Pastel: Single Secret Leader Election with Private Weights

Jung Hee Cheon[1,2], Kevin Choi[1], Intak Hwang[1], Jaehee Kang[1], Alireza Kavousi[3], Junho Lee[1], and Yongdong Yeo[1]

[1] Seoul National University
[2] CryptoLab Inc.
[3] University of Cambridge

Abstract. A single secret leader election (SSLE) protocol allows an election of exactly one leader among n parties in a secrecy-preserving manner, where only the elected leader knows it has won until it publicly announces and claims leadership. Motivated by proof-of-stake blockchains where stakes are stored in an encrypted or committed form for the sake of privacy, we design and implement the first SSLE protocol with private weights. Each party's probability of winning an election in our protocol is weighted by stake, and these weights are private. Prior schemes in the literature only allow probabilistic (i.e. expectation of one but can be zero or many leaders) elections in the same setting.

Preserving privacy also means public verification is paramount in order for a protocol to be practically reliable. In this respect, our protocol achieves security against malicious adversaries with a concrete implementation. In particular, we concretely implement the needed zero-knowledge proofs and thus do not rely on slashing (financial penalties) to achieve end-to-end malicious security, though slashing can still serve as a performance optimization. Along the way, we observe that previous SSLE protocols in a setting without private weights (either unweighted or with public weights) are either maliciously secure with implementation or post-quantum secure but not both, while our protocol achieves both.

Email: jhcheon@snu.ac.kr, kc2853@snu.ac.kr, intak.hwang@snu.ac.kr, jaehee.kang422@snu.ac.kr, ak2829@cam.ac.uk, abab9579@snu.ac.kr, yongdong@snu.ac.kr

Table of Contents

1 Introduction 3
  1.1 Our Contributions 5
  1.2 Technical Overview 5
2 Preliminaries 7
  2.1 Notation 7
  2.2 Single Secret Leader Election 7
  2.3 RLWE/RGSW-based Homomorphic Encryption Scheme 8
  2.4 Threshold Fully Homomorphic Encryption 9
  2.5 Zero-Knowledge Proof 9
3 Private-Stake Single Secret Leader Election 9
4 Pastel 10
  4.1 Semi-Honest Pastel 11
  4.2 Maliciously-Secure Pastel 12
5 Security 14
6 Implementation 16
  6.1 Semi-Honest Pastel 16
  6.2 Malicious Security 16
7 Extension 16
  7.1 Independently Selected Key Committee 16
  7.2 Leader-Sampled Key Committees 17
  7.3 Accountable Pastel 19
  7.4 Updatable Pastel 19
  7.5 Weight Leakage from Pastel Functionality 20
A Supplementary Preliminary 22
  A.1 RLWE/RGSW-based Homomorphic Encryption 22
  A.2 Threshold Fully Homomorphic Encryption 23
  A.3 Non-Interactive Zero-Knowledge Proof 24
B Correctness Analysis of the Homomorphic Circuit 25
  B.1 Encryption and Registration 25
  B.2 Finalization 26
C Malicious Security of Pastel 28
D Supplementary Implementation Details 28
  D.1 Homomorphic Encryption Scheme Parameters of Pastel 28
  D.2 Malicious Security 29

1 Introduction

A property of leader election called *secrecy* allows the elected leader's identity to remain hidden until it reveals itself, which prevents adaptive attacks like the denial-of-service (DoS) attack targeting a future leader that is otherwise, say via public randomness [10, 23], known in advance in a public manner.

While the early secret leader election protocols such as Algorand [19] (leveraging verifiable random functions [13, 29]) lack the needed determinism and have a probabilistic expectation of one leader, Boneh et al. [4] first formally define *single secret leader election (SSLE)*, in which exactly one leader wins.

Subsequent works explore different efficiency–security tradeoffs. These include constructions with post-quantum security [5], adaptive security [8], UC security [9], and security against network-level equivocation attacks [14]. The need for better scalability for blockchain adoption has also motivated other variants of SSLE like *batch* leader election [7], where a single leader is elected for several elections at once. This comes at the cost of weakening the unpredictability guarantees by limiting it to a ring of potential candidates, while allowing the rate of leader election to match that of block production.

Weights. Despite the focus of most prior work, it is in fact usual in a proof-of-stake setting (by definition) that each party's probability of winning an election is weighted by stake. A weighted protocol therefore aims to elect party P_i with probability $\Pr[P_i \text{ is elected}] = \frac{w_i}{\sum_{j=1}^{n} w_j}$, where w_i is the stake or weight of P_i given n parties.

Private Weights and Challenges. Additionally, we consider a setting with *private* weights, motivated by proof-of-stake blockchains in which stakes are stored in an encrypted or committed form for the sake of privacy. These constructions exist not only in theoretical forms [17, 24] (realizing what they call *private proof-of-stake*), but also partially in practice, e.g. in the form of Zama [35]. In the former constructions, which target a more holistic transaction system based on stakes that are private, *anonymous verifiable random functions* power the leader election procedure and thus only allow probabilistic elections as opposed to SSLE. Deployed in 2026, the Zama network employs fully homomorphic encryption (FHE) to provide privacy of stakes at an application level but does not necessarily aim to realize private proof-of-stake as a whole. Our motivations are therefore two-fold: exploring if SSLE is practically possible under the framework of private proof-of-stake, while exploring if the application-level scope of privacy ensured by a Zama-like network leveraging FHE can be extended to realize SSLE. We emphasize that realizing an entire private proof-of-stake is a non-goal in this work.

The weighted setting raises an interesting privacy question. In a system with public weights, a weighted election can be implemented conceptually by *virtualization*: a party with weight w_i is represented by w_i independent virtual entries. Selecting one entry uniformly then elects party P_i with probability proportional to w_i. However, this representation is unsuitable when individual stakes are private. Publishing w_i entries, or otherwise exposing the multiplicity of entries associated with P_i, directly reveals its weight. Concealing this multiplicity requires additional cryptographic machinery and is not obtained automatically by applying an unweighted shuffle- or one-hot-selection-based SSLE constructions [4, 5, 27, 33].

Table 1: Comparison of Pastel with prior SSLE schemes. The columns Round, Weighted, and Post-Quantum are self-explanatory; we elaborate on the rest. Malicious security is discussed theoretically but not implemented in Qelect and Relect. A protocol is *reconfiguration-friendly* if parties may freely enter or exit the elections without incurring setup cost per election. Qelect assumes a trusted dealer that generates randomness for a fixed number of elections, which Relect does not do. As written, the two ThFHE-based approaches and Relect require setup cost per election, though they may also adopt the key committee model of Pastel to be reconfiguration-friendly.

	Round	Weighted	Malicious Security	Post-Quantum	Reconfig.-Friendly
ThFHE-based [4]	$O(1)$	✗	✗	✓	✗
ThFHE-based [16]	$O(1)$	✓	✗	✓	✗
RLWE-based [5]	$O(n)$	✗	✗	✓	✓
MPC-based [3]	$O(\log n)$	✓	✓	✗	✗
Qelect [33]	$O(1)$	✗	✓	✓	✗
Relect [27]	$O(1)$	✗	✓	✓	✗
Mobius [14]	$O(n)$	✗	✓	✗	✓
Pastel	$O(1)$	✓	✓	✓	✓

Private weighted election must therefore solve two related problems. First, a party must prove that its encrypted election record represents an authorized stake without revealing that stake. Second, the protocol must sample from the resulting hidden weighted distribution while revealing neither the selected party nor the individual weights.

Our goal is to construct an SSLE protocol that supports weighted election with private weights. More precisely, the protocol should select a unique leader with probability proportional to its private weight, reveal the election result only to the winner until it claims leadership, and permit public verification of the claim.

Related Work. While our protocol Pastel is the first to concretely instantiate SSLE with private weights, we compare it with prior SSLE schemes (which occur in settings without weights or with public weights) in Table 1. Boneh et al. [4] first formalize SSLE and give constructions based on indistinguishability obfuscation, threshold fully homomorphic encryption (ThFHE), and DDH-based shuffling. Their ThFHE construction achieves constant-round online election and can rely on post-quantum assumptions, but does not natively support weighted elections or active-malicious security. Subsequent work strengthened SSLE under post-quantum, adaptive, and universally composable security notions. In particular, Boneh et al. [5] proposed an RLWE-based construction with post-quantum security and reusable registration, at the cost of sequential re-randomization, while Qelect [33] obtains constant-round post-quantum election using RLWE-based multi-party randomizable commitments. Qelect handles malicious behavior through lightweight prevention and retroactive detection, yielding a relaxed malicious-security guarantee.

Weighted SSLE is particularly important in proof-of-stake systems. A folklore approach is stake-unit virtualization, in which a party registers once per unit of stake, but its cost scales with total stake. Backes et al. [3] avoid this expansion through MPC-based oblivious selection, supporting arbitrary stake distributions with $O(\log n)$ communication rounds and malicious security, but without post-quantum assumptions or non-expiring registration. Homomorphic Sortition [16] instead uses ThFHE to sample from encrypted stake intervals. It supports arbitrary weights, Byzantine participants, and non-expiring registration, allowing independent elections to be executed asynchronously after setup. Mobius [14] addresses a distinct concern (orthogonal to our work) pertaining to network-level equivocation attacks. It achieves Byzantine resilience in a particular adversary model of interest, but it is neither weighted nor post-quantum.

1.1 Our Contributions

Pastel combines dimensions that remain largely separated in the existing SSLE literature: ❶ constant-round online election, ❷ stake-proportional leader election, ❸ malicious security, and ❹ post-quantum assumptions. Unlike prior ThFHE-based approaches [4, 16], it uses packed homomorphic sampling to select a commitment while keeping individual weights encrypted. Pastel also supports a deployment in which threshold key custody is decoupled from the leader-candidate population, enabling a relatively small key committee to serve a larger population. Our contributions are as follows:

- **Private-stake SSLE.** We formalize private-stake single secret leader election (pSSLE), extending SSLE to parties with private committed weights. In pSSLE, each party P_i has private weight w_i and is elected with probability proportional to its weight, while the protocol preserves leader secrecy and public verifiability. Our definition captures the requirement that an encrypted election record be bound to an authorized private weight.
- **Our protocol Pastel.** We construct Pastel, a constant-round pSSLE protocol based on threshold RLWE/RGSW encryption. Pastel encodes weighted commitments as polynomial coefficients and uses an encrypted random monomial, polynomial multiplication, and a trace operation to select one commitment privately. We augment the construction with concrete zero-knowledge proofs for robustness, obtaining concrete malicious security without resorting to slashing (financial penalties).
- **Implementation and Performance Evaluation.** We implement Pastel in Go and evaluate it on a single-threaded CPU. In a semi-honest setting with 128 participants and a 32-member key committee, each participant joins Pastel in 15 ms, and the entire election process completes in 1.5s. In a malicious setting, proof generation takes 2.5s per participant and 0.6s per key-committee member, while verifying all proofs takes 15.1s in total.

1.2 Technical Overview

System Model. We consider a synchronous network with a set of n parties $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, also called the leader candidate population. We assume a malicious

adversary that may engage in any arbitrary misbehavior, runs in probabilistic polynomial time (PPT), and is static where it chooses parties to be corrupted before a protocol run. We assume broadcast rounds of communication.

Additionally, we model a key committee $\mathcal{K} = \{Q_i\}_{i=1}^{m}$ separately from $\mathcal{P}$, which allows a more dynamic protocol assuming that $\mathcal{P}$ may reconfigure more frequently (as parties enter or exit) while $\mathcal{K}$ remains relatively static. The key committee is responsible for generating the secret key of threshold FHE in a distributed fashion (during setup). For instance, this is the model adopted by the Zama network [35], which is already deployed in practice. It is also possible to set $\mathcal{K} = \mathcal{P}$ to subsume the conventional model for threshold cryptography. Also during setup, we assume for our construction that each weight is encrypted, and so these ciphertexts (encrypted monomials via threshold FHE) are public while the weights are private.

Overview of Pastel**.** In this work, we aim for a protocol that selects exactly one leader among participants P_i, each holding a weight w_i, such that $\Pr[P_i \text{ is selected}] = w_i/(\sum_j w_j)$, where individual weights are kept private and the elected leader's identity is hidden until the leader voluntarily reveals one's identity.

We reduce pSSLE to the problem of uniformly selecting an entry from an encrypted weighted commitment list. Each party P_i generates an identity-hiding commitment and retains a witness with which it can later prove ownership if selected. By constructing a list of length $W = \sum_j w_j$ in which P_i's commitment appears w_i times, and uniformly selecting an entry from the list, one can select P_i with probability w_i/W. The main design problem is therefore to construct this weighted list without revealing the multiplicities and to select one of its entries uniformly through homomorphic computation.

The construction of such a list hinges on how commitments are represented. When each party's commitment is itself a ciphertext, such as an encryption of zero [27, 33], it is not straightforward to pack multiple such commitments into a single ciphertext while keeping them distinct and individually addressable. In contrast, when commitments are represented as hash values [4, 16], each hash value can be placed in a distinct vector slot or polynomial coefficient. Provided that W fits within the available plaintext capacity, this representation allows the entire weighted list to be stored in a single ciphertext, reducing ciphertext-level computation and storage by processing all entries together. Moreover, since these hash-based commitments remain encrypted throughout the election, no separate rerandomization procedure [27, 33] is needed, saving the associated communication rounds and per-party computation.

We encode the weighted list in polynomial coefficients, so that inserting w_i copies of P_i's commitment requires only a monomial shift and an addition. Starting from $C_0(X) = 0$, we recursively construct

$$C_i(X) = X^{w_i} C_{i-1}(X) + h_i(1 + X + \cdots + X^{w_i - 1}),$$

where h_i denotes P_i's hash-based commitment. The first term shifts the entries accumulated so far by w_i coefficient positions, while the second term fills the newly created positions with w_i copies of h_i. Consequently, after processing P_i, the coefficients of $C_i(X)$ contain exactly w_j copies of h_j for every $j \le i$. After all parties have participated, the entire weighted commitment list is represented by a single polynomial.

Once the list is coefficient-encoded, uniform selection requires only a random monomial shift followed by extraction of the constant coefficient. Let $C(X) = \sum_{k=0}^{W-1} c_k X^k$ denote the resulting encoding polynomial. The parties homomorphically combine their randomness to obtain a uniformly random position $r = \sum_i r_i \bmod W$ and the corresponding monomial X^{-r}. Multiplying $C(X)$ by this monomial shifts the selected coefficient c_r to the constant position. Thus, the selected commitment can be obtained using one monomial multiplication and one coefficient extraction.

From this polynomial-encoding perspective, earlier schemes do not use the polynomial structure to evaluate the selection directly. They explicitly generate the one-hot selector through heavy operations over packed vector slots [16, 33] before applying it to the commitment list. Relect [27] represents the random position as a monomial, but because its commitment list is not encoded as a single compatible polynomial, it must extract the monomial's coefficients individually and apply them entry-wise.

2 Preliminaries

2.1 Notation

We use λ for our computational model's security parameter and $\mathrm{negl}(\lambda)$ to denote a negligible function. For a non-empty set S, we write $x \xleftarrow{\$} S$ to denote sampling uniformly at random from S. For a distribution χ, we write $x \leftarrow \chi$ to denote sampling according to χ. We use $[n]$ to represent the set $\{1, \ldots, n\}$.

In the context of (threshold) FHE, we may simplify the notation $\mathsf{ThFHE.Enc}(\mathsf{pk}, \cdot)$ described in Section 2.4 and instead represent a ciphertext by $\mathsf{Enc}(\cdot)$. In that case, both $\mathsf{ThFHE.Enc}(\mathsf{pk}, \cdot)$ and $\mathsf{Enc}(\cdot)$ pertain to RLWE encryption while $\mathsf{Enc}_{\mathsf{RGSW}}(\cdot)$ denotes RGSW encryption; both encryption schemes are described in Section 2.3.

2.2 Single Secret Leader Election

We account for prior work [4, 14, 33] but model SSLE slightly differently by abstracting away any distributed key generation (DKG) [2, 18, 31, 32] needed in the setup stage. In particular, we first assume an ideal algorithm for key generation and key use at a theoretical level, and then a separate subprotocol is instantiated to "distribute" that algorithm in practice.

Definition 1 (Single Secret Leader Election [4, 14, 33]). *For a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, a* single secret leader election (SSLE) *scheme over $\mathcal{P}$ is represented by the following tuple of algorithms:*

$\mathsf{SSLE.Setup}(\lambda) \to (\mathsf{pp}, \mathsf{sk}, \mathsf{pk})$ *takes the security parameter λ, and outputs public parameters* pp*, a secret key* sk*, and a public key* pk*. The public parameters* pp *are implicitly given as input to all other algorithms. This setup is run once to initiate a sequence of elections.*

$\mathsf{SSLE.Register}(x_i, \mathsf{st}') \to \mathsf{st}$ *takes a witness x_i by P_i and state* st'*, and outputs the updated state* st*.*

$\mathsf{SSLE.Finalize}(\mathsf{sk}, \mathsf{st}') \to \mathsf{st}$ *takes the secret key* sk *and state* st'*, and outputs the updated state* st*.*

$\mathsf{SSLE.Elect}(x_i, \mathsf{st}) \to \{(0, \perp), (1, \pi_i)\}$ *takes a witness* x_i *by* P_i *and state* st*, and locally outputs* 1 *with a proof of leadership* π_i *if elected and* $(0, \perp)$ *otherwise.*

$\mathsf{SSLE.Verify}(\pi, \mathsf{st}) \to \{0, 1\}$ *takes a claimed proof of leadership* π *by* $P_i \in \mathcal{P}$ *and state* st*, and outputs* 1 *if* π *passes verification and* 0 *otherwise.*

An SSLE scheme should satisfy a number of security properties (formally given by Boneh et al. [4]) including: *Uniqueness*, requiring that only one leader is elected for each protocol run; *Fairness*, requiring that each party has an equal probability $1/n$ of winning; and *Unpredictability*, requiring that the adversary does not learn in advance which (honest) party is being elected.

Distributed Setup. Although the above syntax considers an idealized key custody abstraction (i.e. a trusted algorithm), it is possible (and realistically necessary) to distribute the key generation process. For instance, a DKG [2, 31] can be run among a separate key committee $\mathcal{K}$ of size m, in which case m and t (threshold parameter) are additional inputs to $\mathsf{SSLE.Setup}$. We observe that the participant set $\mathcal{P}$ need not equal $\mathcal{K}$, though $\mathcal{P} = \mathcal{K}$ in prior approaches [4, 16, 33]. The benefit of this explicit separation is that $\mathcal{K}$ may remain static for many elections while $\mathcal{P}$ may change dynamically.

Consequently, we model a ThFHE-based approach that is *reconfiguration-friendly*, i.e. parties may freely enter or exit subsequent elections without incurring the cost of $\mathsf{SSLE.Setup}$ per election. An additional remark is that setting $\mathsf{sk} = \perp$ in the above syntax accounts for prior shuffle-based approaches [4, 5, 14]. The algorithm $\mathsf{SSLE.Finalize}$ is unused in that case. Note if $\mathsf{SSLE.Register}$ has to be sequentially run (as in shuffle-based SSLE), the round complexity grows to $O(n)$.

2.3 RLWE/RGSW-based Homomorphic Encryption Scheme

The following *homomorphic encryption schemes* are used in our paper. We defer the full definition to Appendix A.1.

Definition 2 (RLWE/RGSW Homomorphic Encryption [28, 30]). *An* RLWE encryption *scheme is represented by:* $\mathsf{RLWE.KeyGen}(1^\lambda)$ *for generating public parameters and a key pair,* $\mathsf{RLWE.Enc}(\mathsf{pk}, \mu)$ *for encrypting a message, and* $\mathsf{RLWE.Dec}(\mathsf{sk}, \mathtt{ct})$ *for decrypting a ciphertext.*

The RGSW encryption *scheme we use is represented by: the same key generation algorithm as above, and* $\mathsf{RGSW.Enc}(\mathsf{pk}, \mu)$ *for encrypting a message.*

The external product *between the two schemes is denoted by:*

$$\mathtt{ct}_{\mathsf{RLWE}} \boxtimes \mathtt{ct}_{\mathsf{RGSW}} = \mathtt{ct}_{\mathsf{RLWE}}$$

For a message space ring $\mathcal{R}$*, the* trace map *onto a subring* $\mathcal{R}' \subset \mathcal{R}$ *is represented by:* $\mathsf{Tr}_{\mathcal{R}'}(m) = \sum_{\sigma_i \in \mathsf{Gal}(\mathcal{R}/\mathcal{R}')} \sigma_i(m)$*;* $\mathsf{Gal}(\mathcal{R}/\mathcal{R}')$ *denotes the corresponding Galois group.*

2.4 Threshold Fully Homomorphic Encryption

We use *threshold fully homomorphic encryption* to distribute the decryption process. We defer the full definition to Appendix A.2. Throughout, the concrete underlying schemes are RLWE and RGSW.

Definition 3 (Threshold fully homomorphic encryption [6]). *A* threshold fully homomorphic encryption *scheme is represented by:* $\mathsf{ThFHE.Setup}(\lambda, n, t)$ *for key distribution,* $\mathsf{ThFHE.Enc}(\mathsf{pk}, \mu)$ *for encrypting,* $\mathsf{ThFHE.Eval}(\mathsf{pk}, \mathcal{F}, \mathtt{ct}_1, \cdots, \mathtt{ct}_k)$ *for evaluating a circuit homomorphically (on ciphertexts),* $\mathsf{ThFHE.PartDec}(\mathsf{pk}, \mathtt{ct}, \mathsf{sk}_i)$ *for outputting a partial decryption (by* P_i*), and* $\mathsf{ThFHE.FinDec}(\mathsf{pk}, \{(i, \mathsf{pd}_i)\}_{i\in\gamma})$ *for combining a threshold number of partial decryptions.*

We assume our ThFHE satisfies semantic security [6, Definition 5.4] and simulation security [6, Definition 5.5]. We refer to Appendix A.2 for more details.

2.5 Zero-Knowledge Proof

We review the concept of non-interactive zero-knowledge proof (NIZK) in the ROM, which is commonly obtained by applying the Fiat-Shamir transform [15]. See Appendix A.3 for the full definition.

Definition 4 (NIZK). *A non-interactive zero-knowledge proof (NIZK) in ROM is represented by:* $\mathsf{NIZK.Setup}(\lambda)$ *for generating a common reference string (crs),* $\mathsf{NIZK.Prove}^{\mathsf{RO}}(\mathsf{x}, \mathsf{w})$ *for generating a proof, and* $\mathsf{NIZK.Verify}^{\mathsf{RO}}(\mathsf{x}, \pi)$ *for verifying a proof.*

It should satisfy three security properties: completeness, knowledge soundness, and zero-knowledge, as detailed in Appendix A.3.

3 Private-Stake Single Secret Leader Election

In this section, we formally define *private-stake single secret leader election (pSSLE)* and its security properties. This can be understood as an extension of SSLE, generalized to a weighted version with respect to private stake.

Definition 5 (Private-Stake Single Secret Leader Election). *A* private-stake single secret leader election (pSSLE) *scheme is SSLE with weighted probability of election being proportional to weights that are private. Let* $\mathcal{X}$ *be a space of witnesses, and let* $\mathcal{H}$ *be a space of commitments for a commitment scheme* Π_{Com}*. The outcome of pSSLE is the same: exactly one leader* P_i *from a set of n parties* $\mathcal{P} = \{P_1, \ldots, P_n\}$ *is elected with the probability* $\frac{w_i}{\sum_{j\in[n]} w_j}$*. With each weight denoted by* $w_i \in \mathbb{Z}_{>0}$ *corresponding to* P_i*, pSSLE scheme over* Π_{Com} *inherits the SSLE algorithms in Definition 1, except:*

$\mathsf{pSSLE.Setup}(\lambda, \{w_i\}_{i\in[n]}) \to (\mathsf{pp}, \mathsf{sk}, \mathsf{pk})$ *on input of the security parameter* λ *and a set of weights* $\{w_i\}_{i\in[n]}$*, outputs public parameters* pp *(including commitments to each weight), a secret key* sk*, and a public key* pk*.*

$\mathsf{pSSLE.Register}(\mathsf{cm}_{w_i}, x_i, \mathsf{st}') \to \mathsf{st}$ *on input of a commitment* cm_{w_i} *to weight* w_i*, a witness* $x_i \in \mathcal{X}$ *of* P_i*, and state* st'*, outputs the updated state* st.

Security Properties. The security properties of an unweighted SSLE [4] are adapted to a private-stake setting as follows. Let $\mathcal{A}$ be a PPT adversary controlling a corruption index set $\mathsf{corr} \subseteq [n]$, where $|\mathsf{corr}| \leq n-1$, and let $\mathsf{hon} = [n] \setminus \mathsf{corr}$ denote the honest indices.

The election proceeds as specified. After a one-time $\mathsf{pSSLE.Setup}$, all parties run $\mathsf{pSSLE.Register}$, with the caveat that each P_i for $i \in \mathsf{corr}$ may do so with a maliciously generated input. After the state is finalized via $\mathsf{pSSLE.Finalize}$, all parties run $\mathsf{pSSLE.Elect}$ to receive either $(0, \perp)$ or $(1, \pi_i)$, the process of which the corrupted parties may also attempt to take advantage of.

For an election identifier eid, let

$$\mathcal{L}_{\mathsf{eid}} = \{i \in [n] : \exists \pi_i \text{ such that } \mathsf{pSSLE.Verify}(\pi_i, \mathsf{st}) = 1\}$$

denote the set of indices that can produce an accepting leadership claim. Given this adversary, a secure pSSLE satisfies:

1. **Uniqueness,** if $\Pr[|\mathcal{L}_{\mathsf{eid}}| > 1]$ is negligible.
2. **Fairness,** if for every $i \in [n]$ with $w_i > 0$,
$$\left| \Pr[i \in \mathcal{L}_{\mathsf{eid}}] - \frac{w_i}{\sum_{k \in [n]} w_k} \right| \leq \mathrm{negl}(\lambda).$$
3. **Unpredictability.** Let Good be the event that $\mathcal{L}_{\mathsf{eid}} = \{i^\star\}$ for some $i^\star \in \mathsf{hon}$, and $W_{\mathsf{hon}} = \sum_{i \in \mathsf{hon}} w_i$. Let st be the adversary's pre-claim view, including the public election output and adversarial state. Conditioned on Good, sample J independently of st according to
$$\Pr[J = i] = \frac{w_i}{W_{\mathsf{hon}}} \qquad (i \in \mathsf{hon}).$$
For every PPT distinguisher $\mathcal{D}$,
$$\left|\Pr[\mathcal{D}(\mathsf{st}, \{w_i\}_{i \in [n]}, i^\star) = 1 \mid \mathsf{Good}] - \Pr[\mathcal{D}(\mathsf{st}, \{w_i\}_{i \in [n]}, J) = 1 \mid \mathsf{Good}]\right| \leq \mathrm{negl}(\lambda).$$

Remark. This definition is conservative with respect to stake privacy: the adversary is artificially given the entire weight vector, including honest parties' weights, although the protocol does not publish them. Conditioned on an honest winner, it requires the adversary's pre-claim view to reveal no information about the winner's identity beyond the known weighted distribution. Equivalently, the actual honest winner must be indistinguishable from an independent draw from that distribution, given the view and weights. In particular, an adversary's overall guessing probability is at most $\max_{i \in \mathsf{hon}} w_i / W_{\mathsf{hon}} + \mathrm{negl}(\lambda)$.

4 Pastel

Throughout this section, we propose a pSSLE scheme, Pastel. We first present a semi-honest version of our Pastel scheme in Section 4.1 as a stepping stone. Then, we develop it to achieve malicious security in Section 4.2.

4.1 Semi-Honest Pastel

At a high level, our protocol generates an encoding ciphertext $\mathtt{ct}_{\mathrm{ecd}}$ encrypting each participating party's commitment, each duplicated according to its integer weight[3]. Then we perform blind-rotation-like multiplication by a collaboratively generated randomness ciphertext $\mathtt{ct}_R$, and extract only the selected commitment via a trace operation. We assume that the total weight W is a public power-of-2 positive integer. For a HE plaintext space $\mathbb{Z}[X]/(X^N + 1)$, we set $S := N/W$ and $Y := X^S$. The detailed procedure of semi-honest Pastel follows the uncolored descriptions of Figure 1.

Encoding of Commitment. Encoding a 128-bit hash commitment in a single coefficient requires a correspondingly larger ciphertext modulus, increasing both ciphertext size and homomorphic operation costs. This inefficiency becomes even more obvious when accounting for the additional modulus headroom required for noise-flooding during ThFHE decryption. To mitigate this inefficiency, for N satisfying $N/W \geq 2$, we split each hash commitment h_i into digits in base-B, and store the digits in multiple coefficients.

A base-B decomposition of commitment h_i is a vector $(h_i^{(1)}, \ldots, h_i^{(C)})$ satisfying $h_i = \sum_{j\in[C]} h_i^{(j)} B^{j-1}$. We take smaller stride $S := N/(W \cdot C)$, and encode h_i as $\sum_{j\in[C]} h_i^{(j)} Y^{W\cdot(j-1)}$. Under the basis $\{1, Y, \cdots, Y^{W\cdot C-1}\}$, observe that the corresponding $\mathtt{ct}_{\mathrm{ecd}}$ encrypts:

$$\overbrace{\underbrace{h_1^{(1)}, \ldots, h_1^{(1)}}_{w_1 \text{ times}} \mid \cdots \mid \underbrace{h_n^{(1)}, \ldots, h_n^{(1)}}_{w_n \text{ times}}}^{W \text{ entries}} \mid \cdots \mid \overbrace{\underbrace{h_1^{(C)}, \ldots, h_1^{(C)}}_{w_1 \text{ times}} \mid \cdots \mid \underbrace{h_n^{(C)}, \ldots, h_n^{(C)}}_{w_n \text{ times}}}^{W \text{ entries}} \tag{1}$$

For the encoded ciphertext $\mathtt{ct}_{\mathrm{ecd}}$, multiplication by $\mathtt{ct}_R$ encrypting Y^{-r} shifts the coefficients of $\{Y^k \mid k = r \pmod W\}$ to the coefficients of $\{1, Y^W, \ldots, Y^{W\cdot(C-1)}\}$ up to sign, which corresponds to an elected party's commitment.

Decoding of Decrypted Polynomial. Decrypting the output ciphertext $\mathtt{ct}_{\mathrm{out}}$ yields a rotation of the original vector $(h_i^{(1)}, \ldots, h_i^{(C)})$; the amount of rotation can be detected through signs, since each wrap-around flips the sign.[4] By rotating back, we can recover the original vector and the commitment h_i.

Remark 1 (Computational Efficiency). Unlike many HE-based protocols, Pastel does not require a large ciphertext modulus, which directly affects the computational efficiency. Ours relies on the external product, which supports multiplications without exhausting the modulus. Although our scheme supports pSSLE, it remains independent of the magnitude of the weights, and requires only $O(n)$ homomorphic multiplications, as in the recent non-weighted SSLE scheme [27].

[3] Given arbitrary real-valued weights specified to finite precision, we can always scale them to integers while preserving their relative ratios. We refer the reader to Remark 2 for a discussion of the precision supported by our scheme.

[4] Since zero has no sign, we add 1 to each component before encoding to prevent ambiguity.

Remark 2 (Scalability of Total Weight). The condition $W \leq N$ is a structural requirement of our construction, which encodes data coefficient-wise in a single ciphertext. When each commitment is represented by C coefficients, the capacity constraint becomes $CW \leq N$. Consequently, supporting a larger total weight (or finer weight resolution) requires a larger ring dimension N. One possible extension would be to keep the ring dimension fixed and distribute the weight-expanded list across multiple ciphertexts. However, adapting our blind-rotation-based protocol to such a setting is nontrivial, and we leave this extension for future work.

Remark 3 (Non-expiring Registration and Duplication Attack Prevention). Our Pastel can be easily extended to achieve *non-expiring registration* following prior techniques [16], enabling the elected leader to participate in future runs of the election without needing to re-register.[5] Moreover, following the duplicate-registration safeguard of Boneh et al. [4], one can derive a private component from each witness to form the encrypted tag and publish a separate component at registration; thus registrations with duplicate public components are rejected. For an *updatable variant* of Pastel that avoids reconstructing $\mathtt{ct}_{\mathrm{ecd}}$ from scratch for each election when only the weights of a subset of parties change, we refer the reader to Appendix 7.4.

4.2 Maliciously-Secure Pastel

In the malicious security model, the adversary can deviate in two ways: by sending malformed encryptions in the registration phase, and running incorrect algorithm during the finalization phase. To mitigate these attacks, we employ non-interactive zero-knowledge (NIZK) proofs.

Registration Phase. During registration, each party outputs two ciphertexts $\mathtt{ct}_H, \mathtt{ct}_R$ to other parties. Therefore, each party should attach a NIZK of well-formedness of two ciphertexts. We define the relations as follows.

$$\mathsf{R}_{\mathsf{Enc}} = \left\{ (\mathsf{x}_{\mathsf{Enc}}, \mathsf{w}_{\mathsf{Enc}}) \;\middle|\; \begin{array}{l} \mathsf{x}_{\mathsf{Enc}} = (\mathsf{pk}, \mathsf{ct}_H, \mathsf{ct}_R), \\ \mathsf{w}_{\mathsf{Enc}} = (r, h, \mathbf{v}_H, \mathbf{e}_H, \mathbf{v}_R, \mathbf{E}_R) \\ \mathtt{ct}_H = v_H \cdot \mathsf{pk}^\top + (h, 0) + \mathbf{e}_H \pmod{q} \\ \mathtt{ct}_R = \mathbf{v}_R \cdot \mathsf{pk}^\top + Y^{-r}(p\mathbf{g} \otimes \mathbf{I}_2) + \mathbf{E}_R \pmod{pq} \\ \|v_H\|_\infty, \|\mathbf{v}_R\|_\infty \leq \mathcal{B}_{\mathsf{Enc}}, \|\mathbf{e}_H\|_\infty, \|\mathbf{E}_R\|_\infty \leq \mathcal{B}_{\mathsf{err}} \\ \|h\|_\infty \leq 2^{128/C}, \deg h \leq C \end{array} \right\}$$

Finalization Phase. During the finalization phase, all parties compute encryption of election result, then engage in the distributed decryption protocol from the key committee. If any malicious party deviates from the computation, then it obtains a different ciphertext from the honest party, which would fail to decrypt. Therefore, we only need to guarantee malicious security during the distributed decryption phase, which we define the relation:

[5] That is, instead of revealing the tag in plain (which becomes un-reusable), we can choose a pre-preimage (ticket) and homomorphically compute PRF on it to derive the tag.

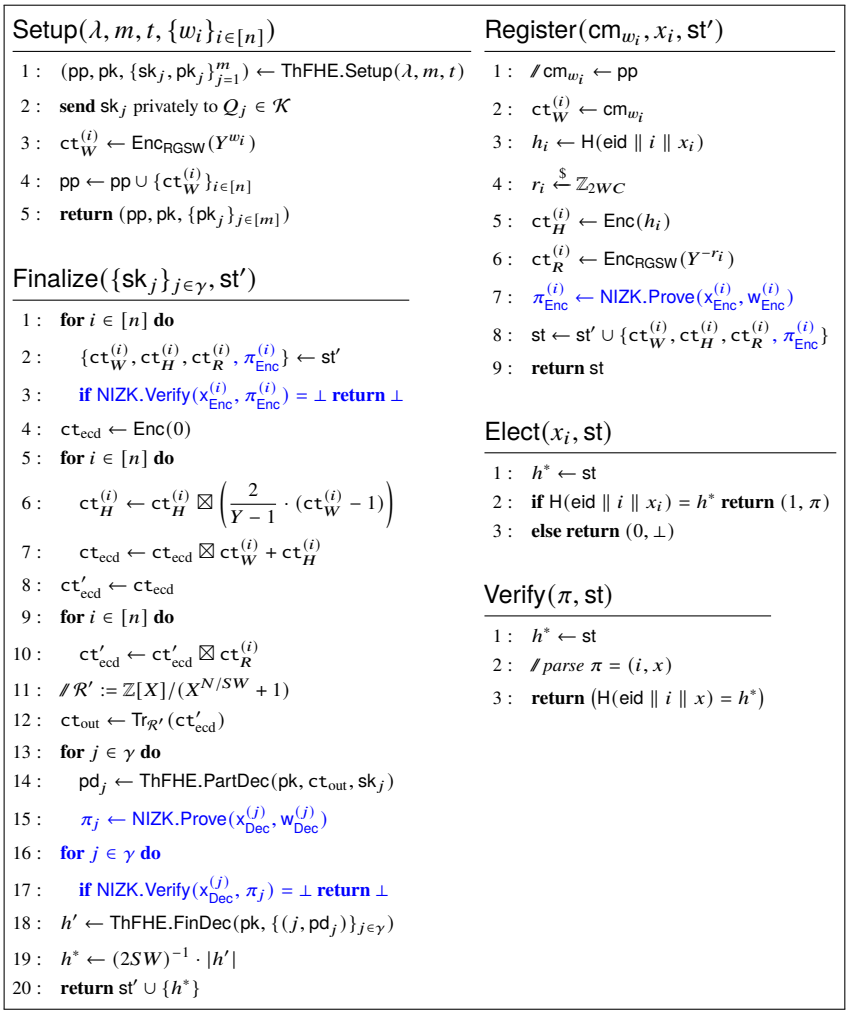

Fig. 1: The Pastel Protocol (modulo duplication attack prevention, explained in Remark 3). Above follows our private-stake SSLE syntax (Section 3) and achieves security against semi-honest (malicious) adversaries. See Section 4 for details.

$$\mathsf{R}_{\mathsf{Dec}} = \left\{ (\mathsf{x}_{\mathsf{Dec}}, \mathsf{w}_{\mathsf{Dec}}) \;\middle|\; \begin{array}{l} \mathsf{x}_{\mathsf{Dec}} = (\mathsf{pk}_i = (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}), \mathtt{ct} = (b, a), p_i), \\ \mathsf{w}_{\mathsf{Dec}} = (\mathsf{sk}_i, e_{\mathsf{pk},i}, e_i) \\ p_i = a \cdot \mathsf{sk}_i + e_i \pmod q \\ \mathsf{pk}_{i,0} = -\mathsf{pk}_{i,1} \cdot \mathsf{sk}_i + e_{\mathsf{pk},i} \pmod q \\ \|\mathsf{sk}_i\|_\infty \le B_{\mathsf{sk}}, \|e_{\mathsf{pk},i}\|_\infty \le B_{\mathsf{err}}, \|e_i\|_\infty \le B_{\mathsf{flood}} \end{array} \right\}$$

Note that we assume that partial public keys pk_i's are produced during the setup phase of ThFHE, from which the public key can be computed as $\mathsf{pk} = \sum \mathsf{pk}_i$.

Slashing. We highlight that the proof of encryption can be removed using *slashing* [27, 33]. This may be helpful when the number of participants is large, as the verification cost of proof of encryption scales with the number of parties, unlike proof of decryption which only scales with the number of key-committee members.

5 Security

Theorem 1 (Security of Semi-Honest Pastel). *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the leader-candidate population, and let $\mathcal{K} = \{Q_1, \ldots, Q_m\}$ be a t-of-m key committee. Assume that the total weight $W = \sum_{i \in [n]} w_i$ is a power of two dividing the ring dimension N. Assuming the RLWE assumption, correctness and security of the underlying t-of-m threshold FHE scheme, and modeling H as a random oracle, Pastel is a private-stake single secret leader election scheme secure against any semi-honest adversary that corrupts at most $n - 1$ leader candidates and fewer than t key-committee members.*

Proof. We prove uniqueness, fairness, and unpredictability.

Uniqueness. Correctness of the homomorphic circuit ensures a unique leader, except with negligible probability. Refer to Appendix B for FHE error analysis. Furthermore, registration tags bind the election identifier, party index, and witness. Except with negligible probability of a random oracle collision (or duplication attacks [4] which can be prevented as per Remark 3), no other registered witness produces that tag.

Fairness. Suppose each party samples $r_i \xleftarrow{\$} \mathbb{Z}_{2CW}$ independently, and let $R = \sum_{i \in [n]} r_i \pmod{2CW}$. Since at least one party is honest, R is uniform in $\mathbb{Z}_{2CW}$. We write $R = WR' + R''$, then $R' \in \mathbb{Z}_{2C}$ and $R'' \in \mathbb{Z}_W$ are uniform and independent of each other.

As in (1), $\mathtt{ct}_{\mathrm{ecd}}$ encrypts

$$\sum_{k \in [W]} h_k(Y) \cdot Y^{k-1}, \qquad h_k(Y) := \sum_{j \in [C]} h_{k,j} Y^{(j-1)W},$$

where number of k satisfying $h_i^{(j)} = h_{k,j}$ for all $j \in [C]$ is exactly w_i. Multiplying $Y^{-R} = Y^{-WR'} \cdot Y^{-R''}$ and taking trace, we extract coefficients of monomials $Y^{(j-1)W}$ over $j \in [C]$. Hence, we obtain

$$\sum_{j \in [C]} h_{R''+1,j} Y^{(j-1)W} \cdot Y^{-WR'},$$

and the position $R'' + 1$ is selected. In particular, $h_i^{(j)} = h_{R''+1,j}$ for all $j \in [C]$ with probability $\frac{w_i}{\sum_{i\in[n]} w_i}$, and decoding recovers the tag h_i.

Meanwhile, decryption failure contributes at most negligible error, so

$$\left|\Pr[i \in \mathcal{L}_{\mathsf{eid}}] - \frac{w_i}{\sum_{k\in[n]} w_k}\right| \leq \mathrm{negl}(\lambda).$$

Unpredictability. Let Good denote the event that $\mathcal{L}_{\mathsf{eid}} = \{i^\star\}$ for some $i^\star \in \mathsf{hon}$, and set $W_{\mathsf{hon}} = \sum_{k\in\mathsf{hon}} w_k$. Since the winner is determined only by R'', conditioning on any candidate's victory leaves R' uniform and independent of R''.

Fix honest candidates i and j of positive weight. We compare the pre-claim views conditioned on their respective victories. Starting with the view conditioned on i winning, we first replace the honest committee members' partial decryptions with simulated shares for the evaluated ciphertext and its plaintext. Threshold-decryption simulation security justifies this step. We then replace, one at a time, the other honest candidates' ciphertext $\mathtt{ct}_H^{(k)}$ with encryptions of $h_i(Y)$. The selected position is unaffected, so the decrypted result does not change. These replacements are indistinguishable by semantic security of the underlying RLWE encryption. Then, we sample $\widetilde{R}''$ uniformly from the weighted positions assigned to j and resample the honest candidates' offsets, conditioned on their aggregate being $\widetilde{R} = WR' + \widetilde{R}''$, while keeping R' unchanged. We replace the corresponding ciphertexts $\mathtt{ct}_R^{(k)}$ accordingly. Since every honest weighted position now contains $h_i(Y)$, the decrypted plaintext remains the same. Semantic security of the underlying RGSW encryption, together with simulation security for partial decryption, makes this transition indistinguishable. Except with negligible probability that the adversary queries either honest secret witness, h_i and h_j are independent and identically distributed random-oracle outputs. We can therefore relabel the common hidden commitment from h_i to h_j. Keeping the modified ciphertexts $\mathtt{ct}_R^{(k)}$ fixed, we reverse the commitment-replacement hybrids with j as the selected candidate and restore the honest committee members' real partial decryptions. This yields the pre-claim view conditioned on j winning. Hence, the views conditioned on any two honest winners are computationally indistinguishable.

By fairness and uniqueness, for every $k \in \mathsf{hon}$,

$$\Pr[i^\star = k \mid \mathsf{Good}] = \frac{w_k}{W_{\mathsf{hon}}} \pm \mathrm{negl}(\lambda).$$

Let J be sampled independently of the adversary's pre-claim view st, with $\Pr[J = k] = w_k/W_{\mathsf{hon}}$ for $k \in \mathsf{hon}$. There are polynomially many candidates, so the indistinguishability of the conditioned views implies that, for every PPT distinguisher $\mathcal{D}$,

$$\left|\Pr[\mathcal{D}(\mathsf{st}, \{w_k\}_{k\in[n]}, i^\star) = 1 \mid \mathsf{Good}] - \Pr[\mathcal{D}(\mathsf{st}, \{w_k\}_{k\in[n]}, J) = 1 \mid \mathsf{Good}]\right| \leq \mathrm{negl}(\lambda).$$

■

Theorem 2 (Malicious Security of Pastel). *Assuming the same from Theorem 1 and additionally a secure NIZK,* Pastel *is a private-stake single secret leader election scheme secure against any malicious adversary that corrupts at most $n - 1$ leader candidates and fewer than t key-committee members.*

Proof. We refer to Appendix C for the proof. ■

6 Implementation

6.1 Semi-Honest Pastel

Methodology. Our implementation is based on Lattigo [26], with m-of-m threshold on the key committee with $m = 32$. The number of participants n ranges from 16 to 2048. Following Ethereum [34], each participant is assigned a random weight of 1–50, with total weight $W = \min(32n, N/C)$. Parameter sets we use are presented in Table 4 of Appendix D. The ThFHE parameters achieve 128-bit computational security according to lattice estimator [1]. All experiments were conducted on an Intel Xeon Gold 6542Y CPU at 2.90 GHz using a single thread, and runtime is averaged over 10 runs.

For threshold decryption, we apply uniform-sampled noise flooding with respect to the estimated evaluation error to achieve 64-bit statistical security.

Results. We report the running time of the semi-honest Pastel in Table 2.

Table 2: Running time and per-party communication of the semi-honest protocol. Runtime is reported as mean ± standard deviation over 10 runs. Communication costs are per leader candidate for Register and per key-committee member for Finalize. For more detailed experimental settings and results, please see Table 5 in Appendix **??**.

	Phase	n	A	B	C
Time	Register (ms)	–	44.7 ± 1.2	43.0 ± 1.1	15.0 ± 0.4
	Finalize (s)	32	1.285 ± 0.014	1.096 ± 0.002	0.392 ± 0.002
		128	4.836 ± 0.028	4.263 ± 0.090	1.502 ± 0.004
		512	19.37 ± 0.17	16.78 ± 0.21	6.03 ± 0.14
		2048	82.39 ± 0.12	72.75 ± 0.09	24.15 ± 0.39
Comm.	Register (MiB)	–	10.5	11.0	3.5
	Finalize (MiB)	–	0.75	0.50	0.25

6.2 Malicious Security

To achieve malicious security, we use the Buckler proof system [21] and its extensions [22], due to its native support for proving lattice relations over FHE-sized parameters. Due to the nature of our relations, we make several optimizations; we refer readers to Appendix D for more details. The benchmark results are presented in Table 3.

7 Extension

7.1 Independently Selected Key Committee

Our default deployment model separates leader eligibility from key custody. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote the total leader candidate population, and let $\mathcal{K} = \{Q_1, \ldots, Q_m\}$

Table 3: Running time of the NIZK proof for each parameter set. Results are reported per-party. Note that Proof of Encryption can be removed if we use slashing instead.

	Proof of Encryption			Proof of Decryption		
Set	Prove (s)	Verify (ms)	Size (MiB)	Prove (s)	Verify (ms)	Size (MiB)
A	9.19 ± 0.15	306.4 ± 1.1	2.31	1.72 ± 0.08	82.4 ± 3.6	1.17
B	4.46 ± 0.07	159.0 ± 0.4	2.74	0.58 ± 0.01	39.2 ± 0.2	0.84
C	2.54 ± 0.03	108.0 ± 3.2	1.79	0.61 ± 0.01	40.1 ± 1.7	0.82

denote an independently selected key committee. The sets $\mathcal{P}$ and $\mathcal{K}$ may overlap, but the security assumption on $\mathcal{K}$ is independent of the number or stake of corrupted leader candidates in $\mathcal{P}$.

The members of $\mathcal{K}$ jointly execute distributed key generation to produce a collective CKKS public key pk and secret-key shares $\{\mathsf{sk}_j\}_{j\in[m]}$. We instantiate threshold decryption with a t-of-m access structure: any set of at least t valid partial decryptions recovers the election output, whereas fewer than t shares reveal no information about it.

Let f denote the maximum number of corrupted or withholding key committee members. We assume that at most f members of $\mathcal{K}$ are Byzantine and choose parameters satisfying $f < t \leq m - f$. Under the standard Byzantine honest-supermajority assumption, one may choose

$$m = 3f + 1 \qquad \text{and} \qquad t = 2f + 1.$$

Thus, at most f corrupted committee members cannot reconstruct the decryption key or reveal the elected leader early, while the remaining $2f + 1$ honest members suffice to complete decryption and preserve liveness. This assumption concerns only the independently provisioned key and does not require an honest-majority assumption over the larger leader candidate population.

7.2 Leader-Sampled Key Committees

An alternative deployment samples $\mathcal{K}$ from the leader candidate population $\mathcal{P}$. In this case, the corruption bound for $\mathcal{K}$ must be derived probabilistically from the adversarial stake fraction in $\mathcal{P}$. At a fixed stake snapshot, let $\mathcal{P}$ have the total stake

$$W = \sum_{P_i \in \mathcal{P}} w_i.$$

Let $\mathcal{A} \subseteq \mathcal{P}$ be the set of adversarial candidates and let

$$\alpha = \frac{\sum_{P_i \in \mathcal{A}} w_i}{W}$$

denote the adversarial stake fraction. We assume that the adversary commits to its corrupted stake before the committee-selection randomness is available.

A key committee $\mathcal{K} = \{Q_1, \ldots, Q_m\} \subseteq \mathcal{P}$ is sampled from $\mathcal{P}$ using publicly verifiable, stake-weighted random selection. Let $X = |\mathcal{K} \cap \mathcal{A}|$ denote the number of

adversarially controlled key-committee members. For analysis, when committee roles are independently sampled proportionally to stake, X is distributed as

$$X \sim \text{Binomial}(m, \alpha).$$

The committee parameters are selected so that the relevant tail probability is negligible in the security parameter.

All-party threshold decryption. In the default Pastel instantiation, the sampled key committee jointly executes DKG to generate a collective RLWE/RGSW public key pk and one secret-key share per member of $\mathcal{K}$. Thus, final decryption requires a valid partial decryption from every member of $\mathcal{K}$.

In this setting, confidentiality of the threshold secret key and leader secrecy/unpredictability hold provided that at least one sampled key-committee member is honest. That is, under independent stake-weighted sampling the failure event has the probability $\Pr[X = m] = \alpha^m$. Thus, for any $\alpha < 1$, increasing m decreases the probability that the adversary controls all shares.

However, all-party threshold decryption provides no Byzantine liveness guarantee against a sampled adversarial key holder. The liveness event requires that every member of $\mathcal{K}$ be available and submit a valid partial decryption. If corrupted members may withhold messages, then the adversary can prevent decryption whenever $X \geq 1$. Under independent sampling, the probability that no adversarial member is sampled is

$$\Pr[X = 0] = (1 - \alpha)^m.$$

For instance, if $\alpha = 0.1$ and $m = 32$, the chance that at least one adversarial key holder is sampled is approximately 96.6%. That corrupted party may simply withhold its partial decryption. The protocol can slash it or rotate the committee afterwards, leading to an accountable abort model [27, 33]. Pastel provides secrecy against any adversary that does not control all sampled key shares; its liveness is conditional on full key-committee participation; a withholding key holder can abort an election, and can be identified and penalized when partial decryptions are authenticated and publicly verifiable.

***t*-of-*m* threshold decryption.** A more fault-tolerant instantiation replaces the all-party access structure with a t-of-m threshold RLWE/RGSW scheme. Final decryption succeeds once any set of at least t key-committee members produces valid partial decryptions.

Let f be the tolerated number of corrupted key-committee members. We choose (m, t, f) such that

$$f < t \leq m - f,$$

and select m so that

$$\Pr[X > f] \leq 2^{-\lambda_{\mathsf{com}}}.$$

Except with probability $2^{-\lambda_{\mathsf{com}}}$, the adversary then has fewer than t shares and the honest key-committee members can complete decryption without adversarial participation. A natural Byzantine-resilient choice is $m = 3f + 1$ and $t = 2f + 1$; this tolerates f faulty committee members and aligns with standard $< 1/3$-Byzantine assumptions when the DKG requires agreement among committee members. If the chosen threshold RLWE/RGSW DKG requires only honest-majority resilience, other parameters satisfying

$f < t \leq m - f$ may be used. Finally, the committee size m should then be chosen so that stake-weighted sampling yields more than f adversarial members only with negligible probability.

7.3 Accountable Pastel

Secret leader election permits an elected leader to withhold its claim. That is, after Pastel decrypts the selected tag $h^\star$, only the elected party knows a witness x_i such that

$$h^\star = H(\mathsf{ct}, x_i).$$

If this party does not reveal x_i before the leader-reveal deadline, the remaining parties cannot identify the withholding leader. We add weak accountability following the committee-based approach of Christ et al. [11]. At registration, party P_i samples its witness x_i, computes

$$h_i = H(\mathsf{ct}, i, x_i),$$

and, in addition to its ordinary Pastel record, publishes a recovery escrow

$$\mathsf{esc}_i \leftarrow \mathsf{ThrPKE.Enc}_{\mathsf{pk}_{\mathsf{rec}}}(i, x_i, \mathsf{ct}; \rho_i),$$

under a t_{rec}-of-m_{rec} recovery public key generated by a possibly independent recovery committee (it could also be the same key committee, though). The registration proof binds esc_i, h_i, and the encrypted weighted record to the same witness and canonical election context.

If no valid leader claim appears before the deadline, any party may invoke $\mathsf{Recover}(\mathsf{ct}, h^\star)$. A qualified recovery committee decrypts registered escrows until it obtains (i, x_i, ct) satisfying

$$H(\mathsf{ct}, i, x_i) = h^\star.$$

It then publishes the decrypted witness and verifiable threshold decryption shares as evidence that P_i could have claimed leadership but withheld. The proof-of-stake system may slash P_i accordingly.

This mechanism is off the normal election path and preserves leader secrecy against any adversary controlling fewer than t_{rec} recovery shares. As in prior committee-based accountable SSLE, recovery requires an additional trust assumption: a qualified malicious recovery committee could otherwise decrypt escrows prematurely. Recovery may also require scanning registered escrows and therefore triggers a fresh registration epoch after it is invoked.

7.4 Updatable Pastel

To avoid reconstructing $\mathtt{ct}_{\mathrm{ecd}}$ for every election, we retain it and update only the coefficients affected by changes in hashes or weights. We consider updates that preserve the public total weight W, while each individual weight w_i remains private. For example, an update may be required after verifying a revealed witness and refreshing its hash, or when weight is transferred between two parties.

The part corresponding to P_j can be updated as follows. Let $s_{j-1} = \sum_{i=1}^{j-1} w_i$ be the starting position of P_j. Starting from an RLWE encryption of 1, we obtain an encryption of $Y^{s_{j-1}}$ using the ciphertexts $\mathtt{ct}_W^{(i)}$ generated during the setup phase. We then encode the hash correction over the positions to be updated and add the resulting ciphertext to $\mathtt{ct}_{\text{ecd}}$. Alternatively, using $\mathtt{ct}_W^{(i)}$ for $i > j$ and the automorphism $Y \mapsto Y^{-1}$, we can update from the opposite direction. Choosing the shorter direction halves the worst-case number of sequential external products.

7.5 Weight Leakage from Pastel Functionality

If elections are repeated with the same weights and the leader in each epoch can be linked to the same participant across epochs, an observer can statistically estimate the weight w_i from the participant's cumulative selection frequency. This is unavoidable leakage inherent in the observable outputs of the pSSLE functionality. As a mitigation, unlinkability among leaders across epochs can be provided at a communication layer separate from the election protocol by employing *anonymous* communication techniques such as onion routing [12, 20]. Similar observations are made by Ganesh et al. [17] and Kerber et al. [24], both of which likewise note the benefits of anonymous communication as a mitigation. Outside the scope of leader election and more generally in light of private proof-of-stake as a holistic transaction system, however, Kohlweiss et al. [25] find a network-level equivocation attack and conclude that anonymous channels must also be *reliable* in order to mitigate weight leakage in a transaction system as a whole (should Pastel be a building block for one).

References

[1] M. R. Albrecht, R. Player, and S. Scott. "On the Concrete Hardness of Learning with Errors". In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203.

[2] R. Bacho and A. Kavousi. "SoK: Dlog-based distributed key generation". In: *2025 IEEE S&P*. 2025.

[3] M. Backes, P. Berrang, L. Hanzlik, and I. Pryvalov. "A framework for constructing single secret leader election from MPC". In: *European Symposium on Research in Computer Security*. 2022.

[4] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco. "Single secret leader election". In: *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. 2020.

[5] D. Boneh, A. Partap, and L. Rotem. "Post-quantum single secret leader election (ssle) from publicly re-randomizable commitments". In: *Advances in Financial Technologies*. 2023.

[6] D. Boneh et al. "Threshold cryptosystems from threshold fully homomorphic encryption". In: *CRYPTO*. 2018.

[7] J. Burdges, E. Crites, H. K. Alper, A. Stewart, and S. Vasilyev. "Sassafras: efficient batch single leader election". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2025, pp. 244–273.

[8] D. Catalano, D. Fiore, and E. Giunta. "Adaptively secure single secret leader election from DDH". In: *ACM PODC*. 2022.

[9] D. Catalano, D. Fiore, and E. Giunta. "Efficient and universally composable single secret leader election from pairings". In: *PKC*. 2023.

[10] K. Choi, A. Manoj, and J. Bonneau. “SoK: Distributed Randomness Beacons”. In: *IEEE Security & Privacy*. 2023.

[11] M. Christ, K. Choi, W. McKelvie, J. Bonneau, and T. Malkin. “Accountable Secret Leader Election”. In: *Advances in Financial Technologies*. 2024.

[12] R. Dingledine, N. Mathewson, and P. Syverson. “Tor: The Second-Generation Onion Router”. In: *13th USENIX Security Symposium (USENIX Security 04)*. San Diego, CA: USENIX Association, Aug. 2004. URL: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[13] Y. Dodis and A. Yampolskiy. “A verifiable random function with short proofs and keys”. In: *International Workshop on Public Key Cryptography*. 2005.

[14] H. Dou, P. Ni, Y. Gao, and J. Xu. “Mobius: Enabling Byzantine-Resilient Single Secret Leader Election with Uniquely Verifiable State”. In: *NDSS*. 2026.

[15] A. Fiat and A. Shamir. “How to prove yourself: Practical solutions to identification and signature problems”. In: *Conference on the theory and application of cryptographic techniques*. 1986.

[16] L. Freitas et al. “Homomorphic sortition–single secret leader election for pos blockchains”. In: *Cryptology ePrint Archive* (2023).

[17] C. Ganesh, C. Orlandi, and D. Tschudi. “Proof-of-stake protocols for privacy-aware blockchains”. In: *Eurocrypt*. 2019.

[18] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. “Secure distributed key generation for discrete-log based cryptosystems”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. 1999.

[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. “Algorand: Scaling byzantine agreements for cryptocurrencies”. In: *Proceedings of the 26th Symposium on Operating Systems Principles*. 2017.

[20] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. “Hiding Routing information”. In: *Information Hiding*. Ed. by R. Anderson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 137–150. ISBN: 978-3-540-49589-5.

[21] I. Hwang, H. Lee, J. Seo, and Y. Song. “Practical Zero-Knowledge PIOP for Maliciously Secure Multiparty Homomorphic Encryption”. In: *ACM CCS 2025: 32nd Conference on Computer and Communications Security*. Ed. by C.-Y. Huang, J.-C. Chen, S.-P. Shieh, D. Lie, and V. Cortier. Taipei, Taiwan: ACM Press, 2025, pp. 4049–4063. DOI: 10.1145/3719027.3765229.

[22] I. Hwang, S. Min, J. Seo, and Y. Song. “On the Security and Privacy of CKKS-Based Homomorphic Evaluation Protocols”. In: *Advances in Cryptology – ASIACRYPT 2025, Part VII*. Ed. by G. Hanaoka and B.-Y. Yang. Vol. 16251. Lecture Notes in Computer Science. Melbourne, VIC, Australia: Springer, Singapore, Singapore, 2025, pp. 296–330. DOI: 10.1007/978-981-95-5122-4_10.

[23] A. Kavousi, Z. Wang, and P. Jovanovic. “SoK: Public Randomness”. In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. 2024.

[24] T. Kerber, A. Kiayias, M. Kohlweiss, and V. Zikas. “Ouroboros crypsinous: Privacy-preserving proof-of-stake”. In: *IEEE Security & Privacy*. 2019.

[25] M. Kohlweiss, V. Madathil, K. Nayak, and A. Scafuro. “On the anonymity guarantees of anonymous proof-of-stake protocols”. In: *IEEE Security & Privacy*. 2021.

[26] *Lattigo v6*. Online: https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA. Aug. 2024.

[27] H. Liang, Z. Liu, Y. Wang, X. Xie, Y. Yu, and F. Zhang. “Relect: single secret leader election via FHE with reduced computation and communication and transparent setup”. In: *ACM CCS*. 2026.

[28] V. Lyubashevsky, C. Peikert, and O. Regev. "On ideal lattices and learning with errors over rings". In: *Eurocrypt*. 2010.
[29] S. Micali, M. Rabin, and S. Vadhan. "Verifiable random functions". In: *IEEE 40th Annual Symposium on Foundations of Computer Science*. 1999.
[30] D. Micciancio and Y. Polyakov. "Bootstrapping in FHEW-like cryptosystems". In: *Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. 2021.
[31] S. Min, G. Hanrot, J. H. Park, A. Passelègue, and D. Stehlé. "Distributed Key Generation for Efficient Threshold-CKKS". In: *ACM CCS*. 2026.
[32] T. P. Pedersen. "A threshold cryptosystem without a trusted party". In: *Workshop on the Theory and Application of of Cryptographic Techniques*. 1991.
[33] Y. Wang and F. Zhang. "Qelect: Lattice-based single secret leader election made practical". In: *USENIX Security*. 2025.
[34] G. Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Ethereum Project Yellow Paper. 2014. URL: `https://ethereum.github.io/yellowpaper/paper.pdf`.
[35] Zama. *Fhevm: A cross-chain protocol for confidential smart contracts*. `https://raw.githubusercontent.com/zama-ai/fhevm/main/fhevm-whitepaper.pdf`. 2025.

A Supplementary Preliminary

A.1 RLWE/RGSW-based Homomorphic Encryption

Let $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ be the cyclotomic ring, and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for an integer q.

Definition 6 (RLWE Encryption). *The* RLWE encryption scheme *is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *described below:*

$\mathsf{RLWE.KeyGen}(1^\lambda) \to (\mathsf{pp}, \mathsf{sk}, \mathsf{pk})$ *takes a security parameter λ and outputs public parameters* pp*, a secret key* sk*, and a public key* pk*. It chooses a power-of-two integer N and distributions χ_{sk}, χ_{err}, and χ_{Enc} over $\mathcal{R}$ for the secret key, error, and encryption randomness, respectively. It then samples* $\mathsf{sk} \leftarrow \chi_{\mathsf{sk}}$, $a \xleftarrow{\$} \mathcal{R}_{pq}$, *and* $e \leftarrow \chi_{\mathsf{err}}$, *and sets* $\mathsf{pk} = (-a \cdot \mathsf{sk} + e, a) \in \mathcal{R}_{pq}^2$.

$\mathsf{RLWE.Enc}(\mathsf{pk}, \mu) \to \mathtt{ct}$ *takes a public key* pk *and a message $\mu \in \mathcal{R}$, and outputs* $\mathtt{ct} = v \cdot \mathsf{pk} + (\Delta \cdot \mu + e_0, e_1) \in \mathcal{R}_q^2$, *sampling* $v \leftarrow \chi_{\mathsf{Enc}}$ *and* $e_0, e_1 \leftarrow \chi_{\mathsf{err}}$.

$\mathsf{RLWE.Dec}(\mathsf{sk}, \mathtt{ct}) \to \mu$ *takes a secret key* sk *and a ciphertext* $\mathtt{ct} = (c_0, c_1) \in \mathcal{R}_q^2$, *and outputs* $\mu = \left\lfloor \frac{1}{\Delta}(c_0 + \mathsf{sk} \cdot c_1 \bmod q) \right\rceil \in \mathcal{R}$.

For the RGSW encryption scheme, fix a gadget vector $\mathbf{g} \in \mathbb{Z}^d$ and a gadget decomposition function $\boldsymbol{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}^d$ satisfying $\langle \boldsymbol{g}^{-1}(a), \mathbf{g} \rangle = a \pmod q$ and $\|\boldsymbol{g}^{-1}(a)\|_\infty \le \mathcal{B}$ for a small bound $\mathcal{B}$. The gadget decomposition $\boldsymbol{g}^{-1}$ naturally extends to $\boldsymbol{g}^{-1} : \mathcal{R}_q \to \mathcal{R}^d$.

Definition 7 (RGSW Encryption). *The* RGSW encryption scheme *is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *described below, where* KeyGen *is the same as in the RLWE encryption scheme of Definition 6:*

$\mathsf{RGSW.Enc}(\mathsf{pk}, \mu) \to \mathtt{ct}$ *takes a public key* pk *and a message $\mu \in \mathcal{R}$ and outputs*

$$\mathtt{ct} = \mathbf{v} \cdot \mathsf{pk}^T + \mu \cdot \begin{bmatrix} p\mathbf{g} & 0 \\ 0 & p\mathbf{g} \end{bmatrix} + \mathbf{E} \in \mathcal{R}_{pq}^{2d \times 2},$$

sampling $\mathbf{v} \leftarrow \chi_{\mathsf{Enc}}^{2d}$ *and* $\mathbf{E} \leftarrow \chi_{\mathsf{err}}^{2d \times 2}$.

$\mathsf{RGSW.Dec}(\mathsf{sk}, \mathtt{ct}) \to \mu$ *takes a secret key* sk *and a ciphertext* $\mathtt{ct} = \begin{bmatrix} \mathbf{b} & \mathbf{a} \\ \mathbf{b}' & \mathbf{a}' \end{bmatrix} \in \mathcal{R}_{pq}^{2d\times 2}$, *and outputs* $\mu = \left\lfloor \frac{1}{p} \langle \mathbf{g}^{-1}(1), \mathbf{b} + \mathsf{sk} \cdot \mathbf{a} \rangle \right\rceil \pmod q \in \mathcal{R}$.

Definition 8 (External Product with RGSW Ciphertext). *Let* $\mathtt{ct}_{\mathsf{RLWE}} = (b, a) \in \mathcal{R}_q^2$ *be an RLWE ciphertext. Let* $\mathtt{ct}_{\mathsf{RGSW}} \in \mathcal{R}_{pq}^{2d\times 2}$ *be an RGSW ciphertext:*

$\mathsf{RGSW.Enc}(\mathsf{pk}, \mu) \to \mathtt{ct}_{\mathsf{RGSW}}$ *takes a public key* pk *and a message* $\mu \in \mathcal{R}$ *and outputs* $\mathtt{ct}_{\mathsf{RGSW}} = \mathbf{v} \cdot \mathsf{pk}^T + \mu \cdot \begin{bmatrix} p\mathbf{g} & 0 \\ 0 & p\mathbf{g} \end{bmatrix} + \mathbf{E} \in \mathcal{R}_{pq}^{2d\times 2}$, *sampling* $\mathbf{v} \leftarrow \chi_{\mathsf{Enc}}^{2d}$ *and* $\mathbf{E} \leftarrow \chi_{\mathsf{err}}^{2d\times 2}$.

Then the external product $\boxtimes : \mathcal{R}_q^2 \times \mathcal{R}_{pq}^{2d\times 2} \to \mathcal{R}_q^2$ *is defined as:*

$$\mathtt{ct}_{\mathsf{RLWE}} \boxtimes \mathtt{ct}_{\mathsf{RGSW}} := \left\lfloor \frac{1}{p} \left[\boldsymbol{g}^{-1}(b)^T \mid \boldsymbol{g}^{-1}(a)^T \right] \cdot \mathtt{ct}_{\mathsf{RGSW}} \right\rceil.$$

We present additional homomorphic operations for automorphisms and trace on $\mathcal{R}$. For an odd integer $j \in \mathbb{Z}_{2N}^\times$, let σ_j denote the automorphism over $\mathcal{R}$ sending X to X^j, and define a trace map onto a subring $\mathcal{R}' \subset \mathcal{R}$ by $\mathsf{Tr}_{\mathcal{R}'}(m) = \sum_{\sigma_i \in \mathsf{Gal}(\mathcal{R}/\mathcal{R}')} \sigma_i(m)$, where $\mathsf{Gal}(\mathcal{R}/\mathcal{R}')$ denotes the corresponding Galois group.

Definition 9 (Automorphism and Trace). *The RLWE encryption admits the following homomorphic operations.*

$\mathsf{Auto}_j(\mathtt{ct}; \mathsf{ak}_j)$ *takes an RLWE ciphertext* $\mathtt{ct} = (b, a) \in \mathcal{R}_q^2$ *and an automorphism key* $\mathsf{ak}_j = \mathsf{RGSW.Enc}(\mathsf{pk}, \sigma_j(\mathsf{sk}))$, *and outputs* $(\sigma_j(b), 0) + (0, \sigma_j(a)) \boxtimes \mathsf{ak}_j \in \mathcal{R}_q^2$.

$\mathsf{Tr}_{\mathcal{R}'}(\mathtt{ct}; \{\mathsf{ak}_j\}_{j \in \mathsf{Gal}(\mathcal{R}/\mathcal{R}')\setminus\{1\}})$ *takes an RLWE ciphertext* $\mathtt{ct} \in \mathcal{R}_q^2$ *and a collection of automorphism keys, and outputs* $\mathtt{ct} + \sum_{j \in \mathsf{Gal}(\mathcal{R}/\mathcal{R}')\setminus\{1\}} \mathsf{Auto}_j(\mathtt{ct}; \mathsf{ak}_j)$.

In particular, if $\mathtt{ct} = \mathsf{RLWE.Enc}(\mathsf{pk}, \mu)$, *we have* $\mathsf{RLWE.Dec}(\mathsf{sk}, \mathsf{Auto}_j(\mathtt{ct}; \mathsf{ak}_j)) = \sigma_j(\mu)$ *and* $\mathsf{RLWE.Dec}(\mathsf{sk}, \mathsf{Tr}_{\mathcal{R}'}(\mathtt{ct}; \{\mathsf{ak}_j\})) = \mathsf{Tr}_{\mathcal{R}'}(\mu)$.

A.2 Threshold Fully Homomorphic Encryption

Definition 10 (Threshold Fully Homomorphic Encryption). *Let* $\mathcal{M}$ *denote the plaintext space,* $\mathcal{C}$ *the ciphertext space,* $\mathcal{C}_{\mathsf{dec}}$ *the processed ciphertext space, and* $\mathcal{M}_{\mathsf{share}}$ *the space of partial decryption shares. A* Threshold Fully Homomorphic Encryption (ThFHE) scheme *is a tuple of PPT algorithms:*

$\mathsf{ThFHE.Setup}(\lambda, n, t) \to (\mathsf{pp}, \mathsf{pk}, \{\mathsf{sk}_i, \mathsf{pk}_i\}_{i\in[n]})$ *takes as input* λ, *the number of parties* n, *and the threshold parameter* t, *and outputs public parameters* pp, *a public key* pk, *and secret-public key shares* $\{\mathsf{sk}_i, \mathsf{pk}_i\}_{i\in[n]}$.

$\mathsf{ThFHE.Enc}(\mathsf{pk}, \mu) \to \mathtt{ct}$ *takes as input a public key* pk *and a plaintext* $\mu \in \mathcal{M}$, *and outputs a ciphertext* $\mathtt{ct} \in \mathcal{C}$ *encrypting* μ.

$\mathsf{ThFHE.Eval}(\mathsf{pk}, \mathcal{F}, \mathtt{ct}_1, \cdots, \mathtt{ct}_k) \to \mathtt{ct}'$ *takes an evaluation key* pk, *a circuit* $\mathcal{F} : \mathcal{M}^k \to \mathcal{M}$, *and* k *number of ciphertexts* $\mathtt{ct}_1, \cdots, \mathtt{ct}_k$ *as inputs, and outputs ciphertext* $\mathtt{ct}'$.

$\mathsf{ThFHE.PartDec}(\mathsf{pk}, \mathtt{ct}, \mathsf{sk}_i) \to \mathsf{pd}_i$ *takes as input a public key* pk*, a secret key share* sk_i*, and a ciphertext* $\mathtt{ct} \in C$*, and outputs a partial decryption share* $\mathsf{pd}_i \in \mathcal{M}_{\mathsf{share}}$.
$\mathsf{ThFHE.FinDec}(\mathsf{pk}, \{(i, \mathsf{pd}_i)\}_{i\in\gamma}) \to \mu$ *takes as input a public key* pk *and a set of partial decryptions* $\{(i, \mathsf{pd}_i)\}_{i\in\gamma}$ *where* $|\gamma| \geq t$*, and outputs a plaintext* $\mu \in \mathcal{M} \cup \{\perp\}$.

Definition 11 (Correctness of ThFHE). *A ThFHE scheme* $\mathsf{ThFHE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *is* correct *if the following probability is overwhelming:*

$$\Pr\left[\mathsf{FinDec}(\{p_i\}_{i\in[n]}) = \mu' \;\middle|\; \begin{array}{r} (\mathsf{pp}, \{\mathsf{sk}_i\}_{i=1}^n, \mathsf{pk}) \leftarrow \mathsf{Setup}(\lambda, n, t) \\ \mathtt{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_i), \forall i \in [a] \\ \mathtt{ct}' \leftarrow \mathsf{Eval}(\mathsf{pk}, C, \{\mathtt{ct}_i\}_{i\in[a]}) \\ \{p_i\}_{i\in[n]} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathtt{ct}') \end{array}\right],$$

where $\mu' = \mathsf{C}(\mu_1, \cdots, \mu_a)$.

Definition 12 (Security of ThFHE). *A ThFHE scheme* $\mathsf{ThFHE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *is* secure *if it satisfies the following two properties:*

- *The underlying encryption scheme* $(\mathsf{Setup}, \mathsf{Enc})$ *is IND-CPA-secure;*
- *There exists a PPT simulator* Sim *such that, for every PPT adversary* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{ThFHE},\mathcal{A}}(\lambda) := \left|\Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{real}}_{\mathsf{ThFHE}}(1^\lambda)) = 1] - \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{ThFHE}}(1^\lambda)) = 1]\right|.$$

Here, $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{ThFHE}}$ *denotes the real threshold decryption experiment using the actual secret key shares* $\{\mathsf{sk}_i\}_{i\in[n]}$*, while* $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{ThFHE}}$ *denotes the ideal experiment in which the partial decryptions are generated by the simulator [6, Definition 5.5].*

A.3 Non-Interactive Zero-Knowledge Proof

Definition 13 (NIZK). *Let* R *be a NP relation. A non-interactive zero-knowledge proof (NIZK) in ROM is represented by the following tuple of algorithms:*

$\mathsf{NIZK.Setup}(\lambda) \to \mathsf{crs}$ *takes the security parameter* λ *and outputs a common reference string* crs*. We implicitly assume* crs *as inputs in the sequel.*
$\mathsf{NIZK.Prove}^{\mathsf{RO}}(\mathsf{x}, \mathsf{w}) \to \pi$ *takes* crs*, statement-witness pair* $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$*, and given access to random oracle* RO*, outputs the proof* π.
$\mathsf{NIZK.Verify}^{\mathsf{RO}}(\mathsf{x}, \pi) \to 0/1$ *takes* crs*, statement* x *and proof* π*, and outputs a bit.*

Definition 14 (Security of NIZK). *A non-interactive zero-knowledge proof should satisfy the following properties:*

Completeness. *For any* $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$*,*

$$\Pr\left[\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{x}, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(\lambda) \\ \pi \leftarrow \mathsf{NIZK.Prove}^{\mathsf{RO}}(\mathsf{crs}, \mathsf{x}, \mathsf{w}) \end{array}\right] \geq 1 - \mathrm{negl}(\lambda).$$

(Adaptive) Knowledge Soundness. *There exists an algorithm $\mathcal{E}$, called a* knowledge extractor*, and a positive polynomial* p *such that the extractor, given statement* x *and oracle access to malicious prover* $\overline{\mathsf{NIZK.Prove}}^{\mathsf{RO}}$ *that queries the random oracle Q times, runs in* $\mathsf{poly}(|\mathsf{x}|, Q)$ *time and outputs a valid witness* w *with*

$$\Pr[(\mathsf{x},\mathsf{w}) \in \mathsf{R} \mid \mathsf{w} \leftarrow \mathcal{E}^{\mathsf{NIZK.Prove}^{\mathsf{RO}}}(\mathsf{x})] \geq \frac{1}{\mathsf{p}(|\mathsf{x}|)} \left(\Pr[\mathsf{NIZK.Verify}^{\mathsf{RO}}(\mathsf{x},\pi) = 1 \mid \pi \leftarrow \overline{\mathsf{NIZK.Prove}}^{\mathsf{RO}}(\mathsf{x},\cdot)] - \mathcal{K}appa(|\mathsf{x}|, Q)\right)$$

where the knowledge error $\mathcal{K}appa : \mathbb{N} \times \mathbb{N} \to [0,1]$ *is negligible.*

Zero-Knowledge. *There exist PPT simulators* $\mathsf{Sim}_1, \mathsf{Sim}_2$ *such that for any* $(\mathsf{x},\mathsf{w}) \in \mathsf{R}$ *and PPT adversary $\mathcal{A}$,*

$$\left| \Pr\left[\mathcal{A}^{\mathsf{NIZK.Prove}^{\mathsf{RO}}(\mathsf{crs},\mathsf{x},\mathsf{w})}(\mathsf{crs}) = 1 \;\middle|\; \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(\lambda)\right] - \Pr\left[\mathcal{A}^{\mathsf{Sim}_2^{\mathsf{RO}}(\mathsf{crs},\mathsf{st},\mathsf{x})}(\mathsf{crs}) = 1 \;\middle|\; (\mathsf{crs},\mathsf{st}) \leftarrow \mathsf{Sim}_1(\lambda)\right] \right| \leq \mathrm{negl}(\lambda).$$

B Correctness Analysis of the Homomorphic Circuit

We show correctness of the homomorphic circuit under sufficient parameter choice through an average-case error analysis. We let both χ_{sk} and χ_{Enc} be the sparse ternary distribution of Hamming weight h, and χ_{err} be the discrete Gaussian distribution with variance σ^2_{err}. We write σ^2_{sk}, σ^2_{Enc} for the coefficient variances of χ_{sk}, χ_{Enc}; $\sigma^2_{\mathsf{sk}} = \sigma^2_{\mathsf{Enc}} = h/N$ in this setting. We assume the standard independence heuristic.

B.1 Encryption and Registration

By definition, a fresh RLWE encryption has error of form $v \cdot e_{\mathsf{pk}} + e_0 + e_1 \cdot \mathsf{sk}$ and a fresh RGSW encryption has error of form $\mathbf{v} \cdot e_{\mathsf{pk}} + \mathbf{E} \cdot \left[1 \;\; \mathsf{sk}\right]$, where e_{pk} is the error of the public key regarded as an encryption of zero. Hence, we obtain the coefficient error variance of fresh encryptions as follows.

Proposition 1 (Error variance of fresh encryptions). *Fresh encryptions* $\mathtt{ct}_{\mathsf{RLWE}} = \mathsf{RLWE.Enc}(\mathsf{pk},\mu)$, $\mathtt{ct}_{\mathsf{RGSW}} = \mathsf{RGSW.Enc}(\mathsf{pk},\mu')$ *have coefficient error variance of*

$$\sigma^2_{\mathsf{RLWE}} = \sigma^2_{\mathsf{RGSW}} = \sigma^2_{\mathsf{err}}\left(N\sigma^2_{\mathsf{Enc}} + 1 + N\sigma^2_{\mathsf{sk}}\right) = \sigma^2_{\mathsf{err}}(2h+1).$$

In particular, the weight ciphertext $\mathtt{ct}^{(i)}_W$ in public parameters and the registered ciphertexts $\mathtt{ct}^{(i)}_H$, and $\mathtt{ct}^{(i)}_R$ all have the error variance of $\sigma^2_{\mathsf{err}}(2h+1)$.

B.2 Finalization

Lemma 1 (Error Variance of External Product). *Let* $\mathtt{ct}_{\mathsf{RLWE}} = \mathsf{RLWE.Enc}(\mathsf{pk}, \mu)$ *and* $\mathtt{ct}_{\mathsf{RGSW}} = \mathsf{RGSW.Enc}(\mathsf{pk}, \mu')$ *have coefficient error variances* σ^2_{in} *and* σ^2_{RGSW}, *respectively. Then, the external product defined in Definition 8 encrypts* $\mu\mu'$, *and its coefficient error variance satisfies*

$$\sigma^2_{\mathsf{out}} \le \|\mu'\|_2^2 \sigma^2_{\mathsf{in}} + \frac{2dN\mathcal{B}^2}{3p^2}\sigma^2_{\mathsf{RGSW}} + \frac{1 + N\sigma^2_{\mathsf{sk}}}{12}.$$

Proof. Let

$$\mathbf{G} := \begin{bmatrix} \mathbf{g}\ 0 \\ 0\ \mathbf{g} \end{bmatrix}, \qquad \boldsymbol{d} := \left[\boldsymbol{g}^{-1}(b)^T \mid \boldsymbol{g}^{-1}(a)^T \right].$$

Write $\mathtt{ct}_{\mathsf{RGSW}} = p\mu'\mathbf{G} + \mathbf{Z}$, where $\mathbf{Z}$ is an RGSW encryption of zero. By the gadget decomposition property, $\boldsymbol{d} \cdot \mathbf{G} = (b, a)$. For each row $\mathbf{z}_i$ of $\mathbf{Z}$, let $e_i := \mathbf{z}_i(1, \mathsf{sk})^T$ be its phase error. Let $\boldsymbol{\rho} = (\rho_0, \rho_1)$ denote the error introduced by coefficient-wise rounding in the external product.

Since $b + a \cdot \mathsf{sk} = \Delta\mu + e_{\mathsf{in}}$, the phase of the output ciphertext is

$$(\mathtt{ct}_{\mathsf{RLWE}} \boxtimes \mathtt{ct}_{\mathsf{RGSW}})(1, \mathsf{sk})^T = \Delta\mu\mu' + \underbrace{\mu' e_{\mathsf{in}}}_{\text{input RLWE error}} + \underbrace{\frac{1}{p}\sum_{i=1}^{2d} d_i e_i}_{\text{RGSW error}} + \underbrace{\rho_0 + \rho_1 \mathsf{sk}}_{\text{rounding error}}.$$

Under the standard independence heuristic, the three error terms have coefficient variances

$$\|\mu'\|_2^2 \sigma^2_{\mathsf{in}}, \qquad \frac{2dN\mathcal{B}^2}{3p^2}\sigma^2_{\mathsf{RGSW}}, \qquad \frac{1 + N\sigma^2_{\mathsf{sk}}}{12},$$

respectively. Summing them proves the claim. ∎

By Lemma 1, the contribution of RGSW ciphertext with error variance $V\sigma^2_{\mathsf{RGSW}}$ in an exterior product is

$$\sigma^2_{\mathsf{ext}}(V) := \frac{2dN\mathcal{B}^2}{3p^2} V\sigma^2_{\mathsf{RGSW}} + \frac{1 + N\sigma^2_{\mathsf{sk}}}{12}.$$

Note that has coefficient error variance of

Lemma 2 (Error variance of updated $\mathtt{ct}_H^{(i)}$**).** *In the line 6 of* Finalize *phase in Fig 1, the updated* $\mathtt{ct}_H^{(i)}$ *has coefficient error variance of* $\sigma^2_{H,i}$, *where*

$$\sigma^2_{H,i} \le 4w_i\sigma^2_{\mathsf{RLWE}} + \sigma^2_{\mathsf{ext}}(CW).$$

Proof. In the external product, the RLWE ciphertext has error variance of σ^2_{RLWE}. Meanwhile, the RGSW ciphertext $\frac{2}{Y-1} \cdot (\mathtt{ct}_W^i - 1)$ encrypts $\mu' = 2(1 + Y + \cdots + Y^{w_i - 1})$, which has norm $\|\mu'\|_2^2 = 4w_i$. Also, the ciphertext has error variance of

$$\left\| \frac{2}{Y-1} \right\|_2^2 \sigma^2_{\mathsf{RGSW}} = CW \cdot \sigma^2_{\mathsf{RGSW}},$$

since $2/(Y-1) = -(1+Y+\cdots+Y^{CW-1})$ has norm CW. Hence, the result follows from Lemma 1. ■

Proposition 2 (Error variance of $\mathtt{ct}'_{\mathsf{ecd}}$). *In the line 10 of Finalize phase in Fig 1, $\mathtt{ct}'_{\mathsf{ecd}}$ has coefficient error variance of*

$$\sigma^2_{\mathsf{ecd}} \;\le\; 4W\,\sigma^2_{\mathsf{RLWE}} \;+\; n\cdot\sigma^2_{\mathsf{ext}}(CW) \;+\; 2n\cdot\sigma^2_{\mathsf{ext}}(1).$$

Proof. The accumulator $\mathtt{ct}_{\mathsf{ecd}}$ starts with a trivial encryption of zero, which obviously have zero error variance.

In the first pass, we update $\mathtt{ct}_{\mathsf{ecd}}$ with $\mathtt{ct}_{\mathsf{ecd}} \boxtimes \mathtt{ct}^{(i)}_W + \mathtt{ct}^{(i)}_H$ over n steps. Since $\mathtt{ct}^{(i)}_W$ encrypts a monomial, each external product adds up to $\sigma^2_{\mathsf{ext}}(1)$ by Lemma 1, and $\mathtt{ct}^{(i)}_H$ contributes up to $4w_i\sigma^2_{\mathsf{RLWE}} + \sigma^2_{\mathsf{ext}}(CW)$ by Lemma 2. Hence, after the first pass, $\mathtt{ct}_{\mathsf{ecd}}$ has error variance up to

$$\sigma^2_{\mathsf{ecd}} \le 4W\cdot\sigma^2_{\mathsf{RLWE}} + n\cdot\sigma^2_{\mathsf{ext}}(CW) + n\cdot\sigma^2_{\mathsf{ext}}(1).$$

Then, the second pass performs n steps of external products $\mathtt{ct}'_{\mathsf{ecd}} \leftarrow \mathtt{ct}'_{\mathsf{ecd}} \boxtimes \mathtt{ct}^{(i)}_R$, which contributes $n\cdot\sigma^2_{\mathsf{ext}}$ to the error variance, giving the result. ■

Remark 4 (Malicious registration). The heuristic bounds above assume honestly generated registrations, matching the semi-honest version of the protocol. Malicious parties may instead choose their error terms rather than sampling them, aligning its contributions across parties. As a result, the adversary may steer the error of registered RLWE ciphertexts $\mathtt{ct}^{(i)}_H$ so that the error is aligned after external product with the known RGSW ciphertext $\mathtt{ct}^{(i)}_W$. Hence, it may contribute nB rather than $O(\sqrt{n}\,\sigma_{\mathsf{RLWE}})$ to the error bound, where B bounds the error of a registered ciphertext.

The RGSW encryptions $\mathtt{ct}^{(i)}_W$ and $\mathtt{ct}^{(i)}_R$ admit no such alignment, since each passes through an external product whose other operand differs at each step and is not known at registration. Hence, the other term is unaffected by the adversary.

Parameters for the maliciously secure protocol should be adjusted accordingly.

In the last step before threshold decryption, the trace is computed as $\mathsf{Tr}_{\mathcal{R}'}(\mathtt{ct}^{(0)}) = \mathtt{ct}^{(L)}$, $L = \log_2(N/C)$, where $\mathtt{ct}^{(j)} := \mathtt{ct}^{(j-1)} + \mathsf{Auto}_{g_j}(\mathtt{ct}^{(j-1)})$, and $\{g_j\}_{j\in[L]}$ are generators of $\mathsf{Gal}(\mathcal{R}/\mathcal{R}')$. To reduce the error variance, we apply the standard trick: we multiply $(N/C)^{-1} \bmod q$ before trace, so that

$$\mathtt{ct}_{\mathsf{out}} = \mathsf{Tr}_{\mathcal{R}'}(((N/C)^{-1} \bmod q)\cdot\mathtt{ct}'_{\mathsf{ecd}}).$$

This way, the error contribution of the input $\mathtt{ct}'_{\mathsf{ecd}}$ to $\mathtt{ct}_{\mathsf{out}}$ is the same as σ^2_{ecd}.

Proposition 3 (Error variance of $\mathtt{ct}_{\mathsf{out}}$). *For $0 \le k < C$, the error of $\mathtt{ct}_{\mathsf{out}}$ at coefficient $k\cdot N/C$ has variance*

$$\sigma^2_{\mathsf{out},k\cdot N/C} \;\le\; \sigma^2_{\mathsf{ecd}} \;+\; \frac{(N/C)^2-1}{3}\,\sigma^2_{\mathsf{ks}}.$$

Proof. We only consider the additional error contribution by the key switching. Given a ciphertext $\mathtt{ct}$ encrypting a message μ and error e, the automorphism $\mathsf{Auto}_g(\mathtt{ct})$ yields a ciphertext with message $\sigma_g(\mu)$ and error $\sigma_g(e) + e_{\mathsf{ks}}$, where the key-switching error e_{ks} has coefficient variance $\sigma^2_{\mathsf{ks}} \leq \sigma^2_{\mathsf{ext}}(1)$. Hence, in the recurrence of the trace computation, we obtain

$$e^{(j)} = e^{(j-1)} + \sigma_{g_j}(e^{(j-1)}) + e_{\mathsf{ks},j}$$

where $e^{(j)}$ is the error of $\mathtt{ct}^{(j)}$. In particular, since σ_{g_j} preserves the $i = k \cdot N/C$-th coefficients, $e^{(j)}_i = 2e^{(j-1)}_i + (e_{\mathsf{ks},j})_i$, so corresponding error variance is $(\sigma^{(j)}_{\mathsf{out},i})^2 = 4(\sigma^{(j-1)}_{\mathsf{out},i})^2 + \sigma^2_{\mathsf{ks}}$. Solving the recurrence equation yields the latter term. ■

To summarize, we obtain the following error variance bound over all the coefficients

$$\sigma^2_{\mathsf{out}} \;\leq\; 4W\sigma^2_{\mathsf{RLWE}} + n\,\sigma^2_{\mathsf{ext}}(CW) + 2n\,\sigma^2_{\mathsf{ext}}(1) \;+\; \frac{(N/C)^2 - 1}{3}\,\sigma^2_{\mathsf{ext}}(1).$$

C Malicious Security of Pastel

We now provide the proof deferred from Theorem 2.

Proof of Theorem 2. Following Section 4.2 in which the two relations $\mathsf{R}_{\mathsf{Enc}}, \mathsf{R}_{\mathsf{Dec}}$ are defined, we note that a malicious adversary may attempt to deviate from Pastel in exactly the two ways specified against by the relations. Assuming a secure NIZK proving these relations, we therefore reduce a malicious adversary for Pastel effectively to a semi-honest one. Accordingly, we apply Theorem 1 to show uniqueness, fairness, and unpredictability. ■

Remark 5 (Rogue Key Committee). As both fairness and unpredictability of Pastel depend on the semantic security of the underlying threshold FHE scheme, we note that this semantic security becomes violated with a *rogue key committee* (i.e. when t or more members of $\mathcal{K}$ are corrupted), the scenario of which we delineate in the following:

- Fairness is compromised when (say) the adversary effectively controlling the threshold FHE secret key can see all the random monomials in plaintext and is allowed to contribute a last-minute random monomial (in the rushing adversary model where it speaks last) to bias the joint randomness.
- Unpredictability is compromised, as the initially contributed hash commitments (as well as the joint randomness) are basically visible to the adversary in plaintext such that the final (winning) hash commitment can easily be linked to the party before any claim of leadership.
- Uniqueness still holds, given a hash function that is collision-resistant.

D Supplementary Implementation Details

D.1 Homomorphic Encryption Scheme Parameters of Pastel

The homomorphic encryption scheme parameters in our implementation are summarized in Table 4.

Table 4: Parameter sets for our implementations. N is the ring dimension, $\log B$ is the size of decomposed hash components, C is the number of components, Δ is scale factor, $\log P, \log Q$ are the bit-size of auxiliary and base modulus, and d is the gadget decomposition count. All sets use a sparse ternary secret of Hamming weight $h = 256$, and discrete Gaussian errors with $\sigma = 3.2$.

Set	N	$\log B$	C	$\log \Delta$	$\log P$	$\log Q$	d
A	2^{14}	128	1	106	180	236	2
B	2^{13}	64	2	93	51	159	4
C	2^{13}	32	4	97	77	132	2

D.2 Malicious Security

We describe the Buckler proof system [21] and our modifications below.

Buckler Proof System. At a high level, Buckler is a proof system for $\mathbb{Z}_P^N$, for a prime of the form $P = b^k + 1$. It gives a PIOP-style proof for the following relations:

$$\mathsf{R}_{\text{lin}} = \{(\mathbf{M}, (\mathbf{a}, \mathbf{b})) \mid \mathbf{b} = \mathbf{M}\mathbf{a}\},\ \mathsf{R}_{\text{arith}} = \{(C, (\mathbf{a}_1, \ldots, \mathbf{a}_\ell, \mathbf{b})) \mid \mathbf{b} = C(\mathbf{a}_1, \ldots, \mathbf{a}_\ell)\}$$
$$\mathsf{R}_{\text{infnm}} = \{(\mathbf{a}, \mathcal{B}) \mid \|\mathbf{a}\|_\infty \le \mathcal{B}\},\ \mathsf{R}_{\text{2nm}} = \{(\mathbf{a}, \mathcal{B}) \mid \|\mathbf{a}\|_2^2 \le \mathcal{B}^2\}$$

where $\mathbf{M} \in \mathbb{Z}_P^{N \times N}$ and $C \in (\mathbb{Z}_P[X])(X_1, \ldots, X_\ell)$. Then, most relations over the ring R_P can be expressed as a composition of these relations, and we can extend this to relations over R_q (or R_{pq}) via modulus switching. We refer to [21] for more details.

Matching the Modulus. In the relation $\mathsf{R}_{\mathsf{Enc}}$, we implicitly use two moduli: q for RLWE encryption, and pq for RGSW encryption. The standard way of handling this mismatch is embedding: we choose a prime $P \approx pq$, and embed elements of R_q into R_p. The arithmetic relation, say $b = -as + e \pmod q$, is translated to

$$b = -as + e + qI \pmod P$$

along with norm check on I that it is small enough to ensure that no overflow happens. However, this requires additional commitment and norm check that can be potentially expensive.

Instead, we simply also create an RLWE encryption on modulus pq. Then, we can match the modulus in the target relation. Note that generating public key and the encryption in pq is a standard practice in FHE to reduce the noise in public-key encryptions.

Norm check on h. The final problem is the norm check on h, which has the largest norm bound. To reduce the cost, we instead prove the norm bound of the binary decomposition on h. Note that the binary decomposition of h has at most 128 coefficients, which is less than N in all relevant parameters. Therefore, we can represent it as a single vector, say $h_d \in \mathbb{Z}_P^N$. Then, we can replace the norm check on h with binary check on h_d with the

Table 5: Running time of the semi-honest protocol. Runtime is reported as mean ± standard deviation over 10 runs. Threshold decryption is independent of n: one committee member's PartDec takes 0.92, 0.59 and 0.30 ms for A, B and C, and FinDec over all $m = 32$ shares 0.86, 0.55 and 0.28 ms.

Metric	n	A	B	C
Register (ms)	–	44.7 ± 1.2	43.0 ± 1.1	15.0 ± 0.4
Finalize (s)	16	0.689 ± 0.006	0.583 ± 0.005	0.209 ± 0.004
	32	1.285 ± 0.014	1.096 ± 0.002	0.392 ± 0.002
	64	2.465 ± 0.020	2.151 ± 0.050	0.769 ± 0.017
	128	4.836 ± 0.028	4.263 ± 0.090	1.502 ± 0.004
	256	9.572 ± 0.053	8.479 ± 0.190	3.006 ± 0.070
	512	19.369 ± 0.171	16.783 ± 0.214	6.028 ± 0.135
	1024	39.858 ± 0.661	34.575 ± 0.511	11.984 ± 0.204
	2048	82.393 ± 0.120	72.753 ± 0.089	24.149 ± 0.390

consistency check with h as follows.

$$h = \begin{bmatrix} \vec{1}_{N/C} & & \\ & \ddots & \\ & & \vec{1}_{N/C} \\ & \mathbf{0} & \end{bmatrix} \quad h_d \wedge h_d \odot (h_d - 1) = \vec{0}$$