure Voting: The state-of-the-art lattice-based voting scheme by [33] enforces the voting client to vote by secret sharing its vote, committing to it and encrypting the shares under each server's public key. The client also proves that the committed shares sum up to 0 or 1. Then each server decrypts its share and computes a valid NIZK proof that the committed share has been tallied correctly. This way, a client delegates the proof computation to the voting servers. Considering 4 servers, each of their server runs for 41 min to conduct a poll with two choices and participation from 11K voters. Meanwhile, our polling smart contract+server protocol performs the same task in a minute, yielding a 40× improvement. We note that this would further increase with the increase in the number of poll choices and the number of servers, as our cost grows sublinearly. The Epoque protocol [22] has similar performance as [33] since they use the same underlying NIZK proofs, and hence our improvement over Epoque would be similar.

Everlasting Secure Voting: There are several everlasting secure voting protocols [45, 57] based on different properties that a voting scheme can satisfy. We do not delve into that for the sake of brevity. However, we note that our long-lasting secure protocol provides similar performance as the state-of-the-art protocol of [57].

B ADDITIONAL PRELIMINARIES

Here, we present the full details of our preliminaries from Sec. 2 for the sake of completeness.

B.1 SCRAPE-Style Degree Test [26]

Setting. Let F be a finite field of prime order with $|F| > n$, let $\mathbb{G}$ be a group of order $|F|$ with independent generators g, h, and let $\mathrm{Com}(m;\rho) = g^m h^\rho$ be the Pedersen commitment (Sec. 2); in particular $\mathrm{Com}(0;0) = 1_\mathbb{G}$. We use the evaluation points $x_i := i$ for $i = 0,\dots,n$, where $x_0 = 0$ is the position of the secret. For a dealing $\mathsf{ShareCom}(s,n,t)$ with polynomials $f(X), r(X)$ of degree $\le t$, $f(0) = s$ and $r(0) = r$, the shares are $[\![s]\!]_i = f(i)$ and $[\![r]\!]_i = r(i)$ for $i \in [n]$. We set $[\![s]\!]_0 := s$ and $[\![r]\!]_0 := r$, and write

$$\mathsf{csh}_0 := c = \mathrm{Com}(s;r), \qquad \mathsf{csh}_i = \mathrm{Com}([\![s]\!]_i; [\![r]\!]_i) \quad (i \in [n]).$$

Let $t < n$ and define

$$\mathcal{N} := \Big\{ w \in F^{n+1} : \sum_{i=0}^{n} w_i\, i^j = 0 \text{ for } j = 0,\dots,t \Big\}, \qquad \dim \mathcal{N} = n-t.$$

A public basis matrix $M \in F^{(n+1)\times(n-t)}$ of $\mathcal{N}$ depends only on (n,t) and is fixed once. We model a hash function $H_{\mathsf{scr}} : \{0,1\}^* \to F^{n-t}$ as a random oracle (Fiat–Shamir). Two facts about $\mathcal{N}$ are used below.

(i) If $w \in \mathcal{N}$ and $\deg(p) \le t$, then $\sum_{i=0}^{n} w_i\, p(i) = 0$.
(ii) If $v \in F^{n+1}$ satisfies $\langle w, v\rangle = 0$ for every $w \in \mathcal{N}$, then $v = (p(0),\dots,p(n))$ for some p with $\deg(p) \le t$.

Item (ii) holds because (i) gives $\mathrm{RS}_t \subseteq \mathcal{N}^\perp$, and both spaces have dimension $(n+1) - (n-t) = t+1$.

The predicate $\mathcal{F}_{\mathsf{SCRAPE}}$. $\mathcal{F}_{\mathsf{SCRAPE}}\big((n,t),\, c,\, \mathsf{csh}_1,\dots,\mathsf{csh}_n\big) \in \{0,1\}$ is a deterministic, publicly computable predicate with no secret input. It is evaluated in PubVer (Sec. 6.1, step 2) on the dealer's commitments, and in the client-side validation of the polling protocol (Sec. 6.2, step 1) once for each option $j \in [4]$ on $(c_j, \mathsf{csh}_{j,1},\dots,\mathsf{csh}_{j,n})$. It is computed as follows.

(1) *Inputs:* set $C_0 := c$ and $C_i := \mathsf{csh}_i$ for $i \in [n]$. If some $C_i \notin \mathbb{G}$, output 0.
(2) *Challenge:* $u := H_{\mathsf{scr}}\big((n,t), \mathbb{G}, g, h, C_0, C_1, \dots, C_n\big) \in F^{n-t}$ and $w := M \cdot u \in \mathcal{N} \subseteq F^{n+1}$.
(3) *Aggregation:* $D := \prod_{i=0}^{n} C_i^{w_i} \in \mathbb{G}$.
(4) *Local verification:* output 1 if $D = 1_\mathbb{G}$, and 0 otherwise.

If a caller needs domain separation (for example a poll identifier pid), it appends it to the hash input.

Lemma B.1 (Completeness). *If $c, \mathsf{csh}_1,\dots,\mathsf{csh}_n$ are produced by $\mathsf{ShareCom}(s,n,t)$, then $\mathcal{F}_{\mathsf{SCRAPE}}((n,t), c, \mathsf{csh}_1,\dots,\mathsf{csh}_n) = 1$.*

Proof. By fact (i) applied to f and $r(X)$, every $w \in \mathcal{N}$ satisfies $\sum_i w_i [\![s]\!]_i = 0$ and $\sum_i w_i [\![r]\!]_i = 0$. By the homomorphism of Com, $D = \prod_i \mathrm{Com}([\![s]\!]_i; [\![r]\!]_i)^{w_i} = \mathrm{Com}\big(\sum_i w_i [\![s]\!]_i; \sum_i w_i [\![r]\!]_i\big) = \mathrm{Com}(0;0) = 1_\mathbb{G}$. □

Lemma B.2 (Soundness). *Let $a = (a_0,\dots,a_n)$ and $b = (b_0,\dots,b_n)$ be fixed vectors over F, and assume that at least one of them is not an evaluation vector (at $0,\dots,n$) of a polynomial of degree at most t. For uniform $u \in F^{n-t}$ and $w = Mu$,*

$$\Pr\Big[\textstyle\sum_{i=0}^{n} w_i a_i = 0 \ \wedge\ \sum_{i=0}^{n} w_i b_i = 0\Big] \le \tfrac{1}{|F|}.$$

Consequently, in the random-oracle model, let a dealer make Q_H queries to H_{scr} and output $c, \mathsf{csh}_1,\dots,\mathsf{csh}_n$ together with openings $\mathsf{csh}_i = \mathrm{Com}(a_i; b_i)$ (with a_0, b_0 the opening of c) such that (a,b) is not a pair of degree-$\le t$ sharings.

Then $\mathcal{F}_{\mathsf{SCRAPE}}((n,t), c, \mathsf{csh}_1, \ldots, \mathsf{csh}_n) = 1$ *with probability at most* $Q_H/|F| + \mathsf{Adv}^{\mathsf{bind}}_{\mathsf{Com}}$.

Proof. Suppose a is not a degree-$\leq t$ evaluation vector (the case of b is the same). By fact (ii) there is $w' \in \mathcal{N}$ with $\langle w', a\rangle \neq 0$. Since M has full column rank, $u \mapsto \langle Mu, a\rangle$ is a nonzero linear form on F^{n-t}, so it is uniform over F and vanishes with probability $1/|F|$. This gives the first claim. For the second claim, the challenge for each queried instance is a fresh uniform value, so a given instance passes the first claim's test with probability at most $1/|F|$, and a union bound over the Q_H queries gives $Q_H/|F|$. This step needs the openings (a, b) to be fixed by the commitments C_i. Except with probability $\mathsf{Adv}^{\mathsf{bind}}_{\mathsf{Com}}$, the dealer cannot open any C_i in two ways, so they are fixed. Otherwise, if $D = 1_{\mathbb{G}}$ while $(\sum_i w_i a_i, \sum_i w_i b_i) \neq (0,0)$, this pair is a second opening of $1_{\mathbb{G}} = \mathsf{Com}(0;0)$, which breaks binding. □

Use in the security proofs. If $\mathcal{F}_{\mathsf{SCRAPE}}$ accepts, then except with probability $Q_H/|F| + \mathsf{Adv}^{\mathsf{bind}}_{\mathsf{Com}}$ the committed values $([\![s]\!]_i)_{i\in[n]}$ and $([\![r]\!]_i)_{i\in[n]}$, together with the opening of c, lie on degree-$\leq t$ polynomials. This property is used for reconstructability in long-lasting protocols.

B.2 Pedersen Commitment [28]:

It has three algorithms:

- $\mathsf{Gen}(1^\lambda)$: Sample $g, h \leftarrow \mathbb{G}$ where $|\mathbb{G}| = O(2^\lambda)$ and set the public parameters as (g, h).
- $\mathsf{Com}(m; s)$: To commit to message m, sample the randomness $s \leftarrow \mathbb{Z}$ and compute the commitment as $c = g^m h^s$.
- $\mathsf{Verify}(c, m, s)$: To verify the opening of the commitment check that $c \stackrel{?}{=} g^m h^s$.

It is a perfectly hiding and computationally binding commitment scheme assuming the hardness of solving Discrete-Log.

B.3 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme [20] is a type of public-key encryption in which the public key of a user can be derived from an arbitrary string representing the user's identity (e.g., an email address).

Formally, an IBE scheme consists of the following probabilistic polynomial-time algorithms:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: Takes a security parameter λ and outputs a master public key mpk and a master secret key msk.
- $\mathsf{Extract}(\mathsf{msk}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$: Takes the master secret key msk and an identity string ID, and outputs the private key $\mathsf{sk}_{\mathsf{ID}}$ corresponding to ID.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, m) \to c$: Takes the master public key mpk, an identity ID, and a message m, and outputs a ciphertext c encrypted under ID.
- $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) \to m$: Takes the private key $\mathsf{sk}_{\mathsf{ID}}$ and a ciphertext c, and outputs the original message m (or $\perp$ if c is malformed).
- $\mathsf{VerKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) \to \{0,1\}$: Takes the master public key mpk, identity ID and an identity secret key $\mathsf{sk}_{\mathsf{ID}}$ and outputs 1 if $\mathsf{sk}_{\mathsf{ID}}$ is accepted as the identity secret key corresponding to ID and 0 otherwise. Note that VerKey does not take a ciphertext as input.

Correctness: An IBE scheme is said to be **correct** if, for every identity string ID and every message m in the message space, the decryption algorithm correctly recovers m from the ciphertext c produced by the encryption algorithm. Formally, for all ID and m:

$$\Pr\left[\mathsf{Dec}(\mathsf{Extract}(\mathsf{msk}, \mathsf{ID}), \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, m)) \neq m\right] \leq \mathsf{negl}(\lambda).$$

For the instantiation below the correctness error is exactly 0 (Theorem B.2).

IND-ID-CPA Security: An IBE scheme is said to be indistinguishable under chosen plaintext attack for a given identity (IND-ID-CPA) if a PPT adversary $\mathcal{A}$ wins the following game with $\mathsf{negl}(\lambda)$ probability.

IND-ID-CPA Game:

(1) The challenger runs $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$ and gives mpk to $\mathcal{A}$.
(2) $\mathcal{A}$ may query the key extraction oracle $\mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$ for any identity ID to receive $\mathsf{sk}_{\mathsf{ID}}$.
(3) The adversary chooses two messages m_0, m_1 and a challenge identity ID^* that has not been queried to the $\mathsf{Extract}(\mathsf{msk}, \cdot)$ oracle.
(4) The challenger flips a bit $b \leftarrow \{0,1\}$ and returns $c^* := \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, m_b)$.
(5) The adversary continues to make key extraction queries for $\mathsf{ID} \neq \mathsf{ID}^*$ and eventually outputs a guess b'.

The IBE scheme is IND-ID-CPA secure if:

$$\left|\Pr[b' = b] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

We use a *modified DLP IBE*, denoted IBE, obtained from the lattice-based IBE of [36], which is based on the GPV scheme [42] over NTRU lattices. It differs from [36] in exactly two places: (M1) Enc samples the encryption randomness (r, e_1) from the uniform ternary distribution *conditioned* on $\mathsf{wt}(r) + \mathsf{wt}(e_1) \leq B_x^2$; and (M2) Extract repeats the Gaussian sampler until the output passes the norm test of VerKey. The original scheme of [36] does not have the deterministic key-verifiability property of Theorem B.1; IBE does. Its security relative to [36] is analyzed in Theorem B.3.

Key Verifiability: An IBE scheme is said to satisfy key verifiability if for any security parameter λ, any identity ID, any message m, and any stateful adversary $\mathcal{A}$, the following probability is negligible in λ.

$$\Pr\left[\begin{array}{c} m' := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) \notin \{m, \perp\}; \\ \mathsf{VerKey}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) = 1 \end{array} \middle| \begin{array}{c} \mathsf{mpk} \leftarrow \mathcal{A}(1^\lambda); \\ c \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, m); \\ \mathsf{sk}_{\mathsf{ID}} \leftarrow \mathcal{A}(c) \end{array}\right]$$

Informally, except with negligible probability, an adversary can not produce a master public key, such that it can later release a decryption key which verifies correctly, but leads to an incorrect decryption to a valid message $m' \neq m$. Since VerKey does not depend on the ciphertext, a dealer who produces the ciphertext cannot influence whether an honestly extracted key verifies (Theorem B.2). For IBE the probability above is exactly 0 (Theorem B.1).

We provide the algorithms of IBE, adapted to the parameters $N = 1024$, $q = 2^{29}$ (Sec. B.3), together with the ciphertext-independent VerKey algorithm below. The decryption algorithm Dec and the sampling and rounding steps of Enc are made explicit so that the proofs in Sec. B.3 are exact. Throughout, $R_q := \mathbb{Z}_q[x]/(x^N + 1)$, $[a]_q$ denotes the centered representative of a in $(-q/2, q/2]$ (applied coefficient-wise), and all norms $\|\cdot\|_2$, $\|\cdot\|_\infty$ are taken over the integer coefficient vectors of centered representatives.

Gaussian Sampler(B, σ, c)

Require: Basis B of an n-dimensional lattice Λ, standard deviation $\sigma > 0$, center $c \in \mathbb{Z}^n$
Ensure: v sampled in $D_{\Lambda,\sigma,c}$

(1) $v_n := 0$
(2) $c_n := c$
(3) for $i := n, \ldots, 1$
  (a) $c'_i := \langle c_i, \tilde{b}_i \rangle / \|\tilde{b}_i\|^2$
  (b) $\sigma'_i := \sigma / \|\tilde{b}_i\|$
  (c) $z_i := \mathsf{SampleZ}(\sigma'_i, c'_i)$
  (d) $c_{i-1} := c_i - z_i b_i$ **and** $v_{i-1} := v_i + z_i b_i$
(4) **return** v_0.

KeyGen(N, q)

Require: N, q
Ensure: Master Secret Key $B \in \mathbb{Z}_q^{2N \times 2N}$ and Master Public Key $h \in R_q$

(1) $\sigma_f = 1.17\sqrt{\frac{q}{2N}}$ ▷ σ_f chosen such that $\mathbb{E}[\|b_1\|] = 1.17\sqrt{q}$
(2) $f, g := D_{N,\sigma_f}$
(3) Norm $:= \max\left\{\|(g, -f)\|, \left\|\left(\frac{q\bar{f}}{f*\bar{f}+g*\bar{g}}, \frac{q\bar{g}}{f*\bar{f}+g*\bar{g}}\right)\right\|\right\}$ ▷ Compute $\|\tilde{B}_{f,g}\|$
(4) if Norm $> 1.17\sqrt{q}$ **go to** step 2
(5) Using extended Euclidean algorithm, compute $\rho_f, \rho_g \in R$ and $R_f, R_g \in \mathbb{Z}$ such that:
  - $\rho_f \cdot f = R_f \bmod (x^N + 1)$
  - $\rho_g \cdot g = R_g \bmod (x^N + 1)$
(6) if $\gcd(R_f, R_g) \neq 1$ **or** $\gcd(R_f, q) \neq 1$ **go to** step 2
(7) Using extended Euclidean algorithm, compute $u, v \in \mathbb{Z}$ such that $u \cdot R_f + v \cdot R_g = 1$
(8) $F := qv\rho_g, \quad G := -qu\rho_f$
(9) $k = \left\lfloor \frac{F * \bar{f} + G * \bar{g}}{f * \bar{f} + g * \bar{g}} \right\rceil \in R$
(10) Reduce F and G: $F := F - k * f, \quad G := G - k * g$
(11) $h = g * f^{-1} \bmod q$
(12) $B = \begin{pmatrix} A(g) & -A(f) \\ A(G) & -A(F) \end{pmatrix}$

Extract(B, ID)

Require: Master Secret Key $B \in \mathbb{Z}_q^{2N \times 2N}$, master public key h, hash function $H : \{0,1\}^* \to \mathbb{Z}_q^N$, user identity ID, sampler width $\sigma := 2 \cdot 1.17\sqrt{q}$
Ensure: User secret key $\mathsf{SK}_{\mathsf{ID}} \in R_q$

If $\mathsf{sk}_{\mathsf{ID}}$ is in local storage
- then output $\mathsf{SK}_{\mathsf{ID}}$ to user ID

Else
- $t := H(\mathsf{ID}) \in \mathbb{Z}_q^N$
- Repeat $(s_1, s_2) := (t, 0) - \text{Gaussian Sampler}(B, \sigma, (t, 0))$ until $\mathsf{VerKey}(h, \mathsf{ID}, s_2) = 1$.
- $\mathsf{sk}_{\mathsf{ID}} := s_2$
- Output $\mathsf{sk}_{\mathsf{ID}}$ to user ID and keep it in local storage

Enc(h, ID, m)

Require: Hash functions $H : \{0,1\}^* \to \mathbb{Z}_q^N$ and $H' : \{0,1\}^N \to \{0,1\}^m$, message $m \in \{0,1\}^m$, Master Public Key $h \in R_q$, identity ID, rounding parameter ℓ, norm bound B_x
Ensure: Encryption $(u, v, c') \in R_q^2 \times \{0,1\}^m$ of m under the public key of ID

(1) Sample $r, e_1, e_2 \in \{-1, 0, 1\}^N$ uniformly at random; **if** $\mathrm{wt}(r) + \mathrm{wt}(e_1) > B_x^2$ **then** resample (r, e_1) (and repeat). Sample $k := \{0,1\}^N$ uniformly.
(2) $t := H(\mathsf{ID})$
(3) $u := r * h + e_1 \in R_q$
(4) $v := r * t + e_2 + (q/2) \cdot k \in R_q$
(5) Drop the least significant bits of v as: $v := 2^\ell \lfloor v/2^\ell \rceil \bmod q$
(6) Compute $c' := m \oplus H'(k)$.
(7) Output (u, v, c')

Dec$(\mathsf{sk}_{\mathsf{ID}}, (u, v, c'))$

Require: User secret key $\mathsf{sk}_{\mathsf{ID}} = s_2$, ciphertext (u, v, c')
Ensure: Message $m \in \{0,1\}^m$ or $\perp$

(1) If $(u, v, c') \notin R_q^2 \times \{0,1\}^m$ or $\mathsf{sk}_{\mathsf{ID}} \notin R_q$, output $\perp$.
(2) $w := [\, v - u * s_2 \,]_q$
(3) For $i \in [N]$: $k_i := 0$ if $|w_i| < q/4$ and $k_i := 1$ otherwise (equivalently $k := \lfloor w/(q/2) \rceil \bmod 2$).
(4) Output $m := c' \oplus H'(k)$

VerKey$(h, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})$

(1) Parse $s_2 := \mathsf{sk}_{\mathsf{ID}} \in R_q$; output 0 if $\mathsf{sk}_{\mathsf{ID}}$ is malformed.
(2) $t := H(\mathsf{ID})$.
(3) $s_1 := [\, t - s_2 * h \bmod q \,]_q$ and $s_2 := [\, s_2 \,]_q$.
(4) If $\|(s_1, s_2)\|_2 > \Gamma_{\mathsf{key}}$ output 0, where $\Gamma_{\mathsf{key}} := 3{,}120{,}000$ for $N = 1024$, $q = 2^{29}$ (the norm is computed over the integers; the derivation of Γ_{key} is given in Sec. B.3).
(5) Output 1.

Parameters and Their Derivation. We instantiate IBE with $N = 1024$ and exactly $q = 2^{29} = 536{,}870{,}912$. Then $q/2 = 2^{28}$ and $q/4 = 2^{27}$ are integers and $2^\ell \mid q$ for every $\ell \leq 29$; the key generation of [36] only needs $\gcd(R_f, q) = 1$ and does not require q to be prime (see the paragraph on the power-of-two modulus below). We use $\lambda = 192$ for the statistical parameters of the Gaussian sampler, so that the sampler never limits the overall security, which is determined by ciphertext recovery (Sec. B.3). Each quantity is derived as follows.

- **Key-generation norm and sampler width σ.** KeyGen enforces $\|\tilde{B}\| \leq 1.17\sqrt{q} \approx 27{,}109$ (it samples f, g with $\sigma_f = 1.17\sqrt{q/2N} \approx 599$ and restarts when $\|\tilde{B}\| > 1.17\sqrt{q}$). With the KL-divergence argument of [36, Sec. 4] the Klein/GPV

sampler is admissible for $\sigma \geq \eta'_\varepsilon(\mathbb{Z})\|\tilde{B}\|$ and $8n^2\varepsilon^2 \leq 2^{-\lambda}$, where $n = 2N = 2048$ and $8n^2\varepsilon^2$ is the divergence bound of [36, Theorem 2]. Solving for ε gives $\varepsilon \leq 2^{-\lambda/2}/(2\sqrt{2} \cdot n) = 2^{-96}/(2\sqrt{2} \cdot 2048) = 2^{-96}/2^{12.5} = 2^{-108.5}$; this is the condition $\varepsilon \leq 2^{-\lambda/2}/(4\sqrt{2} \cdot N)$ of [36, Sec. 4], where the factor N is multiplied by $4\sqrt{2}$ and is *not* under the square root. We choose the slightly smaller $\varepsilon := 2^{-109}$ for slack, for which $\eta'_\varepsilon(\mathbb{Z}) = \frac{1}{\pi}\sqrt{\frac{1}{2}\ln(2 + 2/\varepsilon)} \approx 1.966$. We therefore set

$$\sigma := 2 \cdot 1.17\sqrt{q} \approx 54{,}219 \ (\approx 2.34\sqrt{q}) \ \geq \ 1.966 \cdot 1.17\sqrt{q}.$$

- **Encryption-randomness bound B_x.** The encryption randomness is $x = (r, e_1) \in \{-1, 0, 1\}^{2N}$ with $\|x\|_2^2 = W := \mathrm{wt}(r) + \mathrm{wt}(e_1)$. For uniform ternary coefficient values $W \sim \mathrm{Bin}(2N, 2/3)$, with mean $4N/3 \approx 1365.3$ and standard deviation $\sqrt{4N/9} \approx 21.3$. The deterministic worst case is $\|x\|_2 \leq \sqrt{2N} \approx 45.25$, which is too large for $q = 2^{29}$ (the window for Γ_{key} below is empty). Soundness requires $B_x \cdot \Gamma^{\min}_{\mathsf{key}} + 1 + 2^{\ell-1} < q/4$ with $\Gamma^{\min}_{\mathsf{key}} = 3{,}103{,}591$ (below), i.e. $B_x \leq (q/4 - 1 - 2^{\ell-1})/\Gamma^{\min}_{\mathsf{key}} = 125{,}829{,}119/3{,}103{,}591 \approx 40.54$ for $\ell = 24$. The probability that an honest dealer has to resample decreases with B_x, so we take the largest integer allowed, $B_x := 40$. Enc then rejects (r, e_1) unless $W \leq B_x^2 = 1600$. An exact computation of the binomial tail gives
$$\Pr[W > 1600] = 2^{-99.1},$$
so the resampling loop terminates after one iteration except with probability $2^{-99.1}$. Note that $B_x = 40$ is about 11 standard deviations above the mean of W (the typical norm is $\sqrt{4N/3} \approx 36.95$).
- **Rounding parameter ℓ.** The compression error satisfies $|\rho_i| \leq 2^{\ell-1}$. With $B_x = 40$ and $\Gamma^{\min}_{\mathsf{key}} = 3{,}103{,}591$ we need $2^{\ell-1} < q/4 - 1 - 40 \cdot 3{,}103{,}591 = 10{,}074{,}087$; the largest admissible value is $\ell = 24$ ($2^{23} = 8{,}388{,}608$).
- **Key-norm bound Γ_{key}.** It is the only parameter used by VerKey and must satisfy two constraints.
  Completeness (lower bound). The pair $(s_1, s_2) = (t, 0) - v$ output by the sampler is distributed, up to the relative error of the Klein sampler (proof of Theorem 2 of [36]; $\delta \approx 2 \cdot 2N\varepsilon = 2^{-97}$ for $\varepsilon = 2^{-109}$), as the discrete Gaussian $D_{\Lambda-(t,0),\sigma}$: the Gaussian of width σ centered at the origin and supported on a coset of the $2N$-dimensional lattice $\Lambda = \Lambda_{h,q}$. Write $\rho_\sigma(x) = \exp(-\|x\|^2/2\sigma^2)$. Since $\sigma \geq \eta'_\varepsilon(\mathbb{Z})\|\tilde{B}\|$ we have $\sigma \geq \eta'_\varepsilon(\Lambda)$ (because $\eta_\varepsilon(\Lambda) \leq \|\tilde{B}\|\,\eta_\varepsilon(\mathbb{Z})$ [42, Lemma 3.1]), hence $\rho_\sigma(\Lambda + u) \geq \frac{1-\varepsilon}{1+\varepsilon}\rho_\sigma(\Lambda)$ for every shift u [53, Lemma 4.4]. Banaszczyk's bound [16, Lemma 1.5] holds for every coset of every n-dimensional lattice: with $\rho(x) = e^{-\pi\|x\|^2}$ and $c_0 > 1/\sqrt{2\pi}$, $\rho((\Lambda + u) \setminus c_0\sqrt{n}\,\mathcal{B}) < (c_0\sqrt{2\pi e}\,e^{-\pi c_0^2})^n \rho(\Lambda)$. After the change of variable $c = c_0\sqrt{2\pi}$ and the rescaling to width σ, and dividing by $\rho_\sigma(\Lambda + u)$, we get for every $c > 1$ and $n = 2N$
$$\Pr\left[\|(s_1, s_2)\|_2 > c\,\sigma\sqrt{2N}\right] \ \leq \ \left(c\,e^{(1-c^2)/2}\right)^{2N} \cdot \tfrac{1+\varepsilon}{1-\varepsilon}.$$
  Let c_λ be the smallest $c > 1$ with $(c\,e^{(1-c^2)/2})^{2N} \leq 2^{-\lambda}$; for $\lambda = 192$ and $N = 1024$ this is $c_\lambda = 1.264878\ldots$ With the unrounded values $c_\lambda = 1.264878\ldots$ and $\sigma = 54{,}218.9115\ldots$,
$$\Gamma_{\mathsf{key}} \ \geq \ \Gamma^{\min}_{\mathsf{key}} := \left\lceil c_\lambda\,\sigma\sqrt{2N} \right\rceil = 3{,}103{,}591.$$
  Soundness (upper bound). By Lemma B.3 below, decryption is correct for every Γ_{key} with $B_x\,\Gamma_{\mathsf{key}} + 1 + 2^{\ell-1} < q/4$, i.e.
$$\begin{aligned}\Gamma_{\mathsf{key}} \ \leq \ \Gamma^{\max}_{\mathsf{key}} &:= \frac{q/4 - 1 - 2^{\ell-1}}{B_x} \\ &= \frac{134{,}217{,}727 - 8{,}388{,}608}{40} \\ &= 3{,}145{,}727.\end{aligned}$$
  The window $[3{,}103{,}591,\ 3{,}145{,}727]$ is non-empty, and we fix $\Gamma_{\mathsf{key}} := 3{,}120{,}000$, which lies inside it. For this value $c = \Gamma_{\mathsf{key}}/(\sigma\sqrt{2N}) \approx 1.27157$ and $(c\,e^{(1-c^2)/2})^{2N} = 2^{-201.4}$, so an honestly sampled key violates the bound with probability $p < 2^{-201}$ (accounting for the factors $\frac{1+\varepsilon}{1-\varepsilon}$ and $1 + 2^{-97}$). It satisfies
$$B_x\,\Gamma_{\mathsf{key}} + 1 + 2^{\ell-1} = 133{,}188{,}609 \ < \ q/4 = 134{,}217{,}728.$$

On the Power-of-Two Modulus. [36] states its schemes for a prime $q \equiv 1 \bmod 2N$, primarily to obtain full splitting of $x^N + 1$ and efficient NTT multiplication. We use exactly $q = 2^{29}$ and state precisely where q enters.

(1) *Algorithms.* KeyGen needs f to be invertible in R_q and (F, G) with $f * G - g * F = q$; the test $\gcd(R_f, q) = 1$, $\gcd(R_f, R_g) = 1$ of [36, Alg. 2] is exactly this condition and does not use primality. For $q = 2^{29}$, the condition $\gcd(R_f, q) = 1$ is equivalent to R_f being odd. Since
$$R_f = \mathrm{Res}(f, x^N + 1) \equiv f(1)^N \pmod 2,$$
this condition is equivalent to $f(1)$ being odd. For a uniformly sampled ternary f this occurs with probability $1/2$. The independent condition $\gcd(R_f, R_g) = 1$ must also be satisfied. Thus the overall KeyGen acceptance probability is not determined solely by the parity condition. The Klein sampler, the Gram–Schmidt computation and the norm bound $\|\tilde{B}\| \leq 1.17\sqrt{q}$ do not depend on the arithmetic structure of R_q.
(2) *Correctness and verifiability.* Every proof in this section uses only integer inequalities, the facts $q/2 \in \mathbb{Z}$ and $2^\ell \mid q$ (which hold for $q = 2^{29}$ exactly and avoid wrap-around in the compression step), and the coefficient bound of Lemma B.3. None of them uses primality or splitting. The deterministic results (Theorems B.1 and B.2) are therefore unconditional for $q = 2^{29}$. These exact encoding and rounding arguments rely on q being even, in particular on $q/2 \in \mathbb{Z}$ and $2^\ell \mid q$; they therefore do not transfer unchanged to a prime modulus and would need to be re-derived.
(3) *Security assumptions.* The security of [36] relies on NTRU and Ring-LWE with very small secrets over R_q, for which no worst-case reduction exists even for prime q. For $q = 2^{29}$ we make the corresponding assumptions over R_q with $q = 2^{29}$, and we point out the two places where the structure of q could matter. (i) *Reduction modulo 2.* The map $x \mapsto 1$ is a

Table 5: Concrete attack estimates for the LWE abstraction of ciphertext recovery with $N = 1024$ and $q = 2^{29}$.

Attack	Estimated cost
BKW	$2^{221.7}$
uSVP	$2^{111.5}$
BDD	$2^{110.1}$
Dual	$2^{113.9}$
Dual-hybrid	$2^{110.3}$
BDD-hybrid	$2^{110.1}$
BDD-MITM-hybrid	$2^{156.1}$

ring homomorphism $R_q \to \mathbb{Z}_2$, because $x^N + 1$ evaluates to $2 \equiv 0 \pmod 2$ at $x = 1$, and it turns a ciphertext into an LPN-type sample with constant noise rate (a ternary error coefficient is odd with probability 2/3). It is not a homomorphism modulo any higher power of 2, so the PLWE-type attacks that require $f(\alpha) \equiv 0 \pmod q$ at a small α do not apply, and we know of no attack that benefits from the mod-2 projection. (ii) *Overstretched NTRU.* The sub-lattice attacks apply when $\|(f, g)\| \ll \sqrt{q}$; here $\|(f, g)\| \approx 1.17\sqrt{q}$ is at the unique-SVP threshold at every q, as in [36, Sec. 5.2], so the attack surface is the same as in the prime-modulus analysis.

(4) *Estimator.* The estimate of Sec. B.3 treats the problem as unstructured LWE with $n = 1024$, $q = 2^{29}$ and ternary secret and error. It does not model the algebraic structure above, so it is a heuristic that does not capture a hypothetical attack specific to power-of-two moduli.

(5) *Efficiency.* Without a splitting prime, multiplication in R_q cannot use a single NTT modulo q. Products of 29-bit coefficients of length 1024 are below 2^{69} and can be computed with an NTT modulo several word-size primes and a CRT recombination followed by reduction mod 2^{29}, or with Karatsuba/FFT; this changes the constants of Enc, Dec, VerKey and Extract but not the asymptotics or the ciphertext format.

Security Level. For $N = 1024$ and $q = 2^{29}$ we estimate the cost of recovering the encapsulated key k (equivalently, the Ring-LWE ciphertext-recovery problem with $n = N = 1024$, ternary secret r and ternary error, as in our concrete-security analysis of the IBE) with the lattice estimator. The estimates are given in Table 5. The cheapest attack is BDD (block size $\beta = 278$) at $2^{110.1}$ operations. The current lattice-estimator abstraction therefore *estimates* the cost of ciphertext recovery at about 110 bits for this parameter regime. This is a heuristic estimate of the best known attacks and not a formal reduction (the estimator treats the problem as unstructured LWE, whereas the scheme is Ring-LWE/NTRU-based). The figure $2^{110.1}$ ($q = 2^{29}$) is modern lattice-estimator estimates; they are not the 192-bit level claimed for the $q \approx 2^{27}$ parameters in the original analysis of [36], which relied on the older root-Hermite-factor heuristic. The larger modulus is what makes the deterministic bounds above fit, and the price is the drop from 2^{118} to $2^{110.1}$. The estimate is unaffected by (M1): the attacks depend on the typical norm $\approx 36.95 < B_x$ and the dimension, not on the removed tail (Theorem B.3). The key-recovery problem for (f, g) has a root-Hermite factor $\gamma \approx (\sqrt{N}/1.368)^{1/(2N)} \approx 1.0015$, which by the analysis of [36] does not depend on q and is far from practical.

Deterministic Decryption Bound. For $a, b \in R = \mathbb{Z}[x]/(x^N + 1)$, the i-th coefficient of $a * b$ is $\sum_{j \le i} a_j b_{i-j} - \sum_{j>i} a_j b_{N+i-j} = \langle a, \pi_i(b) \rangle$, where π_i is a signed permutation of the coordinates of b, so $\|\pi_i(b)\|_2 = \|b\|_2$.

Lemma B.3 (Decryption bound)**.** *Let* $\mathsf{mpk} = h \in R_q$ *be arbitrary and* ID *be any identity, with* $t = H(\mathsf{ID})$*. Let* $s_2 \in R_q$ *be such that* $\mathsf{VerKey}(h, \mathsf{ID}, s_2) = 1$ *and let* $s_1 := [t - s_2 * h]_q$*. Let* (u, v, c') *be the output of* $\mathsf{Enc}(h, \mathsf{ID}, m)$ *with randomness* (r, e_1, e_2, k) *(so* $\|(r, e_1)\|_2 \le B_x$ *and* $\|e_2\|_\infty \le 1$*). Then* $w := [v - u * s_2]_q$ *satisfies* $w = (q/2)k + E \bmod q$ *with* $\|E\|_\infty \le B_x \Gamma_{\mathsf{key}} + 1 + 2^{\ell-1} < q/4$.

Proof. Let $v_0 = r * t + e_2 + (q/2)k$ and let $v = v_0 + \rho$ be the compressed value, with $\|\rho\|_\infty \le 2^{\ell-1}$ (since $2^\ell \mid q$ there is no wrap-around). In R_q, using $s_1 \equiv t - s_2 * h$,

$$\begin{aligned} v - u * s_2 &= r * (t - h * s_2) - e_1 * s_2 + e_2 + \rho + (q/2)k \\ &= (q/2)k + \underbrace{r * s_1 - e_1 * s_2 + e_2 + \rho}_{=:E}. \end{aligned}$$

For each coordinate i, $(r * s_1 - e_1 * s_2)_i = \langle (r, e_1), (\pi_i(s_1), -\pi_i(s_2)) \rangle$, so by Cauchy–Schwarz $|(r * s_1 - e_1 * s_2)_i| \le \|(r, e_1)\|_2 \, \|(s_1, s_2)\|_2 \le B_x \Gamma_{\mathsf{key}}$. With $|(e_2)_i| \le 1$ and $|\rho_i| \le 2^{\ell-1}$ we get $|E_i| \le B_x \Gamma_{\mathsf{key}} + 1 + 2^{\ell-1} = 133{,}188{,}609 < q/4$. □

Corollary B.1. *Under the assumptions of Lemma B.3, the following holds* $\mathsf{Dec}(s_2, (u, v, c')) = m$.

Proof. For each i, if $k_i = 0$ then $|w_i| = |E_i| < q/4$, and if $k_i = 1$ then $|w_i| \ge q/2 - |E_i| > q/4$. Hence Dec recovers k exactly and outputs $c' \oplus H'(k) = m$. □

Theorem B.1 (Deterministic soundness / key verifiability)**.** *Let* $\mathsf{mpk} = h \in R_q$ *be arbitrary (possibly maliciously generated) and let* ID *be any identity. For every* m*, every* $\mathsf{sk}_{\mathsf{ID}}$ *with* $\mathsf{VerKey}(h, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) = 1$*, and every* c *output by* $\mathsf{Enc}(h, \mathsf{ID}, m)$ *of IBE,* $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) = m$*. In particular, the key-verifiability game is won with probability exactly* 0.

Proof. Immediate from Corollary B.1: the only property of h and $\mathsf{sk}_{\mathsf{ID}}$ that is used is the norm test $\|(s_1, s_2)\|_2 \le \Gamma_{\mathsf{key}}$ performed by VerKey, and the only property of the ciphertext is the bound $\|(r, e_1)\|_2 \le B_x$ enforced by Enc. The adversary may choose h and $\mathsf{sk}_{\mathsf{ID}}$ after seeing c, but this cannot change the inequality. □

Theorem B.2 (Completeness)**.** *Let* $(h, B) \leftarrow \mathsf{KeyGen}(N, q)$ *(so that* $\|\tilde{B}\| \le 1.17\sqrt{q}$*) and let* ID *be any identity. Then* $\mathsf{Extract}(B, \mathsf{ID})$ *terminates with probability* 1*: each iteration of its loop is rejected independently with probability* $p < 2^{-201}$*, so the number of iterations is geometric with expectation* $1/(1-p)$*. Every key* $\mathsf{sk}_{\mathsf{ID}}$ *that it outputs satisfies* $\mathsf{VerKey}(h, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) = 1$*. Moreover, for every such key, every message* m *and every ciphertext* c *in the support of* $\mathsf{Enc}(h, \mathsf{ID}, m)$ *(i.e. for every outcome of the encryption randomness),* $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) = m$*. In particular the correctness error of IBE is exactly* 0.

Proof. Every key output by Extract has passed the test of VerKey (evaluated exactly on integers), hence VerKey = 1. By the tail bound with $\Gamma_{\mathsf{key}} = 3{,}120{,}000$ (which needs $\|\tilde{B}\| \le 1.17\sqrt{q}$, enforced by KeyGen), one iteration is rejected with probability $p < 2^{-201}$, so the number of iterations is finite with probability 1. Likewise the resampling loop of Enc terminates with probability 1 (it is rejected with probability $2^{-99.1}$). Decryption correctness is Corollary B.1, which holds for every key that passes VerKey and every (r, e_1, e_2, k) with $\|(r, e_1)\|_2 \le B_x$. □

Corollary B.2 (Malicious dealer)**.** *Let* $\mathsf{sk}_{\mathsf{ID}}$ *be the honestly extracted key. Then for any ciphertext* c *chosen by a corrupt dealer,* $\mathsf{VerKey}(h, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}) = 1$ *regardless of* c*. If* c *is an honest encryption of* m *then* $\mathsf{Dec}(\mathsf{sk}_{\mathsf{ID}}, c) = m$*. If* c *is not an honest encryption (including malformed or junk* c*),* Dec *outputs whatever the deterministic algorithm computes on* c *(*$\perp$ *if malformed). This output is a deterministic function of* $(\mathsf{sk}_{\mathsf{ID}}, c)$*, so the receiver and any public verifier running* Dec *with the same* $\mathsf{sk}_{\mathsf{ID}}$ *obtain the same value; if it differs from the dealer's committed value, the complaint is valid. A corrupt dealer can therefore never cause an honest receiver's key to fail* VerKey.

Security of the Modified DLP IBE.. The modifications (M1) and (M2) change the experiment of [36] in two places: the challenge randomness (r, e_1) and the distribution of the keys returned by the extraction oracle. We account for both explicitly. For an identity with $t = H(\mathsf{ID})$, let $\mathcal{D}_t$ be the ideal discrete Gaussian $D_{\Lambda-(t,0),\sigma}$, $\mathcal{K}_t$ the output distribution of one call of the Klein sampler $(t,0) - \mathsf{GaussianSampler}(B, \sigma, (t,0))$, and $\mathcal{K}_t^{\mathsf{ver}}$ the output of Extract, i.e. $\mathcal{K}_t$ conditioned on $\|(s_1, s_2)\|_2 \le \Gamma_{\mathsf{key}}$.

Let $D_{\mathrm{KL}}(P\|Q) = \sum_x P(x)\ln(P(x)/Q(x))$.

Lemma B.4 (Extraction oracle)**.** *For every* t*: (a)* $D_{\mathrm{KL}}(\mathcal{D}_t \,\|\, \mathcal{K}_t) \le 2^{-\lambda} = 2^{-192}$*; (b)* $p_t := \Pr_{\mathcal{K}_t}[\|(s_1, s_2)\|_2 > \Gamma_{\mathsf{key}}] < 2^{-201}$ *and* $D_{\mathrm{KL}}(\mathcal{K}_t^{\mathsf{ver}} \,\|\, \mathcal{K}_t) = -\ln(1 - p_t) < 2^{-200}$.

Proof. (a) $\mathcal{D}_t$ and $\mathcal{K}_t$ are both supported on the whole coset $\Lambda - (t,0)$, and the Klein sampler has relative error at most $\delta = \left|1 - \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^n\right|$ with respect to $\mathcal{D}_t$ whenever $\sigma \ge \eta'_\varepsilon(\mathbb{Z})\|\tilde{B}\|$ (proof of [36, Theorem 2]). As in [36, Theorem 2 and Lemma 5], with the ideal distribution in the first argument, this relative-error bound gives $D_{\mathrm{KL}}(\mathcal{D}_t\|\mathcal{K}_t) \le 2\left(1 - \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^n\right)^2 \approx 8n^2\varepsilon^2 = 2^{-193} \le 2^{-192}$ for $n = 2N = 2048$ and $\varepsilon = 2^{-109}$ (the relative error is $\delta \approx 2n\varepsilon = 2^{-97}$). (b) The first claim is the tail bound of the parameter derivation, transferred from $\mathcal{D}_t$ to $\mathcal{K}_t$ through the relative error δ. For the second, on the accepted set A we have $\mathcal{K}_t^{\mathsf{ver}}(x) = \mathcal{K}_t(x)/(1-p_t)$, so $D_{\mathrm{KL}}(\mathcal{K}_t^{\mathsf{ver}}\|\mathcal{K}_t) = \sum_{x\in A}\frac{\mathcal{K}_t(x)}{1-p_t}\ln\frac{1}{1-p_t} = -\ln(1-p_t)$. Here the direction is forced: $\mathcal{K}_t^{\mathsf{ver}}$ is supported on a subset of the support of $\mathcal{K}_t$, so $D_{\mathrm{KL}}(\mathcal{K}_t\|\mathcal{K}_t^{\mathsf{ver}}) = \infty$. □

We use the following tool from [36] (Lemma 5, due to [58]). Let $\mathcal{E}_P$ be an algorithm making at most Q queries to an oracle that samples from P, and let Q' be a distribution with $D_{\mathrm{KL}}(P \,\|\, Q') \le \delta$. Then the probabilities x (oracle P) and y (oracle Q') that the algorithm outputs 1 satisfy $|x - y| \le \frac{1}{\sqrt{2}}\sqrt{Q\delta}$. The *first* argument of the divergence is the distribution of the oracle that the algorithm is originally run with: in [36] this is the ideal oracle P, and Q' is the oracle that is actually implemented. The proof uses only additivity of the divergence over the Q samples (chain rule) and Pinsker's inequality, hence it holds unchanged for adaptively chosen, *distinct* distributions $P_1, \ldots, P_Q$ with $D_{\mathrm{KL}}(P_i\|Q'_i) \le \delta$ each. This is the form needed here, since each extraction query has its own $t = H(\mathsf{ID})$. Over Q_{ext} extraction queries the total divergence of the transcript is at most $Q_{\mathsf{ext}}\delta$.

Lemma B.5 (CPA security reduces to recovering k)**.** *In the random oracle model for* H'*, let* $\mathcal{A}$ *be an IND-ID-CPA adversary against IBE and let* X *be the event that* $\mathcal{A}$ *queries* H' *at the key* k^* *encapsulated in the challenge ciphertext. Then* $\left|\Pr[b' = b] - \frac{1}{2}\right| \le \Pr[X]$.

Proof. Consider the experiment in which $c'^* := m_b \oplus H'(k^*)$ is replaced by a uniform string. Until X occurs, $H'(k^*)$ is uniform and independent of the adversary's view, so the two experiments are identical unless X occurs. In the modified experiment b is independent of the view, so $\Pr[b' = b] = \frac{1}{2}$. The difference lemma gives the claim. □

Theorem B.3 (Security of IBE relative to [36])**.** *Let* $\mathcal{A}$ *be an adversary against IBE making at most* Q_{ext} *extraction queries, and let* X *be as in Lemma B.5. Consider four experiments:* G_0 *is the experiment for IBE;* G_1 *is* G_0 *with the challenge randomness* (r, e_1) *uniform (no conditioning (M1));* G_2 *is* G_1 *with the extraction oracle returning* $\mathcal{K}_t$ *(no rejection (M2)); and* G_3 *is* G_2 *with the oracle returning the ideal distribution* $\mathcal{D}_t$ *(the experiment analyzed in [36]). Then*

$$\Pr_{G_0}[X] \le \frac{\Pr_{G_1}[X]}{1 - 2^{-99.1}}, \qquad \left|\Pr_{G_1}[X] - \Pr_{G_2}[X]\right| \le \sqrt{Q_{\mathsf{ext}}\, 2^{-200}/2},$$

$$\left|\Pr_{G_2}[X] - \Pr_{G_3}[X]\right| \le \sqrt{Q_{\mathsf{ext}}\, 2^{-192}/2}.$$

Consequently, for $Q_{\mathsf{ext}} \le 2^{110}$*,*

$$\Pr_{G_3}[X] \;\ge\; (1 - 2^{-99.1})\Pr_{G_0}[X] - 2^{-41}.$$

Proof. *First inequality (M1).* (r, e_1) is used only in the challenge ciphertext and is independent of everything else in the experiment, and the event $\mathcal{E} = \{\mathsf{wt}(r) + \mathsf{wt}(e_1) \le 1600\}$ depends only on (r, e_1). Hence the whole of G_0 is distributed as G_1 conditioned on $\mathcal{E}$, and $\Pr_{G_0}[X] = \Pr_{G_1}[X \wedge \mathcal{E}]/\Pr_{G_1}[\mathcal{E}] \le \Pr_{G_1}[X]/(1 - 2^{-99.1})$.

Second inequality (M2). Let the algorithm $\mathcal{B}$ run G_1 and output 1 iff X occurs. By Lemma B.4(b), apply the tool above with $P_i = \mathcal{K}_{t_i}^{\mathsf{ver}}$ (the oracle of G_1), $Q'_i = \mathcal{K}_{t_i}$ (the oracle of G_2) and $\delta = 2^{-200}$. (If an identity is queried repeatedly, the stored key is returned and no new sample is drawn, so at most Q_{ext} samples are used.)

Third inequality. Let $\mathcal{B}'$ be the algorithm that runs the experiment with the ideal extraction oracle $P_i = \mathcal{D}_{t_i}$ (this is G_3) and outputs 1 iff X occurs. By Lemma B.4(a), $D_{\mathrm{KL}}(\mathcal{D}_{t_i}\|\mathcal{K}_{t_i}) \le 2^{-192}$, so the tool above with $P_i = \mathcal{D}_{t_i}$, $Q'_i = \mathcal{K}_{t_i}$ (the implemented Klein sampler, G_2) and $\delta = 2^{-192}$ gives $|\Pr_{G_3}[X] - \Pr_{G_2}[X]| \le \sqrt{Q_{\mathsf{ext}} 2^{-192}/2}$.

For the consequence, the two additive terms are at most $2^{-45.5}$ and $2^{-41.5}$ when $Q_{\mathsf{ext}} \le 2^{110}$, and their sum is below 2^{-41}. □

Why Restricting (r, e_1) *to* $B_x = 40$ *Costs Only* 2^{-99}. Write $\mathsf{Enc}_{40} = \mathsf{Enc}_{\mathsf{orig}} \mid \mathcal{E}$ with $\Pr[\neg\mathcal{E}] = 2^{-99.1}$. This has four consequences.

(1) *Statistical distance.* The probability of the internal tail event, i.e. of a resampling, is exactly $\Pr[W > 1600] = 2^{-99.1}$, while the induced statistical distance between the two encryption distributions satisfies $\Delta(\mathsf{Enc}_{40}, \mathsf{Enc}_{\mathsf{orig}}) \le \Pr[\neg\mathcal{E}] = 2^{-99.1}$ (it need not be equal, since different randomness values can

map to the same ciphertext). In any experiment that uses (r, e_1) once (the challenge), the advantage of any adversary, whatever its running time, changes by at most 2^{-99} additively.

(2) *The tail is the only failure.* For the unconditioned $\mathsf{Enc}_{\mathrm{orig}}$, the deterministic decryption bound of Lemma B.3 can fail only on the tail event $\neg\mathcal{E}$, which has probability $2^{-99.1}$.

(3) *Multiplicative loss for the search problem.* By Lemma B.5, the quantity that governs security is $\Pr[X]$, which is a search event, and the first inequality of Thm. B.3 shows $\Pr_{\mathsf{Enc}_{40}}[X] \le \Pr_{\mathsf{Enc}_{\mathrm{orig}}}[X]/(1-2^{-99.1})$: an attack on the modified scheme with success probability p yields an attack on the original with probability at least $(1-2^{-99.1})p$ at the same cost. This loses a factor $1+2^{-99}$ and no bits of security, whereas the additive bound of item 1 alone would be too weak for a target of 2^{-110}. The remaining additive terms of Theorem B.3 are of the order 2^{-41} and are absorbed in the convention of [36, Sec. 4], where an attack of cost $2^{\lambda'}$ is an algorithm of running time $\le 2^{\lambda'}$ with success probability $\ge 3/4$.

(4) *Estimator and timing.* The estimate of Sec. B.3 depends on the typical norm $\approx 36.95 < B_x$ and on the dimension, not on the removed tail. The number of iterations of the resampling loop is 1 except with probability $2^{-99.1}$, so the timing leakage of Enc about the randomness is at most $2^{-99.1}$.

Scope of the claim. As in [36, Sec. 4], we identify the cost of the event X with the cost of recovering k that the lattice estimator evaluates. A generic reduction from the list of H'-queries to a single candidate k would lose a factor $Q_{H'}$; we do not claim anything beyond the heuristic estimate of Sec. B.3.

Similarly, it can be shown that the Boneh-Franklin IBE scheme [20] also satisfies the key-verifiability property.

In the security proofs of our PVSS protocols, we also argue the adaptive security of the schemes. For that, we need the IBE encryptions to be non-committing [25], i.e., a simulated ciphertext can be later opened consistently as an encryption of any message.

Non-Committing Encryption. We use IBE in the programmable random oracle model for H'. The simulator $\mathsf{Sim}_{\mathsf{Enc}}(h, \mathsf{ID})$ runs the code of Enc (including (M1)) with a uniformly random k, but outputs a uniformly random $c' \in \{0,1\}^m$ instead of $m \oplus H'(k)$, and stores k. To open the ciphertext to a message m, the simulator programs $H'(k) := c' \oplus m$; it aborts if $H'(k)$ has already been defined. Let X be the event that the adversary queries H' at k before the opening. Until X occurs, $H'(k)$ is uniform and independent of the adversary's view, so $m \oplus H'(k)$ is uniform and the real and simulated ciphertexts have the same distribution; after the opening, the programmed value is consistent with c'. Hence the real and simulated views differ with probability at most $\Pr[X]$. The simulated (u, v) has exactly the distribution of the challenge ciphertext of IBE, so the bound of Theorem B.3 on $\Pr[X]$ applies verbatim: an adversary that makes X happen recovers the encapsulated key k. We require that the key $\mathsf{sk}_{\mathsf{ID}}$ of the identity is not released to the adversary before the ciphertexts addressed to ID are opened (otherwise the adversary can decrypt k honestly); this is the situation in our adaptive proofs, where the key is revealed only upon corruption, at which point the simulator knows the dealt share and programs H'.

Post-Quantum Secure IBE.. We say that an IBE scheme is post-quantum secure if all of the above properties hold against an adversary with quantum capabilities. The security of IBE is derived from that of the IBE of [36] (Theorem B.3), which relies on the Ring-LWE [52] and NTRU assumptions. The deterministic properties (Theorems B.1 and B.2) are information-theoretic and hold against quantum adversaries. The reduction in Theorem B.3 is stated in the classical random oracle model; a quantum random-oracle analysis is not provided here.

B.4 Shamir's Secret Sharing Scheme

We recall the definition of an (n, t) Shamir secret sharing scheme as follows:

Definition B.1 (Shamir's Secret Sharing Scheme)**.** Let $\mathbb{F}$ be a finite field. It consists of two algorithms:

- $\mathsf{Share}(s, n, t)$: The dealer selects a random t-degree polynomial

$$f(x) := s + a_1 x + a_2 x^2 + \cdots + a_t x^t \in \mathbb{F}[x],$$

  where the coefficients $a_1, \ldots, a_t \in \mathbb{F}$ are chosen uniformly at random. For each party $i \in [n]$, the dealer computes the share $[\![s]\!]_i = f(i) \in \mathbb{F}$. The dealer outputs $(\{[\![s]\!]_i\}_{i\in[n]})$.
- $\mathsf{Recon}(z, \{i_j, [\![s]\!]_{i_j}\}_{j=1}^{t+1})$: Given any $t+1$ shares, Recon reconstructs $f(z)$ as:

$$f(z) = \sum_{j=1}^{t+1} [\![s]\!]_{i_j} \cdot \lambda_j,$$

  where $\lambda_j(z) = \prod_{\substack{1\le m\le t+1 \\ m\ne j}} \frac{z-i_m}{i_j-i_m}$ are the Lagrange interpolation coefficients evaluated at $x = z$.
- $\mathsf{ReconPoly}(\{i_j, [\![s]\!]_{i_j}\}_{j=1}^{t+1})$: Given any $t+1$ shares, ReconPoly reconstructs the polynomial $f(x)$ as:

$$f(x) = \sum_{j=1}^{t+1} [\![s]\!]_{i_j} \cdot \lambda_j,$$

  where $\lambda_j(x) = \prod_{\substack{1\le m\le t+1 \\ m\ne j}} \frac{x-i_m}{i_j-i_m}$.

The scheme satisfies the following properties:

- **Correctness:** Any set of $t+1$ or more valid shares correctly reconstructs the secret s.
- **Perfect Secrecy:** Any set $\le t$ shares information theoretically hides s.

Shamir sharing scheme also satisfies reconstruction over additively homomorphic commitment schemes defined over message space field $\mathcal{M}$ and randomness space field $\mathcal{R}$ as follows:

- $\mathsf{ShareCom}(n, t)$: Samples a **random secret** $s \in \mathcal{M}$ and randomness $r \in \mathcal{R}$, the dealer selects two random t-degree polynomial

$$f(x) := s + a_1 x + a_2 x^2 + \cdots + a_t x^t \in \mathcal{M}[x],$$

$$g(x) := r + b_1 x + b_2 x^2 + \cdots + b_t x^t \in \mathcal{R}[x]$$

where $a_1, \ldots, a_t \in \mathcal{M}$ and $b_1, \ldots, b_t \in \mathcal{R}$ are chosen uniformly at random. For each party $i \in [n]$, the dealer computes the share $[\![s]\!]_i := f(i) \in \mathcal{M}$ and $[\![r]\!]_i := g(i) \in \mathcal{R}$, and the committed share $\langle [\![s]\!]_i; [\![r]\!]_i \rangle$. The dealer outputs $(s, r, \{[\![s]\!]_i, [\![r]\!]_i, \langle [\![s]\!]_i; [\![r]\!]_i \rangle\}_{i\in[n]})$.

- $\mathsf{ShareCom}(s, n, t)$: Given a **secret** $s \in \mathcal{M}$ and randomness $r \in \mathcal{R}$, the dealer selects two random t-degree polynomial

$$f(x) := s + a_1 x + a_2 x^2 + \cdots + a_t x^t \in \mathcal{M}[x],$$

$$g(x) := r + b_1 x + b_2 x^2 + \cdots + b_t x^t \in \mathcal{R}[x]$$

where $a_1, \ldots, a_t \in \mathcal{M}$ and $b_1, \ldots, b_t \in \mathcal{R}$ are chosen uniformly at random. For each party $i \in [n]$, the dealer computes the share $[\![s]\!]_i := f(i) \in \mathcal{M}$ and $[\![r]\!]_i := g(i) \in \mathcal{R}$, and the committed share $\langle [\![s]\!]_i; [\![r]\!]_i \rangle$. The dealer outputs $(s, r, \{[\![s]\!]_i, [\![r]\!]_i, \langle [\![s]\!]_i; [\![r]\!]_i \rangle\}_{i\in[n]})$.

- $\mathsf{ReconCom}(z, \{i_j, \langle [\![s]\!]_{i_j}; [\![r]\!]_{i_j} \rangle\}_{j=1}^{t+1})$: Given any $t+1$ shares, Recon reconstructs the committed secret $\langle s; r \rangle$:

$$\langle s; r \rangle = \prod_{j=1}^{t+1} \langle [\![s]\!]_{i_j}; [\![r]\!]_{i_j} \rangle^{\lambda_j(z)}.$$

C LONG-LASTING SECURITY MODEL

A protocol is said to be long-lasting secure if it is secure against a PPT classical adversary *during* the protocol execution. And after the protocol terminates, the protocol transcript would remain secure (does not leak any information about honest parties' inputs) against a future PPT quantum adversary.

Real-World: In the real-world execution, the protocol π is carried out between the honest parties $\{P_i\}_{i\in[n]}$ and the real-world adversary $\mathcal{A}$. All the parties are classical PPT Turing machines, and $\mathcal{A}$ has auxiliary information z. At the outset of the protocol, each party P_i has its own input x_i. At the outset of the protocol, $\mathcal{A}$ may or may not corrupt a party. Upon corruption of a party, $\mathcal{A}$ gets access to the internal state and input of that party. From now on, the party will behave according to $\mathcal{A}$'s instructions (since we are in the malicious model). At the end of the protocol, the honest parties produce their outputs. The view of $\mathcal{A}$ consists of the internal states of the corrupt parties and their outputs. We denote the real-world adversarial view as $\mathsf{Real}_{\pi,\mathcal{A},}(1^\lambda, z)$, and it consists of the honest parties' outputs, the corrupt parties' outputs, and the internal states of the corrupt parties.

Ideal-World: In the ideal-world execution, we consider the honest parties $\{P_i\}_{i\in[n]}$, the ideal-world adversary Sim and the functionality $\mathcal{F}$. Sim has access to the real-world adversarial algorithm $\mathcal{A}$ and the protocol π. All the parties are classical PPT Turing machines. Sim has a random tape r and security parameter λ. It simulates the role of $\mathcal{A}$ in the ideal world, and whenever $\mathcal{A}$ corrupts a party in the real world Sim corrupts that party in the ideal world and gets access to its internal state. Sim invokes the algorithm of $\mathcal{A}$, in his head, in another internal protocol execution where Sim simulates the view of the honest parties to $\mathcal{A}$. We will denote this internal copy of $\mathcal{A}$ as $\mathcal{A}_{\mathsf{Int}}$. Based on the reply of $\mathcal{A}_{\mathsf{Int}}$ in the internal execution, Sim behaves accordingly in the ideal world execution. He extracts the inputs of the corrupted parties in the internal execution and invokes $\mathcal{F}$ in the ideal world with those inputs to obtain the output. In the internal execution he simulates the protocol in such a way that $\mathcal{A}_{\mathsf{Int}}$ obtains that output. At the end of the protocol, the view of $\mathcal{A}_{\mathsf{Int}}$ consists of the internal states of the corrupt parties and their outputs. We denote the ideal-world adversarial view as $\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(1^\lambda, z)$, and it consists of the honest parties' outputs, the corrupt parties' outputs, and the internal states of the corrupt parties.

Definition C.1. Let π be a protocol for computing a functionality $\mathcal{F}$. We say that π computes functionality $\mathcal{F}$ with long-lasting security in the presence of static adversaries if for every PPT classical static real-world adversary $\mathcal{A}$, there exists a PPT classical ideal-world adversary Sim, such that:

$$\mathsf{Real}_{\pi,\mathcal{A}}(1^\lambda, z) \overset{\mathsf{pq}}{\approx} \mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(1^\lambda, z).$$

D SECURITY PROOFS

In this section, we provide the security proofs of our PVSS protocols.

D.1 Security Proof of Theorem 5.1

We present the security proof of our post-quantum secure PVSS protocol as follows.

Proof. **Correctness** of the decrypted shares follows from the correctness of the Shamir secret sharing scheme and the correct decryption of the IBE scheme. If the dealer is honest, then each $[\![s]\!]_i$ and $[\![r]\!]_i$ is a correct sharing of s and r and the evaluations of $B(i)$ are committed for $i \in [n]$ where $W(x) := B(x) - \mathsf{ch} \cdot f(x)$ and $[\![s]\!]_i := f(i)$. Each receiver checks that the hash of $W(i) + \mathsf{ch} \cdot [\![s]\!]_i$ is the same as the hash of $B(i)$ under randomness $q(i)$, and this always holds.

The PVSS satisfies **non-interactive dealing** since the dealer does not need to participate in the PVSS once it has dealt the shares non-interactively using the Share algorithm.

The PVSS transcript is also **publicly verifiable** in the random oracle model, as any external party can publicly verify a raised complaint $(\textsc{Complaint}, i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID},i})$ by decrypting $([\![s]\!]_i, [\![r]\!]_i, q_i) := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID},i}, \mathsf{E}_i)$ and checking it against the proof π.

Reconstructability follows, because if the verification succeeds, then the only way an invalid dealing goes undetected is if the dealer encrypts shares which does not correspond to a valid t-degree sharing of a polynomial, and yet none of the $n - t$ honest receivers raised a complaint. This only occurs if $\mathrm{H}(W(i) + \mathsf{ch} \cdot \widetilde{[\![s]\!]_i}, \widetilde{q_i}) == \mathrm{H}(B(i), q_i)$ where $(\widetilde{[\![s]\!]_i}, \widetilde{q_i})$ are sent by the malicious dealer to the i-th honest receiver and $\widetilde{[\![s]\!]_i} \neq [\![s]\!]_i$. This requires the adversarial dealer to either predict ch or find a collision in H, both of which occur with negligible probability when H is modeled as a random oracle. All the n shares are committed, and whenever $t+1$ receivers provide their openings $\{i_j, [\![s]\!]_{i_j}, [\![r]\!]_{i_j}, q_{i_j}\}_{j\in[t+1]}$ it can be checked that the provided openings are correct against the commitments.

For **liveness**, the adversary is allowed to corrupt up to $t < \frac{n}{2}$ since reconstructing the secret requires $t + 1$ shares from honest receivers.

It can be observed that the dealing remains hidden from a set of t adversarial parties as the dealer uses a t-degree sharing and the adversary lacks one share. The commitments csh_i, c and cpt_i hide

the underlying committed shares since H is a random oracle, and so it is preimage resistance. And the encryptions E_i hide the values $([\![s]\!]_i, [\![r]\!]_i, q_i)$ due to the IND-ID-CPA Security of the IBE scheme. The commitments are instantiated using the random oracle and the IBE is post-quantum secure, and hence the scheme achieves **post-quantum secrecy**. □

On Adaptive Security: Our PVSS protocol can be proven to be adaptively secure by assuming the random oracle is programmable and the IBE is non-committing by using a programmable random oracle. In our PVSS, the simulator simulates an honest dealing by sampling two random t-degree polynomials for sharing a dummy secret and randomness r, and committing the shares as csh_i using the random oracle and encrypting those shares using the IBE. When the i-th receiver gets corrupted, the simulator obtains its share and the simulator equivocates csh_i to open the share by programming the oracle and programming the IBE to open to it. Equivocation succeeds since $[\![r]\!]_i$ remains hidden from the adversary without corrupting the i-th party and hence the simulator can successfully equivocate csh_i. The same argument holds for equivocating the commitment cpt_i and its opening $q(i)$. The encryptions are equivocated based on the non-committing property of the IBE.

D.2 Security Proof of Theorem 6.1

We present the security proof of our long-lasting secure PVSS protocol as follows.

Proof. **Correctness** of the decrypted shares follows from the correctness of the Shamir secret sharing scheme and the correct decryption of the IBE scheme. If the dealer is honest, then each $[\![s]\!]_i$ and $[\![r]\!]_i$ is a correct sharing of s and r, and each i-th commitment csh_i is a commitment to $[\![s]\!]_i$ using randomness $[\![r]\!]_i$. Each receiver checks that the commitments pass the SCRAPE test, using the homomorphic property of Com and the correctness of the SCRAPE test, and the commitments correctly decommit to $[\![s]\!]_i$ and $[\![r]\!]_i$ by relying on the correctness of the commitment.

The PVSS satisfies **non-interactive dealing** since the dealer does not need to participate in the PVSS once it has dealt the shares non-interactively using the Share algorithm.

The PVSS transcript is also **publicly verifiable** in the random oracle model, as any external party can publicly verify a raised complaint $(\textsc{Complaint}, i, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID},i})$ by decrypting $([\![s]\!]_i, [\![r]\!]_i) := \mathsf{Dec}(\mathsf{sk}_{\mathsf{ID},i}, \mathsf{E}_i)$ and checking it against the provided commitment csh_i to server's share.

Reconstructability follows, because if the dealing passes the SCRAPE test, then the only way an invalid dealing goes undetected is if the commitments are not to a degree t-sharing, or the dealer encrypts shares that do not correspond to a valid t-degree sharing of a polynomial, and yet none of the $n - t$ honest receivers raised a complaint. The former case occurs with negligible probability due to the soundness of the SCRAPE test B.1. The second case occurs only if the commitments pass the SCRAPE test, but the dealer opens the commitments to shares which do not fall on the same degree t polynomial. However, this requires breaking the binding of the commitment scheme, which occurs with negligible probability.

For **liveness**, the adversary is allowed to corrupt up to $t < \frac{n}{2}$ since reconstructing the secret requires $t + 1$ shares from honest receivers.

It can be observed that the dealing remains hidden from a set of t adversarial parties as the dealer uses a t-degree sharing and the adversary lacks one share. The commitments csh_i and c perfectly hide the underlying committed shares. And the encryptions E_i hide the values $([\![s]\!]_i, [\![r]\!]_i)$ due to the IND-ID-CPA Security of the IBE scheme. Hence, the scheme achieves **long-lasting secrecy** as the encryptions remain post-quantum secure in the future and commitments are perfectly hiding. □

On Adaptive Security: Our PVSS protocol can be proven to be adaptively secure by assuming that Com is adaptively secure and the IBE is non-committing (by using a programmable random oracle). In our PVSS, the simulator simulates an honest dealing by sampling two random t-degree polynomials for sharing a dummy secret and randomness r, and committing the shares as csh_i using Com and encrypting those shares using the IBE. When the i-th receiver gets corrupted, the simulator obtains its share and the simulator equivocates csh_i to open the share by equivocating Com and equivocating the IBE to open to it and the equivocated randomness. Equivocation succeeds due to the adaptive security of Com and the non-committing property of the IBE.

E ADDITIONAL CRYPTOGRAPHIC DETAILS

This section presents the detailed NIZK protocols used in the polls. We use Pedersen Commitment to instantiate Com. We recall them below.

E.1 Proof of Correct Commitment to the Vote

The client proves that it voted 1 for a single option (among four options) and zero for the other options by proving the following:

- **Input:** $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{c}_4, \mathsf{vk}, \mathsf{pid})$
- **Witness:** $(\mathsf{v}_1, \mathsf{r}_1, \mathsf{v}_2, \mathsf{r}_2, \mathsf{v}_3, \mathsf{r}_3, \mathsf{v}_4, \mathsf{r}_4)$
- **Statement:** $(\mathsf{c}_j = \mathsf{Com}(\mathsf{v}_j; \mathsf{r}_j)) \ \ \& \ \ (\mathsf{c}_j = \mathsf{Com}(0; \mathsf{r}_j) \vee \mathsf{c}_j = \mathsf{Com}(1; \mathsf{r}_j)) \ \ \& \ \ \mathsf{c}_1 + \mathsf{c}_2 + \mathsf{c}_3 + \mathsf{c}_4 = \mathsf{Com}(1) \ \forall j \in [4]$.
- $\mathsf{Prove}_{\mathsf{vote}}$: The prover runs the following:
  (1) For each $j \in [4]$: Run $\pi_j := \mathsf{Prove}_{\mathsf{set}}((\mathsf{c}_j, \{0, 1\}), (\mathsf{v}_j, \mathsf{r}_j))$.
  (2) Sample $c' := \mathsf{Com}(0; r')$. Run $\pi' := \mathsf{Prove}_{\mathsf{set}}((C', \{0\}), (0, r'))$.
  (3) Compute $R := \sum_{j \in [4]} \mathsf{r}_j + r'$.
  (4) Compute proof $\pi_{\mathsf{vote}} := (\{\pi_j\}_{j \in [4]}, c', \pi', R)$.
- $\mathsf{Verify}_{\mathsf{vote}}$: The verifier runs the following:
  (1) For each $j \in [4]$: Check $\mathsf{Verify}_{\mathsf{set}}((\mathsf{c}_j, \{0, 1\}), \pi_j) = 1$.
  (2) Check $\mathsf{Verify}_{\mathsf{set}}((c', \{0\}), \pi') = 1$.
  (3) Check $\mathsf{Com}(1; R) = \Pi_{j \in [4]} \mathsf{c}_j \cdot c'$.
  If any of the above checks fail then output 0, otherwise, output 1.

Verifying the above three conditions suffices to check that the vote is correctly committed. The first condition ensures that each commitment is a bit commitment. The second and third checks ensure that the sum of all the committed bits is 1. Combining the two arguments, we ensure that only one of the four options was voted with a 1, and the other three were voted 0. We use the CDS-OR proof system from [30] to instantiate the set membership proof.

We provide the pid and vk as part of the Fiat-Shamir hash function input.

E.2 Proof of Set Membership of a Committed Element

We use the CDS OR proof [30], instantiated with Schnorr proofs, to prove that a value committed using a Pedersen commitment belongs to a public set. We apply the Fiat–Shamir transformation to obtain a non-interactive proof. We follow the presentation in [21, Chapter 19].

Let $C = \mathsf{Com}(x; r) = g^x h^r$ be a Pedersen commitment and let $X = \{x_1, \ldots, x_N\}$ be a public set. For every $i \in [N]$, define

$$Y_i := C/g^{x_i}.$$

Notice that

$$x = x_i \quad \Longrightarrow \quad Y_i = h^r.$$

Thus, for each $i \in [N]$, consider the Schnorr relation

$$R_i(Y_i, r_i) : \qquad Y_i = h^{r_i}.$$

The set-membership statement can therefore be written as

$$\bigvee_{i \in [N]} R_i(Y_i, r_i).$$

- **Input:** $(C, X, \mathsf{vk}, \mathsf{pid})$, where $C = \mathsf{Com}(x; r) = g^x h^r$ and $X = \{x_1, \ldots, x_N\}$.
- **Witness:** (x_j, r) for some $j \in [N]$ such that
$$C = g^{x_j} h^r.$$
- **Statement:**
$$\bigvee_{i \in [N]} \left(C = g^{x_i} h^{r_i}\right).$$
Equivalently,
$$\bigvee_{i \in [N]} \left(Y_i = h^{r_i}\right), \qquad Y_i := C/g^{x_i}.$$
- $\mathsf{Prove}_{\mathsf{set}}$: The prover runs the following:
  (1) Let $j \in [N]$ be such that $x = x_j$. For every $i \in [N] \setminus \{j\}$, sample $c_i, z_i \leftarrow \mathbb{Z}_q$ and compute the simulated Schnorr first message
  $$T_i := h^{z_i} \left(C/g^{x_i}\right)^{-c_i}.$$
  (2) For the true relation j, sample $u_j \leftarrow \mathbb{Z}_q$ and compute $T_j := h^{u_j}$.
  (3) Compute the Fiat–Shamir challenge c as
  $$c := H(\mathsf{vk}, \mathsf{pid}, g, h, C, X, T_1, \ldots, T_N).$$
  (4) Compute $c_j := c - \sum_{i \in [N] \setminus \{j\}} c_i$.
  (5) Compute the response for the true relation as $z_j := u_j + c_j r$.
  (6) Output $\pi := \left(\{T_i, c_i, z_i\}_{i \in [N]}\right)$.
- $\mathsf{Verify}_{\mathsf{set}}$: The verifier runs the following:
  (1) Compute
  $$c := H(\mathsf{vk}, \mathsf{pid}, g, h, C, X, T_1, \ldots, T_N).$$
  (2) Check
  $$c \stackrel{?}{=} \sum_{i \in [N]} c_i.$$
  (3) For every $i \in [N]$, check
  $$h^{z_i} \stackrel{?}{=} T_i \left(C/g^{x_i}\right)^{c_i}.$$
  If any of the above checks fail, output 0; otherwise, output 1.

For the true branch j, correctness follows from

$$\begin{aligned} h^{z_j} &= h^{u_j + c_j r} \\ &= h^{u_j} (h^r)^{c_j} \\ &= T_j \left(C/g^{x_j}\right)^{c_j}, \end{aligned}$$

where

$$C/g^{x_j} = h^r.$$

For every $i \neq j$, the verification equation holds by the construction of T_i.

Statistical zero-knowledge. The underlying interactive CDS protocol is statistically zero-knowledge. In particular, the simulator does not need to know the witness index j. For every $i \in [N]$, it samples

$$c_i, z_i \leftarrow \mathbb{Z}_q$$

and computes

$$T_i := h^{z_i} \left(C/g^{x_i}\right)^{-c_i}.$$

It then sets

$$c := \sum_{i \in [N]} c_i$$

and programs the random oracle at

$$(\mathsf{vk}, \mathsf{pid}, g, h, C, X, T_1, \ldots, T_N)$$

to return c.

For the real branch j, the honest prover samples $u_j \leftarrow \mathbb{Z}_q$ and sets

$$T_j = h^{u_j}, \qquad z_j = u_j + c_j r.$$

Equivalently,

$$T_j = h^{z_j} \left(C/g^{x_j}\right)^{-c_j}.$$

Since u_j is uniformly distributed, T_j is uniformly distributed in the group. Similarly, in the simulated transcript, z_j is uniformly distributed and hence

$$T_j = h^{z_j} \left(C/g^{x_j}\right)^{-c_j}$$

has the same distribution. Thus the simulated and real transcripts are identically distributed conditioned on the programmed random oracle response.

Importantly, this statistical zero-knowledge property does not rely on the hardness of the discrete logarithm problem. Even an unbounded verifier that can compute discrete logarithms can obtain a candidate opening for every x_i from

$$C/g^{x_i} = h^{r_i}.$$

The CDS transcript nevertheless does not reveal which i corresponds to the witness used by the prover.

For the proof if the adversary makes at most q_H random-oracle queries, the probability that it queries the programmed point before the simulator programs it is at most $\frac{q_H}{|\mathbb{Z}_q|}$. Consequently, the statistical distance between the real and simulated distributions is at most $\frac{q_H}{|\mathbb{Z}_q|}$. Hence, the resulting NIZK is statistically zero-knowledge in the programmable random-oracle model.

The soundness uses the forking lemma and relies on the Discrete Log assumption. We refer to [21] for the complete proof details.