Homomorphic Signature-based Witness Encryption and Applications

Alireza Kavousi*[1] and István András Seres†[2]

[1]University College London
[2]Eötvös Loránd University

Abstract

Signature-based witness encryption (SWE) schemes recently emerged as a viable alternative to instantiate timed-release cryptography in the honest majority setting. In particular, assuming threshold trust in a set of parties that release signatures at a specified time, one can "encrypt to the future" using an SWE scheme. SWE offers *stateless decryption*, where parties producing the signatures do not need to know the ciphertexts in advance to perform decryption. This statelessness makes SWE vastly superior to regular threshold encryption for decentralized blockchain applications such as sealed-bid auctions and voting. However, the lack of homomorphism in existing SWE schemes severely bottlenecks their throughput and limits their deployment. In this work, we formally introduce *Homomorphic* SWE (HSWE) to resolve these limitations. We present a systematic suite of HSWE constructions designed to scale across varying message spaces without sacrificing statelessness. We first establish foundational HSWE schemes for binary messages using Cocks IBE and standard Learning With Errors (LWE). To support polynomially-sized message spaces, we construct a pairing-based scheme that features a highly efficient multi-pairing optimization for cross-tag aggregation. For exponentially large message spaces, we transition to a Ring-LWE construction that achieves high-throughput polynomial packing with post-quantum security. Moreover, we address the inherent privacy limitation of public stateless aggregation by proposing a framework that perfectly hides individual plaintexts and only reveals the final aggregated result. We also present an empirical performance evaluation in Rust to highlight the impact of homomorphism in practical deployments.

1 Introduction

Witness encryption (WE) is a powerful cryptographic primitive that enables encrypting a message with respect to some NP statement such that its decryption is done via the corresponding witness. In general, a practical WE scheme for *all NP* seems out of reach as the existing schemes rely on heavy cryptographic machinery and thus are highly impractical. However, there exist several witness encryption schemes that support important NP languages, *e.g.*, see [FHAS25]. As an efficient and promising realization, signature-based witness encryption (SWE) allows encryption with respect to some tag as a statement with its corresponding witness being some signature on it [DHMW23]. SWE is seen by many as a practical way to achieve the functionality of encrypting messages to the future [RSW96]. Recently, practical time-lock encryption schemes [DHMW23, GMR23] have seen a resurgence in applied cryptography and blockchain applications, partly due to the difficulty of deploying primitives using computational puzzles. SWE foregoes these delicate difficulties (see Appendix A for an extensive discussion) and builds efficient timed-release cryptography assuming *threshold trust*. Figure 1 illustrates a high-level overview of the existing approaches to build timed-release cryptography.

Imagine a threshold of trusted parties that release certain information, *e.g.*, signatures, commitment openings, or verifiable random function (VRF) outputs, at a specified future time. In practice, such quorums of parties already exist and are widely used in applications, *e.g.*, validator committees in blockchains. A well-known example is the Drand distributed randomness beacon (DRB) [DRA20] run by the League of Entropy industrial consortium that releases BLS signatures on UNIX timestamps every 30 seconds [GMR23]. Similarly, 32 elected Ethereum validators BLS-sign the epoch number, which is incremented at every 384 seconds. Additional notable services for randomness generation are the Chainlink VRF [BCC+21] or the Supra dVRF service [GHK+24]. One can build

*a.kavousi@cs.ucl.ac.uk
†seresistvanandras@gmail.com

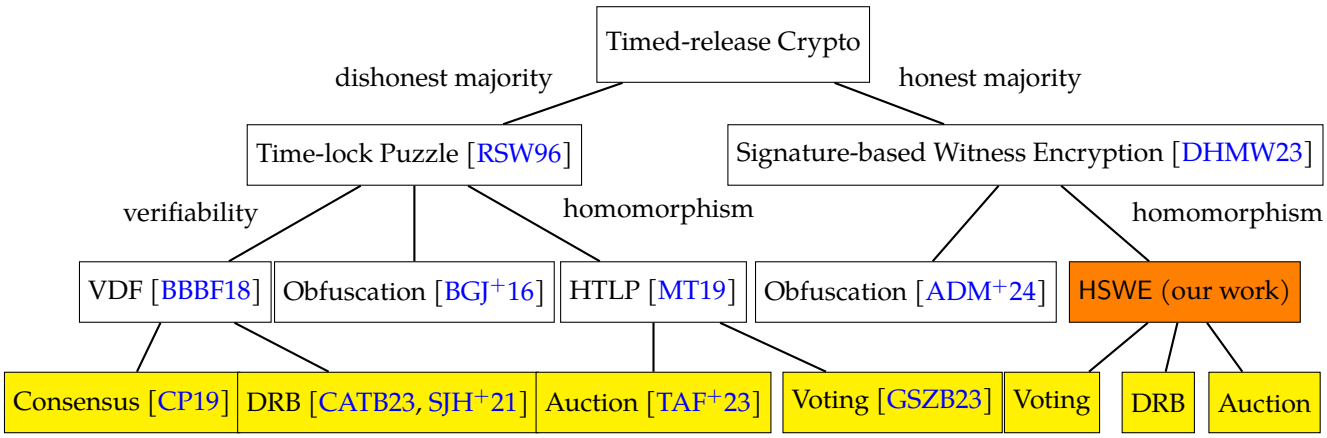


Figure 1: Timed-release cryptography [RSW96] from a bird's-eye view. Only a limited number of papers and schemes are shown due to space constraints. In this paper, we contribute to the honest-majority line of work and extend the applicability and practicability of signature-based witness encryption schemes by adding homomorphism to their message spaces, see the orange node. Unlocked applications of time-based cryptography (including the potential ones due to HSWE) are denoted with yellow color.

timed-release cryptography by encrypting a message m towards a future timestamp or blockchain epoch number ρ, which later can be decrypted by a corresponding valid signature σ (*i.e.*, a witness) on ρ under a (possibly distributed) public key pk. SWE schemes have advantages over common threshold encryption schemes [Des92], making them an excellent choice for use in blockchain settings with dynamic participation. We highlight the two main qualities as follows:

Stateless decryption. Crucially, unlike threshold encryption, SWE allows *stateless* decryption, *i.e.*, the decryption can occur independently from and without prior knowledge about the ciphertexts, as it only requires a signature on some publicly known tag. These properties make SWE an interesting option for being used on blockchains where validators already generate signatures on epoch/block numbers for various purposes (*e.g.*, in Ethereum: RANDAO contributions, attestations), offering an efficient piggyback to have publicly verifiable decryption.

Programmable privacy. The notion of programmable privacy on blockchains is usually realized by combining expressive smart contracts with cryptography. This leads to privacy guarantees that are not fixed once and for all, but can instead be expressed and enforced as part of the application logic itself [BGLM24]. SWE schemes allow implementing simple decryption policies, unlike threshold encryption schemes. Specifically, each ciphertext can only be decrypted if and only if a valid signature is released on a certain message, *e.g.*, round number, a digital contract, etc. In particular, these conditional decryption policies allow the implementation of time-based cryptography as a popular instantiation.

Moreover, SWE schemes hold significant promise for various on-chain applications such as privacy-preserving DAO/governance voting and sealed-bid auctions [TAF+23], primarily due to their communication efficiency. In particular, a batch of SWE ciphertexts of size $\mathcal{O}(B)$ can be decrypted with $\mathcal{O}(n)$ communication cost per validator in a threshold setting that is independent of the batch size. In comparison, a typical threshold encryption scheme would incur an asymptotic cost of $\mathcal{O}(nB)$ for decrypting the same batch.

1.1 Homomorphic Signature-based Witness Encryption

Motivated by the growing demand for efficient and secure solutions in various privacy-preserving on-chain applications (*e.g.*, DAO/governance voting, sealed-bid auctions), we explore how to achieve *homomorphism* in signature-based witness encryptions (HSWE). We design several constructions that are applicable in both threshold and non-threshold settings, offering a range of security-performance trade-offs tailored to different use cases.[1] By incorporating homomorphism, the goal is to significantly boost performance. Existing popular SWE schemes, such as

[1]We present our HSWE constructions under the assumption that a valid signature is provided as the decryption witness. In practice, particularly within blockchain environments, generating this signature relies on a decentralized committee with an honest majority executing a threshold signature protocol, the mechanics of which are orthogonal to our core constructions and thus abstracted away.

the ones based on Identity-Based Encryption (IBE) and BLS signature [DHMW23, GMR23], require a pairing operation for decrypting each individual ciphertext. In contrast, our solution allows for the homomorphic decryption of a batch of ciphertexts of size $\mathcal{O}(B)$ using only a *single pairing*, reducing the computational cost from $\mathcal{O}(B)$ to $\mathcal{O}(1)$. Our work can be considered an efficient alternative for Homomorphic Time-Lock Puzzles (Crypto'19) [MT19], though without suffering from the inherent issues relevant to time-based cryptography, as outlined in Appendix A. In a nutshell, we make the following contributions in this paper.

Formal Definitions & Impossibility Results. We formalize the structural requirements for constructing secure HSWE schemes. We prove an impossibility result: any SWE scheme built upon an encryption scheme with an injective secret-to-public key mapping strictly requires a *unique* signature scheme. This theoretical bound not only explains why our classical constructions successfully pair unique signatures (*e.g.,* BLS) with homomorphic encryption, but also demonstrates how modern lattice-based mappings (which are inherently many-to-one) provide a structural bypass to this limitation.

HSWE **for Binary Message Spaces.** We establish foundational stateless HSWE constructions optimized for single-bit messages, which are highly efficient for applications such as decentralized coin-flipping. We provide two distinct instantiations: a classical scheme based on the Cocks IBE [Coc01] and Rabin's signature [Rab79], and a post-quantum secure scheme based on standard Learning With Errors (LWE) lattices utilizing the pre-image sampleable trapdoor functions of Gentry, Peikert, and Vaikuntanathan [GPV08].

Scaling to Polynomially-Sized Message Spaces. For applications requiring larger, yet bounded payloads, we propose a practical, CPA-secure HSWE scheme based on the Boneh-Franklin IBE [BF01] and BLS signatures [BLS01]. This pairing-based scheme supports a $\mathsf{poly}(\lambda)$-sized message space and enables homomorphic evaluation both under common tags and across *distinct* tags. Crucially, for cross-tag aggregation, we introduce an extraction technique that cancels blinding factors using a multi-pairing, strictly reducing decryption complexity from $\mathcal{O}(k^2)$ to $\mathcal{O}(k)$ time without leaking side information.

Achieving Exponentially Large Message Spaces. To support massive payloads in a stateless setting, we introduce a high-throughput, post-quantum instantiation based on Ring-LWE (RLWE). By packing data into polynomial coefficients, this scheme natively supports the homomorphic aggregation of exponentially large message spaces.

Privacy-Preserving HSWE**.** Beyond message size scaling, we introduce a privacy-preserving HSWE variant. This construction protects the confidentiality of individual plaintexts while still preserving the additive homomorphism required to evaluate the aggregated ciphertexts.

We complement our theoretical constructions with an *empirical performance* evaluation in Rust to highlight the impact of homomorphism in practical deployments.

1.2 Applications of HSWE Schemes

The following applications motivate the design of *homomorphic* SWE schemes.

Privacy-preserving Voting. Homomorphic encryption schemes have a long history of being applied in e-voting schemes. Most voting schemes (*e.g.,* rank choice, Borda-votes, range voting) apply additive scoring functions. Thus, linearly homomorphic encryption schemes suffice to instantiate these popular voting schemes. Notable examples are the application of the Paillier encryption scheme or the exponential ElGamal encryption scheme in Helios [Adi08]. HSWE schemes enable efficient voting schemes by reducing the tallying cost of $\mathcal{O}(n)$ decryption operations to typically an $\mathcal{O}(1)$ decryption since the secret ballots can be homomorphically aggregated into a single ciphertext.

Privacy-preserving Auctions. In auctions, similarly to voting schemes, users' bids can be homomorphically aggregated into a single aggregated bid using an HSWE scheme. At the end of the auction, it is enough to decrypt a single aggregated ciphertext instead of decrypting each and every user's encrypted bid. Furthermore, this design when applying HSWE enables a one-round protocol, unlike a regular commit-reveal auction design. As a concrete example, HSWE could be used as an alternative to the linearly homomorphic time-lock puzzles used in building one-round sealed-bid auctions [GSZB23].

Secure On-chain Randomness Beacons. Distributed randomness beacons (DRB) are extensively applied and deployed in lotteries, proof-of-stake consensus algorithms, games, and many more [KWJ24, CMB23, RG22]. Recently, it has been shown that unbiasable distributed randomness beacons in the dishonest majority setting require delay functions [BBCE24]. Again, the delay functionality can be achieved using SWE schemes in the honest majority setting. Homomorphism comes into play in this application when each participant's randomness contribution could be "squashed" homomorphically into a single ciphertext. Once all randomness contributions are received, homomorphism allows participants to decrypt a single aggregated ciphertext encompassing all randomness contributions instead of decrypting every single contribution. As a concrete scenario, we now briefly sketch such an unbiasable DRB design for the Ethereum Beacon Chain based on an HSWE scheme. Crucially, this design allows for generating secure *on-chain* randomness that can mitigate RANDAO manipulation [AW24, NTSL25, TLN25].

In particular, the 32 validators of an epoch e could encrypt their random contributions $\{R_i^e\}_{i=1}^{32}$ towards a verification key pk using our HSWE scheme. We can take two approaches to instantiate this verification key pk:

External service. In principle, an external service's (*e.g.*, Drand) signature issued for pk on the epoch e could be used for decrypting the validator's randomness contributions. Likely, the reliance on an external service is unwanted for a standalone, high-stake cryptocurrency like Ethereum.

Validator's public key. The corresponding verification key pk could belong to the last validator in the epoch. This approach suffers from liveness issues in case the validator fails publishing the signature on the epoch number e. Without the valid signature from pk, the DRB output could not be decrypted. To tackle this liveness issue, the protocol could demand the (last) validator to secret share their signing key to a designated committee so that the committee can recover the signature when needed. Alternatively, the $\{R_i^e\}_{i=1}^{32}$ could be encrypted towards multiple public keys for increased robustness.

We leave the cryptographic and engineering details of such (on-chain) DRB protocol as future work.

1.3 Technical Overview

A design principle of the existing SWE schemes is built upon having an identity-based encryption (IBE) scheme whose decryption key is a valid signature produced by a suitable signature scheme. Both [DHMW23, GMR23] follow this design paradigm based on the Boneh-Franklin IBE scheme [BF01] and the BLS signature scheme [BLS01]. We follow their footsteps for our suite of stateless HSWE schemes. Particularly, we expand this recipe across different mathematical foundations to systematically scale the supported message space. We build two foundational HSWE schemes for one-bit messages based on the Cocks IBE scheme and Rabin's signature [Rab79], as well as the post-quantum lattice-based GPV IBE [GPV08]. We then scale to polynomially-sized message spaces utilizing a homomorphic variant of the Boneh-Franklin IBE. Going further, we show that one can achieve HSWE for exponentially large message spaces by transitioning to Ring-LWE (RLWE) over polynomial rings, explicitly avoiding the algebraic vulnerabilities of weaker commutative public-key hybrids. We also present a framework to turn our baseline HSWE into a privacy-preserving variant. We provide more technical details on our constructions as follows.

1.3.1 HSWE for Binary Message Spaces

We propose two foundational schemes optimized for single-bit payloads, useful in applications like decentralized coin-flipping or binary voting.

From Cocks IBE and Rabin Signatures. This classical scheme supports a binary message space, *i.e.*, $m \in \{\pm 1\}$, with XOR-homomorphism. It builds on the observation that the Cocks IBE scheme's decryption key can be viewed as a Rabin signature on a suitable message. The ciphertexts are made homomorphic using the techniques introduced by Clear et al. [LaV16].

From GPV IBE and Hash-and-Sign Signatures. To provide a post-quantum counterpart, we build a lattice-based HSWE from the pre-image sampleable trapdoor function (PSF) that underlies the Gentry, Peikert, and Vaikuntanathan (GPV) framework [GPV08]. For each identity tag ρ, we set the target $y = H(\rho)$ and use a short preimage x (satisfying $Ax \equiv y \pmod q$) as the decryption key. The encryption follows an LWE-style structure, where the ciphertext is a noisy linear encryption with the message bit embedded as a large offset. Thanks to the linear structure of LWE, ciphertexts under the same identity can be added component-wise, correctly decrypting to the parity of the underlying bits provided the accumulated noise remains bounded.

1.3.2 Scaling to Polynomially-Sized Message Spaces

Due to the inherent conflict between homomorphism and the strongest notion of CCA security [AGHV25], the best we can achieve is an efficient CPA-secure HSWE scheme. To support larger payloads, we modify the Boneh-Franklin IBE scheme by lifting the messages into the exponent. This enables the additive homomorphism of ciphertexts, though it restricts the message space to a $\mathsf{poly}(\lambda)$-size, as the decryptor must solve a small discrete logarithm instance. While homomorphic evaluation typically occurs under a common tag, we demonstrate that one can homomorphically aggregate ciphertexts under *distinct* tags. Naively, canceling the blinding factors for k distinct tags introduces cross-terms that require $\mathcal{O}(k^2)$ pairings. We resolve this by introducing a multi-pairing extraction technique that bypasses cross-term evaluation, perfectly recovering the aggregated plaintext in strict $\mathcal{O}(k)$ linear time without leaking side information. We prove the CPA-security of our scheme under the Decisional Bilinear Diffie-Hellman (DBDH) assumption.

1.3.3 Achieving Exponentially Large Message Spaces

To address the open problem of supporting arbitrarily large payloads in a strictly stateless architecture, we extend our lattice-based construction to the Ring Learning With Errors (RLWE) setting. By transitioning the foundational LWE assumption to cyclotomic polynomial rings $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, we can pack exponentially large messages into the coefficients of a single polynomial. Because RLWE natively separates the blinding factor from the message space via polynomial addition—rather than relying on vulnerable commutative exponentiation hybrids—it achieves high-throughput homomorphic aggregation safely. The decryption key remains a single short vector extracted via a ring-based GPV trapdoor, preserving the highly desirable stateless nature of the signer. We prove this extension is CPA secure under the Decision-RLWE assumption in the random oracle model.

1.3.4 Privacy-Preserving HSWE

An inherent property of our stateless HSWE constructions is that once the signature σ for a specific tag is released, *any* individual ciphertext ct_i encrypted under that tag can be decrypted. However, in some applications, such as decentralized voting or sealed-bid auctions, individual submissions (*i.e.*, ballots or bids) must remain strictly confidential even after the epoch ends and the aggregated result is revealed. To address this, we introduce a privacy-preserving variant. The core idea is to randomize each user i's ciphertext ct_i with an additional, user-specific blinding factor s_i. The user then secret-shares s_i among the distributed signing parties. At decryption time, the signing parties jointly reconstruct only the *sum* of the blinding factors, $\sum s_i$, corresponding to the exact batch of aggregated ciphertexts. Given that no individual s_i is ever reconstructed, the underlying individual ciphertexts remain perfectly hidden, while the aggregated ciphertext can still be successfully decrypted.

2 Background

2.1 Notations

We denote by $\lambda \in \mathbb{N}$ the security parameter and by $x \xleftarrow{\$} \mathcal{S}$ an element x being randomly sampled from a set $\mathcal{S}$. We further denote the set of integers $\{1, \ldots, n\}$ by $[n]$. By $\langle g \rangle = \mathbb{G}$, we denote g as a generator of the cyclic elliptic curve group $\mathbb{G}$ with its (scalar) finite field being $\mathbb{F}$. We consider probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ and denote by $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ any polynomial or negligible functions running in the security parameter.

2.2 Bilinear Pairings

A bilinear pairing is a mapping between three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The pairing is said to be symmetric when $\mathbb{G}_1 = \mathbb{G}_2$ and has the following properties:

Bilinearity: It requires that for all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$, and $a, b \in \mathbb{F}_p$: $e(u^a, v^b) = e(u, v)^{ab}$

Non-degeneracy: It requires that $e(g_1, g_2) \neq 1$, where g_1, g_2 are the generators for $\mathbb{G}_1, \mathbb{G}_2$.

Efficiency: It requires that the mapping $e(\cdot, \cdot)$ be computed efficiently.

A Type-III bilinear group requires that there is no efficiently computable mapping between $\mathbb{G}_1$ and $\mathbb{G}_2$. Without loss of generality, we consider a symmetric pairing where $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$. Our schemes can be easily adapted to the asymmetric setting, *i.e.*, Type-III bilinear group.

2.3 Computational Assumptions

Here, we present the cryptographic assumptions over bilinear groups that our pairing-based constructions rely upon for their security. Let $\mathsf{bg} = (e, p, \mathbb{G}, \mathbb{G}_T, g, g_T)$ be a bilinear group where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map, and p is a λ-bit prime.

Computational Diffie-Hellman (CDH) Assumption. The CDH assumption holds in $\mathbb{G}$ if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr[\mathcal{A}(g, g^\alpha, g^\beta) = g^{\alpha\beta} : \alpha, \beta \xleftarrow{\$} \mathbb{F}_p] \leq \mathsf{negl}(\lambda) \ .$$

Decisional Diffie-Hellman (DDH) Problem. The DDH problem in a group $\mathbb{G}$ requires an adversary to distinguish between valid Diffie-Hellman tuples and random tuples. The DDH assumption holds if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Big| \Pr[\mathcal{A}(g, g^\alpha, g^\beta, g^{\alpha\beta}) = 1 : \alpha, \beta \xleftarrow{\$} \mathbb{F}_p] - \Pr[\mathcal{A}(g, g^\alpha, g^\beta, g^\gamma) = 1 : \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{F}_p] \Big| \leq \mathsf{negl}(\lambda) \ .$$

Gap Diffie-Hellman (GDH) Group. We say that $\mathbb{G}$ is a Gap Diffie-Hellman group if the Decisional Diffie-Hellman (DDH) problem is easy in $\mathbb{G}$, but the CDH assumption still holds. In our setting, $\mathbb{G}$ naturally forms a GDH group because the bilinear map e provides an efficient algorithm to solve the DDH problem (by checking if $e(g^\alpha, g^\beta) = e(g, g^\gamma)$), while computing $g^{\alpha\beta}$ is assumed to remain intractable. The security of the BLS signature relies on $\mathbb{G}$ being a GDH group.

Bilinear Diffie-Hellman (BDH) Assumption. The BDH assumption holds if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr[\mathcal{A}(g, g^\alpha, g^\beta, g^\gamma) = e(g, g)^{\alpha\beta\gamma} : \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{F}_p] \leq \mathsf{negl}(\lambda) \ .$$

It is helpful to note that the BDH assumption is an extension of the CDH assumption to bilinear groups. Its hardness implies the hardness of the CDH problem in both $\mathbb{G}$ and $\mathbb{G}_T$. For instance, if the CDH problem in $\mathbb{G}$ could be efficiently solved to obtain $g^{\alpha\beta}$, one could trivially solve the BDH instance by computing $e(g^{\alpha\beta}, g^\gamma) = e(g, g)^{\alpha\beta\gamma}$.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption. The DBDH assumption holds if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Big| \Pr[\mathcal{A}(g, g^\alpha, g^\beta, g^\gamma, e(g, g)^{\alpha\beta\gamma}) = 1 : \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{F}_p] - \Pr[\mathcal{A}(g, g^\alpha, g^\beta, g^\gamma, e(g, g)^\eta) = 1 : \alpha, \beta, \gamma, \eta \xleftarrow{\$} \mathbb{F}_p] \Big| \leq \mathsf{negl}(\lambda) \ .$$

2.4 BLS Signature

The BLS signature was introduced by Boneh, Lynn, and Shacham [BLS01] and consists of the following algorithms:

$\mathsf{BLS.KeyGen}(1^\lambda)$**.** It samples a secret key $\mathsf{sk} \xleftarrow{\$} \mathbb{F}_p$ and sets the public key $\mathsf{pk} = g^{\mathsf{sk}} \in \mathbb{G}$.

$\mathsf{BLS.Sign}(\mathsf{sk}, m)$**.** It computes the signature $\sigma := H(m)^{\mathsf{sk}} \in \mathbb{G}$, where $H : \{0,1\}^* \to \mathbb{G}$ is a hash-to-curve function, and modeled as a random oracle.

$\mathsf{BLS.Verify}(\mathsf{pk}, m, \sigma)$**.** It outputs 1 if $e(\sigma, g_2) = e(H(m), \mathsf{pk})$ holds, and 0 otherwise.

The BLS signature scheme enjoys uniqueness and also satisfies a strong unforgeability property under the co-CDH assumption [BLS01]. In a threshold setting [Bol02], the secret key sk is shared among a set of signers via a distributed key generation (DKG) setup phase, where anyone can generate a partial signature $\sigma_i (= g^{\mathsf{sk}_i})$ using their share sk_i, similar to the single-signer scheme. Any threshold subset of partial signatures can then produce the threshold signature $\sigma = H(m)^{\mathsf{sk}}$ via Lagrange interpolation in the exponent.

2.5 Shamir Secret Sharing

A (t, n) Shamir secret sharing [Sha79] allows a dealer to distribute a secret $s \in \mathbb{F}_p$ among a set of n shareholders via $\mathsf{Share(s)} \to \{s_1, \ldots, s_n\}$, where the dealer samples $f(x) \in_R \mathbb{F}_p^t[X]$ such that $f(0) = s \wedge \forall i \in [n] : f(i) = s_i$. The secret s can only be uniquely reconstructed by at least $t+1$ shares $\mathsf{Recon}(s'_{\chi_1}, \ldots, s'_{\chi_{t+1}}) \to s$, while no information about the secret is revealed otherwise. Reconstruction of the secret can be performed by Lagrange interpolation. In particular, the Lagrange-basis polynomials $l_j(x) = \prod_{\substack{1 \le m \le t+1 \\ m \neq j}} \frac{x - \chi_m}{\chi_j - \chi_m}$ allow the reconstruction of the secret as $s = f(0) = \sum_{j=1}^{t+1} s'_{\chi_j} l_j(0)$.

2.6 Number Theoretical Assumptions

We build upon the following number theoretic assumptions in $\mathbb{Z}_N^*$ for a semiprime N with unknown factorization.

Definition 1 (Factoring Assumption)**.** Let define P_λ as the set of all prime numbers with λ-bit security. Then for all PPT algorithms $\mathcal{A}$ we have:

$$\Pr[\mathcal{A}(N) = (p, q) : p \xleftarrow{\$} P_\lambda, q \xleftarrow{\$} P_\lambda, N = p \cdot q] \le \mathsf{negl}(\lambda)$$

The quadratic residuosity problem is arguably the oldest cryptographic assumption and was described by Gauss in Disquisitiones Arithmeticae in 1801. In modern cryptography, it was first used by Goldwasser and Micali to build a probabilistic encryption scheme [GM19]. Informally, the quadratic residuosity assumption states that in an RSA group, it is computationally infeasible to distinguish between quadratic residues and non-residues without knowing the group's order.

Definition 2 (Quadratic Residuosity Assumption)**.** We say that deciding quadratic residuosity is hard relative to a group generator algorithm $N \xleftarrow{\$} GGen(\lambda)$ if for all PPT algorithms $\mathcal{A}$, there exists a $\mathsf{negl}(\cdot)$ such that:

$$\left| \Pr\left[\mathcal{A}(N, \mathsf{qr}) = 1 \ : \mathsf{qr} \xleftarrow{\$} \mathsf{QR}_N\right] - \Pr\left[\mathcal{A}(N, \mathsf{qnr}) = 1 \ : \mathsf{qnr} \xleftarrow{\$} \mathsf{QNR}_N\right] \right| \le \mathsf{negl}(\lambda),$$

where QR_N and QNR_N denote the set of quadratic residues and non-residues, respectively.

2.7 Rabin Signature Scheme

The Rabin signature is a probabilistic signature scheme whose security guarantee has been shown to be equivalent to the complexity of integer factorization [Rab79]. It is standardized in [Kal00], and defined as follows.

Definition 3 (Rabin signature scheme)**.** We assume the existence of a cryptographically secure hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$. The Rabin signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ consists of the following three efficient algorithms.

$\mathsf{Rabin.KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$**.** The $\mathsf{KeyGen}(\cdot)$ algorithm outputs $\mathsf{sk} = (p, q)$ λ-bit primes and $\mathsf{pk} = (n, b)$, where $n = p \cdot q$ and $0 \le b < n$. Finally, let $d := b/2 \mod n$.

$\mathsf{Rabin.Sign}(\mathsf{sk}, m) \to \sigma$**.** First, the signer samples $u \xleftarrow{\$} \mathbb{Z}_N^*$. The signer computes $c := H(m, u)$. If $c + d^2 \mod n$ is a quadratic non-residue, then start over[2] by sampling random u. Otherwise, the signer computes $x \mod n$ using the Chinese Remainder Theorem (CRT) such that $x^2 + b \equiv H(m, u) \mod n$ is satisfied. Return: $\sigma = (u, x)$.

$\mathsf{Rabin.Verify}(\mathsf{pk}, m, \sigma) \to \{0, 1\}$**.** Parse $\mathsf{pk} = (n, b)$ and $\sigma = (x, u)$. The verifier returns 1 if $x^2 + b \equiv H(m, u) \mod n$, and 0 otherwise.

In the Rabin signature scheme, one can choose b to be any constant residue class $\mod n$. For simplicity, we chose $(n, 0)$.

[2] This happens with probability 0.75 at each round. Therefore, after 4 iterations, on average, the signer finds a quadratic residue as required.

2.8 Lattice-Based Assumptions

To support our post-quantum HSWE constructions and their scaling to exponentially large message spaces, we rely on standard lattice problems alongside pre-image sampleable trapdoor functions.

Decisional LWE and RLWE Assumptions. Let $n \geq 1$ be a lattice dimension, $q \geq 2$ a prime modulus, and χ a discrete Gaussian error distribution. Let $s \xleftarrow{\$} \mathbb{Z}_q^n$ be a uniformly random secret vector. The Decision-LWE$_{n,q,\chi}$ assumption states that for any PPT adversary, any polynomial number of samples of the form $(A, As + e \bmod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, where $A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and $e \leftarrow \chi^m$, is computationally indistinguishable from uniformly random samples in $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$.

To support exponentially large message spaces efficiently, we rely on the Ring-LWE (RLWE) variant [LPR10]. Let $R = \mathbb{Z}[X]/(X^d+1)$ be a cyclotomic polynomial ring where d is a power of 2, and $R_q = R/qR$. Let $s \leftarrow \chi$ be a short secret polynomial. The Decision-RLWE$_{d,q,\chi}$ assumption asserts that polynomial samples $(a_i, a_i \cdot s + e_i \bmod q) \in R_q \times R_q$, where $a_i \xleftarrow{\$} R_q$ and $e_i \leftarrow \chi$, are computationally indistinguishable from uniform pairs in $R_q \times R_q$.

Pre-Image Sampleable Trapdoor Functions (PSF). Our lattice-based extraction mechanisms utilize the GPV framework [GPV08], abstracted as a PSF. A PSF is a family of one-way functions $\mathcal{F} = \{f_A\}$ equipped with efficient algorithms $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{SamplePre})$. $\mathsf{Gen}(1^\lambda)$ outputs a public evaluation index A and a corresponding trapdoor T. While evaluating $f_A(x)$ is easy, finding a preimage x for a given y is computationally hard without T (One-Wayness). However, given T and any $y \in \mathrm{Im}(f_A)$, the algorithm $\mathsf{SamplePre}(T, y)$ efficiently outputs a correct preimage $x \in f_A^{-1}(y)$. Crucially for security, the output distribution of $\mathsf{SamplePre}$ is statistically close to a fixed distribution (e.g., a discrete Gaussian) that is completely independent of the specific trapdoor T used.

3 Homomorphic SWE

This section introduces the notion of *homomorphic* SWE (HSWE). First, we recall the definition of plain SWE scheme. Afterwards, we define an HSWE scheme and present a few results on requirements for its building blocks.

3.1 Definition

Definition 4 (Signature-based Witness encryption (SWE))**.** Given an existentially unforgeable signature scheme under chosen-message attacks $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, we define the SWE encryption scheme $\mathcal{E}_\Sigma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$**.** Given the security parameter 1^λ, this probabilistic algorithm outputs pp. The public parameters pp describe the message space $\mathcal{M}$, the tag space $\mathcal{R}$, and the description of the cryptographic groups. Implicitly, all following algorithms take pp as input.

$\mathsf{Enc}(m, \rho, \mathsf{pk}) \to \mathsf{ct}$**.** The encryption algorithm takes as input the message $m \in \{0,1\}^*$, a tag $\rho \in \{0,1\}^\lambda$, and a public key pk from the underlying signature scheme Σ. The encryption algorithm outputs the ciphertext ct.

$\mathsf{Dec}(\mathsf{ct}, \rho, \sigma) \to m$**.** Given the ciphertext ct, the tag ρ, a signature σ, the decryption algorithm outputs a message m.

Note that in the decryption algorithm $\mathsf{Dec}(\cdot)$, the signature σ acts essentially as the decryption secret key. We provide formal security definitions for SWE in Appendix B.

Definition 5 (Homomorphic SWE (HSWE))**.** Given an existentially unforgeable signature scheme under chosen-message attacks $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, a homomorphic SWE scheme is a tuple of PPT algorithms $\mathcal{E}_\Sigma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a set of circuits where $C \in \mathcal{C}_\lambda$ and (Setup, Enc, Dec) are defined as in Definition 4.

$\mathsf{Eval}_{\mathsf{pk},\rho}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to \mathsf{ct}$**.** A probabilistic algorithm that takes as input a circuit $C \in \mathcal{C}_\lambda$, a set of ciphertexts $\{\mathsf{ct}_i\}_{i=1}^n$ and outputs a ciphertext ct with respect to some pk and ρ.

Note that the above HSWE definition allows homomorphism for any efficiently computable, probabilistic circuit $C \in \mathcal{C}_\lambda$. However, this work focuses solely on additive homomorphism, which has already unlocked interesting applications. We leave the exploration of other homomorphism types to future work.

Definition 6 (HSWE Correctness)**.** Given a set of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, an HSWE scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is correct if $\forall\lambda \in \mathbb{N}, \forall C \in \mathcal{C}_\lambda, \forall\rho \in \mathcal{R}$ and for all inputs/plaintexts $(m_1, \ldots, m_n) \in \mathcal{M}^n$, the following condition holds:

$$\Pr[\mathsf{Dec}(\mathsf{Eval}_{\mathsf{pk},\rho}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_n), \rho, \sigma) \neq C(m_1, \ldots, m_n)] \leq \mathsf{negl}(\cdot)(\lambda)$$

3.2 HSWE from Unique Signatures

It is well-known that trapdoor permutations imply public-key encryption [KL07]. For example, the RSA $f(x) = x^e \mod N$ or the Rabin $f(x) = x^2 \mod N$ trapdoor permutations imply the RSA and Rabin encryption schemes. Note that in both cases, $f(\cdot)$ is assumed to be a one-way function, but it can be efficiently inverted with a trapdoor (*i.e.,* the factorization of N). We show that a large class of (H)SWE schemes using trapdoor permutations cannot be built from a non-unique signature scheme.

Lemma 1. There is no SWE scheme $\mathcal{E}_\Sigma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ built from a trapdoor permutation $\Pi = (\mathsf{Gen}, \mathsf{Sample}, f, \mathsf{Inv})$ that applies a non-unique signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Proof. Assume towards contradiction that there exists such an SWE scheme $\mathcal{E}_\Sigma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ that is built on a non-unique signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Let σ_0, σ_1 be two different, valid signatures on a tag ρ. For a message m, the ciphertext $f(m)$ must be correctly decryptable with two different signatures σ_0, σ_1. For our assumed SWE scheme, decryption is carried out via the $\mathsf{Inv}_{\mathsf{td}}(\cdot)$ function for some trapdoor td. However, decryption fails $\mathsf{Inv}_{\sigma_0}(f(m)) \neq \mathsf{Inv}_{\sigma_1}(f(m))$ as $b \in \{0, 1\} : \mathsf{Inv}_{\sigma_b}(\cdot)$ is a permutation. □

The previous lemma already rules out several impossible combinations of encryption and signature schemes to build SWE schemes. However, the result of Theorem 1 does not say anything about numerous encryption schemes. Next, we extend the result by considering the public-key encryption algorithm used as part of SWE scheme to be an injective function.

Lemma 2. There is no SWE scheme $\mathcal{E}_\Sigma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with an injective encryption scheme $\mathcal{E}$ (*i.e.,* the underlying $\mathcal{SK} \to \mathcal{PK}$ mapping[3] is injective) that applies a non-unique signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Proof. Let us assume that the Setup algorithm of $\mathcal{E}_\Sigma$ is run correctly. The public encryption key of an $\mathcal{E}_\Sigma$ is a round number ρ.[4] The corresponding decryption key is a valid signature σ on ρ with respect to a *fixed* verification key pk. If there were multiple valid signatures σ_0, σ_1 on ρ under pk, then due to the injectivity of the encryption scheme, there were different public encryption keys ρ_0, ρ_1 corresponding to the decryption keys σ_0, σ_1. This means that one of the decryptions would fail with probability 1, violating the correctness of the SWE scheme. □

Bypassing the Impossibility Result. Lemma 2 establishes a boundary: if the encryption mapping is injective, distinct valid signatures for the same tag would map to incompatible decryption keys, causing correctness to fail. Consequently, to construct a functional SWE (and by extension, HSWE), one must pursue one of the following routes:

1. **Using unique signatures:** Pair an injective encryption scheme with a signature scheme that yields exactly one valid signature per tag (e.g., pairing Boneh-Franklin IBE with BLS signatures).
2. **Using non-injective encryption:** Pair a non-unique signature scheme with an encryption mapping that is inherently "many-to-one," ensuring that multiple distinct valid signatures can successfully unlock the same ciphertext (e.g., lattice-based LWE encryption).

These two routes form the exact blueprint for our constructions in this word. We further contextualize this boundary in the following remarks.

Remark 1. Note that Theorem 2 aligns with the findings of Garg et al. [GKPW24], who show that SWE schemes exist for signature schemes where verification is a public linear constraint system. Theorem 2 implies that one cannot build an SWE with an injective encryption algorithm using a signature scheme with a higher-degree polynomial verification equation, as the existence of multiple valid roots (signatures) for a single tag ρ violates correctness.

This theoretical bound essentially explains the design dichotomy in our own constructions. For our classical schemes, we must pair an injective encryption scheme with a strictly *unique* signature (e.g., the BLS signature

[3] Note there is no explicit key generation algorithm in (H)SWE schemes, but the underlying encryption scheme involves some key generation algorithm.

[4] In our constructions, $f(\cdot)$ is a hash-to-group function, *e.g.,* hash-to-curve.

Setup: $\mathsf{Setup}(1^\lambda) \to \mathsf{pp} := (N, \rho, u)$ such that $N = p \cdot q \wedge p \equiv q \equiv 3 \pmod 4$ and p, q are primes. A public tag ρ (*e.g.*, epoch number or timestamp) and a value u such that the Jacobi symbol $\left(\frac{H(\rho,u)}{N}\right) = 1$.

Encryption: $\mathsf{Enc}(m, \rho, N) \to \mathsf{ct} := c$

1. Sample $t \xleftarrow{\$} \mathbb{Z}_N^*$ such that $\left(\frac{t}{N}\right) = m$.
2. Compute $c = t + \frac{H(\rho,u)}{t} \bmod N$. // Observe that the encryptor needs to know u at encryption time.

Return: $\mathsf{ct} = c$.

Key Extraction (Signing): $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho) \to \pi_\rho := r$, such that $r^2 = H(\rho, u) \bmod N$ // Note that r is a Rabin signature on the public tag ρ. The value u can be published in advance together with ρ.

Decryption: $\mathsf{Dec}(\mathsf{ct}, \rho, \pi_\rho) \to m$

1. Parse the ciphertext ct as c.
2. Let $r := \pi_\rho$.
3. Compute $m = \left(\frac{c+2r}{N}\right)$.

Return: m.

Figure 2: A CPA-secure HSWE scheme for binary message space.

with BF IBE [BF01]). However, our lattice-based HSWE utilizing the GPV signature [GPV08] neatly bypasses this impossibility result. Because the underlying LWE mapping $f_A(x) = Ax \bmod q$ is many-to-one rather than injective, it safely permits the use of the non-unique GPV signatures (where exponentially many short vectors x map to the same tag y), revealing a structural advantage of lattice-based SWE architectures.

Remark 2. The apparent conflict between Garg et al.'s framework [GKPW24] for linearly verifiable signatures and our Theorem 2 is neatly resolved by the specific constraint of injectivity. While we prove that SWE is impossible for randomized (non-unique) signatures when the underlying encryption mapping is injective—since distinct valid signatures would correspond to incompatible decryption keys—Garg et al.'s construction circumvents this exact barrier. They utilize a "many-to-one" verification structure where the encryption targets the invariant result of the linear verification equation, rather than the variable signature bits themselves.[5] Because all valid randomized signatures in a structure-preserving signature scheme (like Boneh-Boyen [BB04]) satisfy this identical linear constraint, they all successfully unlock the same ciphertext, effectively bypassing the injectivity requirement.

4 HSWE for Binary Messages

As a feasibility result, we first introduce an HSWE scheme for one-bit messages, *i.e.*, $m \in \{\pm 1\}$, supporting XOR-homomorphism. Such an HSWE scheme is highly useful in decentralized applications like coin-flipping protocols. Our first instantiation is based on the Cocks IBE scheme [Coc01], which can be made homomorphic using the techniques developed by LaVigne [LaV16].

Recall that the extracted key r satisfies $r^2 = H(\rho, u) \bmod N$, meaning r is a valid Rabin signature on the message tag ρ. We can expand the decryption equation as follows:

$$c + 2r = t + \frac{H(\rho, u)}{t} + 2r = t(1 + 2rt^{-1} + H(\rho, u)t^{-2}) = t(1 + rt^{-1})^2 \bmod N$$

The correctness of the HSWE scheme naturally follows from Section 4, since the Jacobi symbol is multiplicative and

[5] For a randomized signature scheme, let $\Sigma_m = \{\sigma \mid \mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1\}$ be the set of valid signatures for a message m, where $|\Sigma_m| > 1$. Linear verifiability implies that every $\sigma \in \Sigma_m$ satisfies the same pairing equation, e.g., $e(\sigma, Y) = e(g, g)$, where Y is derived from pk and m. The Witness Encryption (WE) constructs the ciphertext mask using the invariant right-hand side, $K = e(g, g)^\alpha$. Decryption succeeds with *any* valid witness $\sigma \in \Sigma_m$ because the bilinearity collapses the variations in the signature bits to the same fixed value: $e(\sigma, Y^\alpha) = e(\sigma, Y)^\alpha = e(g, g)^\alpha = K$.

squares evaluate to 1:

$$\left(\frac{c+2r}{N}\right) = \left(\frac{t(1+rt^{-1})^2}{N}\right) = \left(\frac{t}{N}\right) \cdot \left(\frac{(1+rt^{-1})^2}{N}\right) = m \cdot 1 = m$$

Recall that our ciphertext, *cf.* Figure 2, is fundamentally a Cocks IBE ciphertext for which a simple XOR-homomorphism exists. For full technical details on the homomorphic evaluation of these ciphertexts, we refer to [LaV16].

We note a specific design trade-off in our parameter generation: at the expense of doubling the ciphertext size, one could omit u from the public parameters. Specifically, our strict requirement that $\left(\frac{H(\rho,u)}{N}\right) = 1$ allows us to compress the ciphertext down to a single element c. However, this requires the signing parties to publish u alongside every ρ in advance, since without the secret factorization of N, no external party can compute the quadratic residuosity of the hash. If this slight increase in public parameter size is undesirable for a given application, it can be trivially traded off by reverting to a standard, two-element Cocks ciphertext. In our target applications, reducing strictly on-chain ciphertext sizes is prioritized over parameter sizes, making this optimization highly favorable.

Theorem 3. The HSWE scheme for binary messages presented in Figure 2 is CPA-secure in the random oracle model, assuming the hardness of the Quadratic Residuosity assumption (*cf.* Definition 2).

Proof Sketch. The adversary $\mathcal{A}$'s challenge plaintexts are $m_0, m_1 \in \{-1, 1\}$. Let c^* be the challenge ciphertext encrypting m_b for a random bit $b \in_R \{0, 1\}$, such that decryption yields $\left(\frac{c^*+2r}{N}\right) = m_b$. A straightforward reduction from the Quadratic Residuosity problem demonstrates that the adversary cannot computationally distinguish between the encryptions of $+1$ and -1 without the factorization of N. For the formal sequence of games and the complete proof, we refer to the extensive analysis of the Cocks IBE [Coc01]. □

Remark 3. We highlight that the Rabin signature scheme (see Appendix 2.7) is not a unique signature. Because $N = pq$ is a semiprime, each quadratic residue modulo N has exactly four distinct square roots, implying that each target tag ρ has four different valid Rabin signatures r. Thus, we note that our construction in Figure 2 successfully bypasses the impossibility result established in Theorem 2. It achieves this because the underlying Cocks encryption mapping is non-injective (specifically, the map $t \mapsto t + H(\rho, u)t^{-1}$ is two-to-one). This demonstrates that it is indeed possible to design SWE schemes for non-unique signature schemes, provided they are paired with a structurally compatible, non-injective encryption mechanism.

4.1 A Lattice-Based HSWE

We observe that one can build a stateless HSWE based on the identity-based encryption (IBE) scheme and hash-and-sign digital signature scheme presented by Gentry, Peikert, and Vaikuntanathan (GPV) [GPV08]. Note that the same pre-image sampleable trapdoor function (PSF) underlies both schemes, making them perfectly compatible to build a lattice-based HSWE.

The PSF with Gaussian sampling involves finding a short preimage $x \in \mathbb{Z}_q^m$ for a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$ with trapdoor T, such that it holds $f_A(x) : Ax = y \bmod q$. For an identity tag ρ, one can set the target $y := H(\rho) \in \mathbb{Z}_q^n$ so that the PSF inversion x plays the role of the signature and, consequently, the decryption key for our HSWE. The resulting construction is additively homomorphic for ciphertexts encrypted under the same tag.[6] We must be careful, however, about noise growth, as only a bounded number of additions is correctly decryptable. We present our lattice-based HSWE in Figure 3.

Homomorphic Addition (Aggregation). Given k ciphertexts $c_i = (u_i, v_i)$ encrypting bits b_i under the *same* identity ρ, the aggregator computes:

$$c_\oplus = \sum_{i=1}^{k} c_i = \left(\sum_{i=1}^{k} u_i, \sum_{i=1}^{k} v_i\right) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$$

Decryption with the key x yields:

$$\sum_{i=1}^{k} v_i - \left\langle \sum_{i=1}^{k} u_i,\ x \right\rangle \equiv \lfloor \frac{q}{2} \rfloor \left(\sum_{i=1}^{k} b_i \bmod 2 \right) + \sum_{i=1}^{k} (e_{2,i} - \langle e_{1,i}, x \rangle) \pmod q$$

[6] We present the HSWE for a binary message space, though it can be generalized to larger spaces following the same approach as in [GPV08], or natively expanded via Ring-LWE as detailed in Section 6.

Setup: $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$

The setup generates a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$ and a GPV master trapdoor T. Let $H : \{0,1\}^* \to \mathbb{Z}_q^n$ be a hash function modeled as a random oracle.

Return the public parameters $\mathsf{pp} = (A, q, n, m, H)$ and the master secret key $\mathsf{msk} = T$.

Encryption: $\mathsf{Enc}(b, \rho, \mathsf{pp}) \to \mathsf{ct}$

1. Parse the message bit $b \in \{0,1\}$. Compute the target tag $y = H(\rho) \in \mathbb{Z}_q^n$.
2. Sample a uniformly random secret vector $s \leftarrow \mathbb{Z}_q^n$.
3. Sample small noise errors $e_1 \leftarrow \chi^m$ and $e_2 \leftarrow \chi$ from a discrete Gaussian.
4. Compute $u = A^\top s + e_1 \bmod q$ and $v = \langle y, s \rangle + e_2 + \lfloor \frac{q}{2} \rfloor b \bmod q$.

Return the ciphertext: $\mathsf{ct} = (u, v)$.

Key Extraction (Signing): $\mathsf{Sign}(\mathsf{pp}, \mathsf{msk}, \rho) \to \pi_\rho$

Using the master trapdoor T and the target $y = H(\rho)$, the signing authority runs the preimage sampler to find a short vector $x \in \mathbb{Z}^m$ satisfying $Ax = y \bmod q$.

Decryption: $\mathsf{Dec}(\mathsf{ct}_\oplus, \rho, \pi_\rho) \to b^*$

1. Parse the aggregated ciphertext $\mathsf{ct}_\oplus = (u_\oplus, v_\oplus)$ and the key $x = \pi_\rho$.
2. Compute the message $w = v_\oplus - \langle u_\oplus, x \rangle \bmod q$.
3. Return $b^* = 0$ if w is closer to 0, or $b^* = 1$ if w is closer to $\lfloor \frac{q}{2} \rfloor$.

Figure 3: A lattice-based HSWE for binary message space.

Hence, $c_\oplus$ correctly decrypts to the XOR sum of the bits ($\sum_i b_i \bmod 2$) provided the aggregated noise:

$$\eta_{\text{tot}} = \sum_{i=1}^{k} (e_{2,i} - \langle e_{1,i}, x \rangle)$$

satisfies $|\eta_{\text{tot}}| < q/4$.

Noise Growth and Aggregation Bounds. Let σ be the Gaussian parameter for the error terms e_1, e_2, and let $\|x\|$ be the length bound of the private key (derived from Gaussian trapdoor sampling). The decryption error for a single ciphertext is $\eta = e_2 - \langle e_1, x \rangle$. Because e_1 and e_2 are independent, the variance of this noise term is $\sigma^2(1 + \|x\|^2)$.

When homomorphically aggregating k independent ciphertexts, the errors accumulate. Relying on the sum of independent subgaussian random variables, the variance scales linearly with k, meaning the effective standard deviation grows exactly as $\sigma\sqrt{k(1 + \|x\|^2)}$. To ensure that the aggregated noise satisfies the decryption correctness condition $|\eta_{\text{tot}}| < q/4$ with overwhelming probability $1 - 2^{-\lambda}$ (where λ is the security parameter), we apply a standard Gaussian tail bound constant $C = O(\sqrt{\lambda})$. Thus, the system parameters must be chosen to strictly satisfy:

$$C \cdot \sigma\sqrt{k(1 + \|x\|^2)} < \frac{q}{4}$$

Assuming $\|x\| \gg 1$, this condition simplifies to bounding the maximum number of supported additions k before decryption may fail:

$$\sqrt{k} < \frac{q}{4C\sigma\|x\|}$$

This explicit bound demonstrates that our scheme supports a number of homomorphic additions quadratic in the ratio of the modulus q to the initial noise. We present a detailed analysis on deriving the exact tail bound in Appendix D.

Remark 4. A valid private key is a short vector $x \in \mathbb{Z}^m$ satisfying $Ax = y \bmod q$. The trapdoor sampling procedure outputs one such solution, sampled from a discrete Gaussian over the integer lattice. Thus, there are exponentially many possible private keys for the same identity, differing by lattice vectors. Note that decryption works for *any* x such that $Ax = y \bmod q$, not just the one generated during extraction/signing. However, the error term $\eta = (e_2 - \langle e_1, x \rangle)$ strictly depends on x. A longer-norm key x severely amplifies the noise from e_1, making decryption unreliable. Therefore, the GPV procedure of sampling a specifically *short* x guarantees that the noise remains bounded and decryption succeeds with overwhelming probability.

Theorem 4. Suppose Decision-LWE is hard. Then, for any PPT adversary $\mathcal{A}$ carrying out an IND-CPA attack against our scheme which makes at most q_H random-oracle queries and q_E signing queries, there exists a PPT algorithm $\mathcal{B}$ that solves Decision-LWE with advantage

$$\mathrm{Adv}_{\mathrm{LWE}}^{\mathcal{B}} \;\geq\; \frac{1}{q_H}\mathrm{Adv}_{\mathrm{SWE}}^{\mathcal{A}} - \epsilon_{\mathrm{sim}},$$

where the error ϵ_{sim} is negligible in the security parameter. In particular, since q_H is polynomial, $\mathrm{Adv}_{\mathrm{SWE}}^{\mathcal{A}}$ must be negligible.

Proof. We show that if a PPT adversary $\mathcal{A}$ can break the IND-CPA security of our scheme, we can construct an algorithm $\mathcal{B}$ that solves the Decision-LWE problem.

Setup. $\mathcal{B}$ receives a Decision-LWE challenge from an external challenger: a random matrix $A \in \mathbb{Z}_q^{n \times m}$, a target vector $y^* \in \mathbb{Z}_q^n$, and a challenge ciphertext pair $(u, v^*) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$. $\mathcal{B}$'s goal is to determine if (u, v^*) is a valid LWE sample or uniformly random. $\mathcal{B}$ publishes A as the public parameters. To prepare for the simulation, $\mathcal{B}$ guesses which of the q_H random oracle queries $\mathcal{A}$ will target, picking a random index $j^* \in \{1, \ldots, q_H\}$.

Hash Oracle Queries. $\mathcal{B}$ must answer hash queries for identities ρ_j without knowing a master trapdoor. It does this by working backwards:

- **For the target guess ($j = j^*$):** $\mathcal{B}$ sets $H(\rho_{j^*}) = y^*$ (the LWE target vector).
- **For all other queries ($j \neq j^*$):** $\mathcal{B}$ first samples a short vector x_j from the discrete Gaussian distribution. It then computes $y_j = Ax_j \bmod q$ and sets $H(\rho_j) = y_j$. Because x_j is drawn with a sufficiently large standard deviation, standard lattice properties ensure y_j looks uniformly random [GPV08], perfectly simulating a real random oracle. $\mathcal{B}$ saves the tuple (ρ_j, y_j, x_j).

Key Extraction (Signing) Queries. When $\mathcal{A}$ asks for the private key for an identity ρ_j: If $j = j^*$, $\mathcal{B}$ must abort, as it does not know the key for the LWE target. Otherwise, $\mathcal{B}$ simply looks up the saved tuple and hands $\mathcal{A}$ the short vector x_j. Because $\mathcal{B}$ generated x_j using the correct Gaussian distribution during the hash phase, this perfectly mimics a real key extraction.

Challenge Phase. $\mathcal{A}$ outputs a target identity ρ and two messages m_0, m_1. If $\rho \neq \rho_{j^*}$, our guess was wrong, and $\mathcal{B}$ aborts. If the guess was right, $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and embeds the LWE challenge directly into the ciphertext:

$$c^* = \left(u,\; v^* + \left\lfloor \frac{q}{2} \right\rfloor m_b\right)$$

If the LWE instance is real, c^* is a perfectly valid encryption of m_b. If the LWE instance is uniform, c^* is completely random and hides b.

$\mathcal{A}$ outputs a guess b'. If $b' = b$, $\mathcal{B}$ guesses that the LWE instance was real; otherwise, it guesses uniform. As long as $\mathcal{B}$ correctly guesses the target identity (which happens with probability $1/q_H$), the simulation is perfectly indistinguishable from the real attack game. Therefore, $\mathcal{B}$'s advantage in solving LWE is $\frac{1}{q_H}\mathrm{Adv}_{\mathrm{SWE}}^{\mathcal{A}} - \epsilon_{\mathrm{sim}}$ (where ϵ_{sim} is a negligible statistical error). Since LWE is hard, $\mathcal{A}$'s advantage must be negligible. □

Setup: $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$

Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of large prime order p, equipped with an efficiently computable, non-degenerate bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let $g \in \mathbb{G}$ and $g_T \in \mathbb{G}_T$ be generators of their respective groups. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ be a cryptographically secure hash-to-curve function modeled as a random oracle. The setup samples a master secret key $s \in_R \mathbb{F}_p$ and computes the master public key $P = g^s \in \mathbb{G}$.

Return the public parameters $\mathsf{pp} = (\mathbb{G}, \mathbb{G}_T, p, e, g, g_T, \mathsf{H}, P)$ and the master secret key $\mathsf{msk} = s$.

Encryption: $\mathsf{Enc}(m, \rho, \mathsf{pp}) \to \mathsf{ct}$

1. Parse the public tag ρ (e.g., epoch number or timestamp) and the message $m \in \mathbb{Z}_p$.
2. Sample an ephemeral scalar $r \in_R \mathbb{F}_p$ and compute $U = g^r$.
3. Compute the public target element $g_\rho = e(\mathsf{H}(\rho), P)$.
4. Compute $V = g_T^m \cdot {g_\rho}^r$.

Return the ciphertext: $\mathsf{ct} = (U, V)$.

Key Extraction (Signing): $\mathsf{Sign}(\mathsf{pp}, \mathsf{msk}, \rho) \to \pi_\rho$

Using the master secret key s, the signing authority computes the BLS signature on the tag ρ as $\pi_\rho = \mathsf{H}(\rho)^s \in \mathbb{G}$.

Return the extracted decryption key: π_ρ.

Decryption: $\mathsf{Dec}(\mathsf{ct}_\oplus, \rho, \pi_\rho) \to m^*$

1. Parse the aggregated ciphertext $\mathsf{ct}_\oplus = (U_\oplus, V_\oplus)$ and the extracted key π_ρ.
2. Compute the unblinded target element $Z = V_\oplus \cdot e(\pi_\rho, U_\oplus)^{-1}$.
3. Solve the discrete logarithm problem in $\mathbb{G}_T$ for $Z = g_T^{m^*}$ to obtain the aggregated payload m^*.

Return: m^*.

Figure 4: A CPA-secure HSWE for polynomially-sized message space.

5 HSWE for Polynomially-Sized Messages

In this section, we present a CPA-secure HSWE scheme that supports a $\mathsf{poly}(\lambda)$-sized message space. We adapt the design of the Boneh-Franklin IBE [BF01] to make it additively homomorphic. To achieve this, we embed the message in the exponent of the target group generator. Consequently, the decryptor is required to solve a small discrete logarithm instance during decryption, which strictly bounds the practical message space to a polynomial size. The full scheme is described in Figure 4.

The correctness of the base scheme in Figure 4 follows from standard bilinear properties. We briefly demonstrate its additive homomorphism under a common tag. Let $\mathsf{ct}_1 = (U_1, V_1) = (g^{r_1}, g_T^{m_1} {g_\rho}^{r_1})$ and $\mathsf{ct}_2 = (U_2, V_2) = (g^{r_2}, g_T^{m_2} {g_\rho}^{r_2})$ be two independent ciphertexts encrypted under the same tag ρ. By component-wise multiplying the ciphertexts, we obtain:

$$\mathsf{ct}_\oplus = (U_1 U_2, V_1 V_2) = (g^{r_1+r_2}, g_T^{m_1+m_2} g_\rho^{r_1+r_2}) := (U_\oplus, V_\oplus) \tag{5.1}$$

During decryption, due to the bilinearity of the pairing map where $e(\pi_\rho, U_\oplus) = e(\mathsf{H}(\rho)^s, g^{r_1+r_2}) = e(\mathsf{H}(\rho), g^s)^{r_1+r_2} = g_\rho^{r_1+r_2}$, the blinding factors perfectly cancel out, correctly yielding the homomorphically aggregated plaintext:

$$V_\oplus \cdot e(\pi_\rho, U_\oplus)^{-1} = g_T^{m_1+m_2} \cdot g_\rho^{r_1+r_2} \cdot g_\rho^{-(r_1+r_2)} = g_T^{m_1+m_2}$$

Theorem 5. The HSWE scheme in Figure 4 is IND-CPA secure in the random oracle model under the Decisional Bilinear Diffie-Hellman (DBDH) assumption.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that wins the IND-CPA security game against our HSWE scheme with a non-negligible advantage $\epsilon(\lambda)$. We construct a PPT simulator $\mathcal{B}$ that uses $\mathcal{A}$ to break the DBDH assumption.

Before proceeding with the DBDH reduction, we apply a standard argument from the Boneh-Franklin IBE scheme (Lemma 4.2, [BF01]). Because our extraction mechanism is identical to BF-IBE, we can abstract away the random oracle programming and extraction queries. Specifically, if $\mathcal{A}$ makes at most q_H hash queries, there exists an algorithm that guesses the target challenge identity ρ^* in advance with a probability reduction of $1/q_H$. We assume $\mathcal{B}$ correctly guesses this target tag ρ^*.

The DBDH Reduction: The DBDH challenger $\mathcal{C}$ samples $\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{F}_p$ and a fair coin $b \xleftarrow{\$} \{0,1\}$. It provides $\mathcal{B}$ with the bilinear parameters and the challenge tuple $(A, B, C, T) = (g^\alpha, g^\beta, g^\gamma, T)$. If $b = 0$, $T = e(g,g)^{\alpha\beta\gamma}$. If $b = 1$, $T \xleftarrow{\$} \mathbb{G}_T$. $\mathcal{B}$ simulates the CPA game for $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ sets the master public key as $P = A$, implicitly setting the unknown master secret to $s = \alpha$. It sends pp to $\mathcal{A}$.

Random Oracle Programming: For the target identity ρ^*, $\mathcal{B}$ programs the random oracle such that $\mathsf{H}(\rho^*) = B$, implicitly setting $\mathsf{H}(\rho^*) = g^\beta$.

Challenge: $\mathcal{A}$ submits two equal-length messages $m_0, m_1 \in \mathbb{Z}_p$ for the target tag ρ^*. $\mathcal{B}$ flips a coin $d \xleftarrow{\$} \{0,1\}$ and constructs the challenge ciphertext ct^* using the remaining DBDH components:

$$U^* = C \quad \text{and} \quad V^* = g_T^{m_d} \cdot T$$

$\mathcal{B}$ returns $\mathsf{ct}^* = (U^*, V^*)$ to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs its guess d'. If $d' = d$, $\mathcal{B}$ outputs 0 (guessing T is real). Otherwise, $\mathcal{B}$ outputs 1 (guessing T is random).

Probability Analysis: We analyze $\mathcal{B}$'s probability of winning the DBDH game, $\Pr[\mathcal{B} \to b]$.

- **Case $b = 0$ (Real Tuple):** Here, $T = e(g,g)^{\alpha\beta\gamma}$. Substituting our implicit assignments, the blinding factor is $T = e(g^\alpha, g^\beta)^\gamma = e(P, \mathsf{H}(\rho^*))^\gamma = g_{\rho^*}^\gamma$. Since $U^* = g^\gamma$, this is a perfectly formed ciphertext where the ephemeral randomness is $r = \gamma$. Thus, $\mathcal{A}$ is playing the real IND-ID-CPA game, and its probability of guessing correctly is $\Pr[d' = d | b = 0] = \frac{1}{2} + \epsilon(\lambda)$.
- **Case $b = 1$ (Random Tuple):** Here, T is uniformly distributed in $\mathbb{G}_T$. Consequently, $V^* = g_T^{m_d} \cdot T$ acts as a perfect one-time pad, uniformly distributed in $\mathbb{G}_T$ and completely independent of m_d. In this view, the ciphertext contains no information about the chosen message, so $\mathcal{A}$'s probability of guessing d is exactly $\Pr[d' = d | b = 1] = \frac{1}{2}$.

The overall advantage of $\mathcal{B}$ in solving the DBDH problem is:

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}}^{\mathrm{DBDH}} &= \Big| \Pr[\mathcal{B} \to 0 | b = 0] - \Pr[\mathcal{B} \to 0 | b = 1] \Big| \\
&= \Big| \Pr[d' = d | b = 0] - \Pr[d' = d | b = 1] \Big| \\
&= \Big| \left(\frac{1}{2} + \epsilon(\lambda) \right) - \frac{1}{2} \Big| = \epsilon(\lambda)
\end{aligned}$$

Factoring in the $1/q_H$ guessing penalty from the identity abstraction, $\mathcal{B}$'s final advantage is $\frac{\epsilon(\lambda)}{q_H}$, which is non-negligible. This concludes the proof. □

5.1 HSWE with Homomorphism Across Tags

We observe that our CPA-secure HSWE in Figure 4 also allows for homomorphism across *distinct tags*. That is, one can homomorphically aggregate two (or more) ciphertexts ct_1 and ct_2 generated under two (or more) tags ρ_1 and ρ_2, but under the same public key P. To do so, the idea is to treat the tag as a public key with an unknown discrete logarithm ρ', *i.e.*, $\mathsf{H}(\rho) = g^{\rho'}$. It is useful to remark that the hash-to-curve function $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ in the original scheme also produces an elliptic curve point $\mathsf{H}(\rho)$ with an unknown discrete logarithm [BLS01], and here

we present it with an explicit representation to show how homomorphism works out. In our original definition for HSWE (see Section 5) the homomorphism was merely limited under a common tag. Let $\mathsf{ct}_1, \mathsf{ct}_2$ be two ciphertexts under two distinct tags ρ_1 and ρ_2:

$$\mathsf{ct}_1 = (U_1, V_1) = (g^{r_1}, g_T^{m_1} g_{\rho_1}^{r_1}) = (g^{r_1}, g_T^{m_1} e(g,P)^{\rho_1' r_1})$$
$$\mathsf{ct}_2 = (U_2, V_2) = (g^{r_2}, g_T^{m_2} g_{\rho_2}^{r_2}) = (g^{r_2}, g_T^{m_2} e(g,P)^{\rho_2' r_2})$$

By component-wise aggregating the ciphertexts, we obtain:

$$\mathsf{ct}^* = (g^{r_1+r_2}, g_T^{m_1+m_2} e(g,P)^{\rho_1' r_1 + \rho_2' r_2}) := (U^*, V^*)$$

Given two BLS signatures on tags $\pi_{\rho_1} = g^{\rho_1' s}$ and $\pi_{\rho_2} = g^{\rho_2' s}$ and due to the bilinearity of the pairing map, one can correctly obtain the homomorphically added plaintext as follows:

$$e(\pi_{\rho_2}, U_1) = e(g^{\rho_2' s}, g^{r_1}) = e(g,P)^{\rho_2' r_1}$$

$$e(\pi_{\rho_1}, U_2) = e(g^{\rho_1' s}, g^{r_2}) = e(g,P)^{\rho_1' r_2}$$

Therefore,

$$\begin{aligned} g_T^{m_1+m_2} &= V^* e(\pi_{\rho_1}\pi_{\rho_2}, U^*)^{-1} e(\pi_{\rho_2}, U_1) e(\pi_{\rho_1}, U_2) \\ &= V^* e(P^{\rho_1'+\rho_2'}, g^{r_1+r_2})^{-1} e(\pi_{\rho_2}, U_1) e(\pi_{\rho_1}, U_2) \\ &= V^* e(g,P)^{-(\rho_1' r_1 + \rho_1' r_2 + \rho_2' r_1 + \rho_2' r_2)} e(\pi_{\rho_2}, U_1) e(\pi_{\rho_1}, U_2) \\ &= V^* e(g,P)^{-(\rho_1' r_1 + \rho_1' r_2 + \rho_2' r_1 + \rho_2' r_2)} e(g,P)^{\rho_2' r_1} e(g,P)^{\rho_1' r_2} \\ &= V^* e(g,P)^{-(\rho_1' r_1 + \rho_2' r_2)} \end{aligned}$$

The Linear Optimization via Multi-Pairing. Extending the above approach for aggregation k ciphertext requires canceling out a quadratic $\mathcal{O}(k^2)$ number of cross terms $\rho_i' r_j$ by computing pairing which might be costly. However, we can circumvent this bottleneck entirely. To homomorphically aggregate k ciphertexts under distinct tags, the decryptor computes $V^* = \prod_{i=1}^k V_i$, while keeping ciphertext components $(U_1, \ldots, U_k)$. The decryptor retrieves the aggregated plaintext by computing the multi-pairing $\prod_{i=1}^k e(\pi_{\rho_i}, U_i)$, perfectly canceling the blinding factors. That is, he decryptor computes a linear multi-pairing to perfectly cancel the blinding factors without ever generating cross-terms:

$$V^* \cdot \left(\prod_{i=1}^k e(\pi_{\rho_i}, U_i) \right)^{-1} = g_T^{\sum m_i} \cdot \frac{\prod e(g,P)^{\rho_i' r_i}}{\prod e(g,P)^{\rho_i' r_i}} = g_T^{\sum m_i}$$

This reduces the required number of pairings from quadratic to strictly linear, $\mathcal{O}(k)$. Furthermore, since the resulting aggregated ciphertext $\mathsf{ct}^* = (U_1, \ldots, U_k, V^*)$ consists solely of unmodified components from the original ciphertexts, this approach strictly preserves the CPA security of the underlying scheme without relying on additional cryptographic assumptions. We believe that in a similar fashion, one can claim homomorphism for ciphertexts encrypted towards different public keys $P_1, P_2 \in \mathbb{G}$ but under the same tag $\mathsf{H}(\rho)$.

5.2 Discussions

We just presented a practical HSWE scheme in Section 5. However, the fact that some components of the ciphertexts are in the target group $\mathbb{G}_T$ incurs a few practical challenges that we discuss next.

Proving Statements About the Plaintext. In our primary applications (e.g., sealed-bid auctions or private voting), users must often generate zero-knowledge proofs demonstrating the well-formedness of their ciphertexts—for instance, proving that a vote is a valid boolean value or that a bid falls within a permitted range. Computing these proofs natively in the target group $\mathbb{G}_T$ is computationally prohibitive. To optimize this, the prover can map the target group element V to an equivalent source group element $V' \in \mathbb{G}$. By computing $V' = g^m h^r$ (where $h \in \mathbb{G}$ is a secondary generator with an unknown discrete logarithm with respect to g), the prover can efficiently generate statements about V' using standard Schnorr-based protocols. The prover then relies on established techniques [DHMW23] to publicly prove the consistency between the source group element V' and the target group ciphertext component V.

Efficient Decryption and On-Chain Verification. Decryption natively requires solving a discrete logarithm instance $Z = g_T^m$ in the target group $\mathbb{G}_T$. For end-user devices, this is typically handled via baby-step giant-step algorithms or well-known lookup table techniques. As demonstrated by Chatzigiannis et al. [CCN21], these tables can be significantly compressed, making decryption highly practical even for mobile clients. However, if the aggregated plaintext must be evaluated *on-chain* by a smart contract, computing this discrete logarithm during transaction execution is entirely infeasible. Instead, the aggregation node should compute the discrete logarithm m off-chain. The node then submits m to the smart contract, which mathematically verifies the decryption correctness by executing a native pairing check (e.g., verifying that $V_\oplus = g_T^m \cdot e(\pi_\rho, U_\oplus)$) rather than brute-forcing the exponent itself.

Avoiding $\mathbb{G}_T$ Target Group Operations. For certain applications, the target group operation in $\mathbb{G}_T$ is not available to the app developer or prohibitively expensive to use. Therefore, in most real-world applications, one wants to minimise the number of target group operations. As per the Pectra hard fork activated at 8th April 2025, the Ethereum Virtual Machine (EVM) supports BLS12-381 group operations and the hash-to-curve function [VOSS20]. However, at the time of writing, the EVM still does not support *efficient* $\mathbb{G}_T$ group operations. Implementing $\mathbb{G}_T$ group operations in the EVM is certainly possible. However, arithmetic in $\mathbb{G}_T = \mathbb{F}_{p^k}$ ($k = 12$ in case of the BLS12_-381 curve) might be costly. A future hard fork could enable a precompile contract enacting efficient $\mathbb{G}_T$ group operations. We leave the exploration of these concrete deployment costs to future work.

Thresholdizing HSWE. An advantage of our HSWE framework is that threshold decryption is achieved easily by inheriting the properties of the underlying signature scheme. To thresholdize the HSWE, the trusted committee simply executes a standard Distributed Key Generation (DKG) [BK25] protocol for the target signature scheme (*e.g.,* BLS) to establish the master public key P. During the extraction phase, each committee member broadcasts a partial signature on the tag ρ. Once a threshold $t + 1$ of partial signatures is aggregated, the resulting standard signature π_ρ acts as the singular decryption key. This completely decouples the threshold coordination from the encryption and aggregation logic, allowing for stateless threshold decryption without modifying the HSWE ciphertexts.

6 HSWE for Exponential Message Spaces

While our LWE-based construction efficiently supports binary messages, many real-world applications require encrypting and aggregating large integers or rich data structures. To resolve this without sacrificing statelessness or falling into the algebraic vulnerabilities of classical commutative hybrids (such as Paillier-RSA cross-exponentiation attacks), we transition to the Ring Learning With Errors (RLWE) problem. Because RLWE operates over polynomial rings, we can natively pack exponentially large messages into the coefficients of a single polynomial, achieving high-throughput homomorphic aggregation. Since the underlying extraction mechanics perfectly mirror our binary scheme (*cf.* Section 4.1), we present a compact description highlighting the ring adaptations.

Protocol Description. Let n be a power of 2, defining the cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$. We define the ciphertext quotient ring $R_q = R/qR$ and the plaintext ring $R_t = R/tR$ for moduli $t \ll q$. A single message $m \in R_t$ natively encodes n integers modulo t. The public parameters consist of a random polynomial vector $\mathbf{a} \in R_q^k$ and a random oracle $H : \{0,1\}^* \to R_q$, while the master secret is a GPV ring-trapdoor $T_\mathbf{a}$.

- **Encryption:** To encrypt $m \in R_t$ under tag ρ, compute $y = H(\rho)$. Sample a short secret $s \leftarrow \chi$ and short errors $\mathbf{e}_1 \leftarrow \chi^k$, $e_2 \leftarrow \chi$. The ciphertext is $\mathsf{ct} = (\mathbf{u}, v) \in R_q^k \times R_q$, where:

$$\mathbf{u} = \mathbf{a}s + \mathbf{e}_1 \bmod q \quad \text{and} \quad v = ys + e_2 + \lfloor \frac{q}{t} \rfloor m \bmod q$$

- **Extraction & Evaluation:** The extraction algorithm uses $T_\mathbf{a}$ to sample a short key $\mathbf{x}_\rho \in R^k$ such that $\langle \mathbf{a}, \mathbf{x}_\rho \rangle = y \bmod q$. To aggregate N ciphertexts under the same tag, the evaluator simply computes component-wise polynomial addition: $\mathsf{ct}^* = (\sum \mathbf{u}_i, \sum v_i) = (\mathbf{u}^*, v^*) \bmod q$.

- **Decryption:** The decryptor computes the blinded polynomial $w = v^* - \langle \mathbf{u}^*, \mathbf{x}_\rho \rangle \bmod q$. By scaling w by t/q and rounding the coefficients to the nearest integer, the aggregated message $m^* = \sum m_i \bmod t$ is recovered.

Correctness and Noise Growth. The algebraic cancellation perfectly mirrors the binary scheme. Expanding the decryption equation for the aggregated ciphertext ct^* and applying the key relation $\langle \mathbf{a}, \mathbf{x}_\rho \rangle = y \bmod q$ perfectly cancels the $y \sum s_i$ blinding factors, yielding:

$$w = \lfloor \frac{q}{t} \rfloor \sum_{i=1}^{N} m_i + \sum_{i=1}^{N} e_{2,i} - \left\langle \sum_{i=1}^{N} \mathbf{e}_{1,i}, \mathbf{x}_\rho \right\rangle \bmod q$$

Let the aggregated noise be $\eta_{\text{tot}} = \sum_{i=1}^{N} (e_{2,i} - \langle \mathbf{e}_{1,i}, \mathbf{x}_\rho \rangle)$. Since RLWE noise grows subgaussically, the standard deviation of η_{tot} scales strictly with $\sqrt{N}$. As long as parameters q and t ensure $|\eta_{\text{tot}}| < \frac{q}{2t}$ with overwhelming probability, the rounding operation successfully isolates the homomorphically aggregated plaintext $m^* \bmod t$.

Theorem 6. Suppose the Decision-RLWE problem is hard. Then the proposed RLWE-based HSWE scheme is IND-CPA secure in the random oracle model.

Proof Sketch. The security of this extension follows the identical sequence of hybrid games established for our foundational LWE scheme (*cf.* Theorem 4). The simulation relies on the external Decision-RLWE challenger over the ring R_q rather than standard LWE.

To answer random oracle queries $H(\rho_j)$ without the master trapdoor, the simulator reverse-programs the oracle. It first samples a short polynomial vector $\mathbf{x}_j \leftarrow \mathcal{D}_{R^k,\sigma}$, computes $y_j = \langle \mathbf{a}, \mathbf{x}_j \rangle \bmod q$, and sets $H(\rho_j) = y_j$. By the GPV smoothing lemma generalized to ideal lattices [GPV08], the distribution of y_j is statistically close to uniform in R_q, perfectly mirroring a genuine random oracle. This allows the simulator to cleanly extract valid keys for all non-challenge identities.

For the target identity ρ^*, the simulator embeds the uniform/RLWE challenge pair $(\mathbf{u}^*, v^*)$ directly into the challenge ciphertext. If the adversary can distinguish the simulated ciphertext from random, the simulator successfully breaks the Decision-RLWE assumption with an advantage scaled by the standard $1/q_H$ guessing penalty. □

6.1 Privacy-Preserving HSWE

A potential downside of our baseline stateless schemes is that once the tag signature π_ρ has been released, *every* individual ciphertext generated under that tag becomes publicly decryptable, not just the homomorphically aggregated one. In sensitive applications—such as sealed-bid auctions or decentralized voting—this translates to a complete lack of individual bid or ballot privacy once the epoch concludes. This motivates us to devise a framework that retains the privacy of each individual plaintext indefinitely, allowing for the decryption of *only* the final aggregated ciphertext. We formally define the security experiment for a privacy-preserving HSWE in Figure 6.

To achieve this, we introduce a generic protocol that upgrades our HSWE schemes into a privacy-preserving variant. The underlying cryptographic trick is to blind the message-bearing component of each user's HSWE ciphertext using a random scalar, and then secret-share that scalar among the distributed signing committee. At decryption time, the signing parties do not reconstruct the individual blinding factors; instead, they homomorphically reconstruct only the *aggregated* randomness corresponding strictly to the submitted batch of ciphertexts. They release this aggregated blinding factor alongside the tag signature.

As noted in our technical overview, this transformation inherently comes at the cost of rendering the underlying HSWE scheme *stateful*. The signing parties must now be explicitly aware of the exact set of HSWE ciphertexts being aggregated in order to reconstruct the correct aggregated randomness. For concreteness, we present our privacy-preserving HSWE scheme in Figure 5, instantiating it with prime-order groups and Shamir's secret sharing [Sha79]. However, the technique is generally agnostic and can be extended to polynomial rings (for our RLWE scheme)[7] and other linear secret-sharing variants. In Appendix C, we discuss a verifiable variant of this framework using zero-knowledge proofs.

Theorem 7. The HSWE scheme in Figure 5 guarantees individual ciphertext privacy (IND-CPA), provided that strictly fewer than the threshold t of the signing parties are corrupted by the adversary.

Proof Sketch. The correctness of the aggregated decryption follows from the linearity of both the homomorphic encryption scheme and Shamir's Secret Sharing. Because each individual HSWE ciphertext ct is blinded with a uniformly random scalar s (computing $\mathsf{ct}' = \mathsf{ct} \cdot h^{\mathsf{s}}$), the resulting blinded ciphertext acts as a one-time pad in the target space, making any two individual ciphertexts statistically indistinguishable. Assuming less than a threshold

[7] As polynomial rings contain zero-divisors, standard Shamir's secret sharing cannot be safely applied over the ring itself. Instead, the linear secret sharing is applied coefficient-wise over the base field $\mathbb{Z}_q$, which perfectly preserves both correctness and the required homomorphism.

Public Parameters: A public tag ρ and the underlying HSWE group parameters pp. Let h be a secondary generator in the target message space (e.g., $h \in \mathbb{G}_T$).

Encryption: For a user encrypting message m:

1. Sample a uniformly random blinding scalar $\mathsf{s} \in_R \mathbb{F}_p$.
2. Compute Shamir shares of the blinding factor $\mathsf{Share}(\mathsf{s}) \rightarrow s_1, \ldots, s_n$ and securely transmit s_i to each signing committee member $i \in [n]$.
3. Run the baseline encryption $\mathsf{Enc}(m, \rho, \mathsf{pp}) \rightarrow \mathsf{ct}$.
4. Blind the message-bearing component of the ciphertext to produce $\mathsf{ct}' = \mathsf{ct} \cdot h^{\mathsf{s}}$ (e.g., in the pairing scheme, $U' = U \cdot h^{\mathsf{s}}$).

Return the blinded ciphertext ct'.

Key Extraction: $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho) \rightarrow \pi_\rho$ // Remains identical to the baseline scheme.

Stateful Decryption: For a specific batch of k blinded ciphertexts $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$:

1. Each signing party $i \in [n]$ locally sums their shares for this specific batch: $s^*_i = \sum_{j=1}^{k} s_{j,i}$.
2. A threshold of parties broadcast their aggregated shares to publicly reconstruct the aggregated blinding factor in the exponent: $\mathsf{Recon}(s^*_1, \ldots, s^*_{t+1}) \rightarrow h^{\mathsf{s}^*} = h^{\sum_{j=1}^{k} \mathsf{s}_j}$.
3. The aggregator homomorphically combines the blinded ciphertexts: $\mathsf{ct}^*_{\text{blind}} = \prod_{j=1}^{k} \mathsf{ct}'_j$.
4. The aggregator removes the reconstructed blinding factor: $\mathsf{ct}^* = \mathsf{ct}^*_{\text{blind}} \cdot (h^{\mathsf{s}^*})^{-1}$.
5. Run the baseline decryption $\mathsf{Dec}(\mathsf{ct}^*, \rho, \pi_\rho)$ to recover the aggregate payload $m^* = \sum_{j=1}^{k} m_j$.

Figure 5: A stateful, privacy-preserving HSWE scheme.

of the signing committee is corrupted, the adversary cannot reconstruct any individual s_j. Consequently, no PPT adversary can deduce any information about an individual plaintext m_j from its corresponding ct'_j, even when the aggregated randomness h^{s^*} is published to decrypt the final sum. □

7 Performance

Theoretical Performance. We compare the theoretical performance (public key and message space sizes along with encryption and decryption complexity) of our proposed HSWE schemes in Table 1.

Practical Performance Evaluation. In addition to our theoretical analysis, we provide an empirical performance evaluation to demonstrate the practicality of our pairing- and lattice-based constructions. We evaluate our pairing-based HSWE construction from Section 5, instantiated over the BLS12-381 curve. We focus on this scheme because it supports compact ciphertexts and efficient additive aggregation for bounded message spaces and potentially is our most practical scheme. We implemented the construction in Rust using the Arkworks ecosystem and evaluated encryption, decryption, same-tag aggregation, cross-tag aggregation, and the privacy-preserving same-tag wrapper on a MacBook Pro with an Apple M4 chip and 18 GB of RAM. All timings were obtained in release mode. Our implementation is available under the MIT license at https://github.com/AKavousi/HSWE-Implementation.

The implementation uses a precomputed lookup table to recover the bounded plaintext from the target-group element produced during decryption. We exclude the construction of this table from the reported online timings, since it is a one-time offline preprocessing cost. For a plaintext space bounded by M, the table contains $M + 1$ target-group encodings and supports constant-time hash-map lookup.

Figure 7 reports encryption and decryption times for message lengths from 0 to 16 bits. Encryption requires between 3.53 and 3.87 ms, while decryption requires between 3.02 and 3.31 ms. The costs are effectively independent

Let $\mathcal{C}$ be the challenger and $\mathcal{A}$ the adversary.

1. $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ and generates committee keys. $\mathcal{C}$ sends pp to $\mathcal{A}$.
2. $\mathcal{A}$ makes polynomial queries to the encryption and extraction oracles.
3. $\mathcal{A}$ outputs a target tag $\rho^\star$ and two challenge message vectors of length k: $\mathbf{m}_0 = (m_{0,1}, \ldots, m_{0,k})$ and $\mathbf{m}_1 = (m_{1,1}, \ldots, m_{1,k})$, subject to the strict constraint that their aggregates are equal: $\sum_{j=1}^{k} m_{0,j} = \sum_{j=1}^{k} m_{1,j}$.
4. $\mathcal{C}$ samples a fair coin $b \xleftarrow{\$} \{0,1\}$. For each $j \in [k]$:
   - $\mathcal{C}$ runs $\mathsf{Enc}(m_{b,j}, \rho^*, \mathsf{pp}) \to \mathsf{ct}_{b,j}$.
   - $\mathcal{C}$ samples $\mathsf{s}_j \in_R \mathbb{F}_p$ and computes $\mathsf{ct}'_{b,j} = \mathsf{ct}_{b,j} \cdot h^{\mathsf{s}_j}$.
5. $\mathcal{C}$ gives the set of blinded ciphertexts $\{\mathsf{ct}'_{b,1}, \ldots, \mathsf{ct}'_{b,k}\}$ to $\mathcal{A}$.
6. $\mathcal{A}$ may continue querying oracles (excluding the extraction of individual blinding factors for the challenge ciphertexts) and eventually outputs a guess b'.
7. The experiment outputs 1 if $b' = b$, and 0 otherwise.

Figure 6: The indistinguishability experiment for the privacy-preserving HSWE framework.

HSWE Scheme	Assumption	$\lvert\mathcal{M}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert$	Enc	Dec	Stateless
Cocks IBE (Section 4)	Quadratic Residuosity	2	$\lvert\mathbb{Z}_N\rvert$	$\lvert\mathbb{Z}_N\rvert$	$5\lvert\mathbb{Z}_N\rvert+\mathsf{H}$	$\lvert\mathbb{Z}_N\rvert$	●
LWE (Section 4.1)	Decision-LWE	2	$\lvert\mathbb{Z}_q^{n\times m}\rvert$	$\lvert\mathbb{Z}_q^m\rvert+\lvert\mathbb{Z}_q\rvert$	$O(nm)\lvert\mathbb{Z}_q\rvert+\mathsf{H}$	$O(m)\lvert\mathbb{Z}_q\rvert$	●
Pairing/BLS (Section 5)	DBDH	$\mathsf{poly}(\lambda)$	$\lvert\mathbb{G}\rvert$	$\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert$	$2\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert+\mathsf{P}$	$\lvert\mathbb{G}_T\rvert+\mathsf{P}+\mathsf{DL}$	●
Ring-LWE (Section 6)	Decision-RLWE	$\lvert R_t\rvert = t^n$	$\lvert R_q^k\rvert$	$\lvert R_q^k\rvert+\lvert R_q\rvert$	$O(k\cdot n\log n)\lvert\mathbb{Z}_q\rvert+\mathsf{H}$	$O(k\cdot n\log n)\lvert\mathbb{Z}_q\rvert$	●
Privacy Wrapper (Section 6.1)	Generic (Secret Sharing)	Base $\lvert\mathcal{M}\rvert$	Base $\lvert\mathsf{pk}\rvert$	Base $\lvert\mathsf{ct}\rvert$	$\mathsf{Enc}_{\mathsf{base}}+\mathsf{SS}$	$\mathsf{Dec}_{\mathsf{base}}+\mathsf{Recon}$	○

Table 1: Comparing the theoretical performance of our proposed HSWE constructions. The message space, public key, and ciphertext sizes are denoted by $\lvert\mathcal{M}\rvert$, $\lvert\mathsf{pk}\rvert$, and $\lvert\mathsf{ct}\rvert$, respectively. With a slight abuse of notation, the number of group operations is denoted by $\lvert\mathbb{G}\rvert$ and $\lvert\mathbb{G}_T\rvert$, pairing operations by P, and hashing by H. For lattice schemes, polynomial multiplication via the Number Theoretic Transform dictates the $O(n \log n)$ complexity. The privacy-preserving framework (bottom row) illustrates the explicit cost of transitioning to a stateful protocol using secret sharing (SS) and reconstruction (Recon).

of the message length in this range, since the dominant operations are group exponentiations and pairing computations rather than processing the bounded plaintext value. The right-hand plot shows that naïve individual decryption increases approximately linearly with batch size, whereas decrypting an already aggregated ciphertext remains effectively constant.

We next evaluate the privacy-preserving same-tag aggregation wrapper and report the result in Figure 8. For batch sizes between 1 and 512, final aggregate decryption remains nearly constant, requiring 3.17–3.48 ms. Aggregate-blind reconstruction requires less than 0.01 ms, and unblinding requires 2.08–2.25 ms. The dominant cost is instead the per-ciphertext preprocessing stage, which blinds each ciphertext and creates its Shamir shares. This stage grows from 2.17 ms for one ciphertext to 1059.31 ms for 512 ciphertexts; the total private-tally latency at 512 ciphertexts is 1067.15 ms. Thus, the wrapper preserves the low cost of aggregate decryption while explicitly trading additional per-submission work for individual-ciphertext privacy.

Moreover, we evaluate cross-tag aggregation, where ciphertexts are encrypted under distinct tags but the same public key. Cross-tag aggregation is inexpensive, requiring 0.0035 ms for one ciphertext and 1.65 ms for 512 ciphertexts. Cross-tag decryption, however, scales linearly with the number of distinct tags: it grows from 3.20 ms for one tag to 1628.70 ms for 512 tags. This is expected because the decryptor must process one tag-specific witness and one ciphertext component per distinct tag. Accordingly, cross-tag aggregation serves a different purpose from same-tag aggregation. Same-tag aggregation is preferable when all ciphertexts share a single release condition, as one witness enables constant-cost aggregate decryption. Cross-tag aggregation instead supports combining ciphertexts whose independent release conditions must be retained, such as values associated with different epochs,

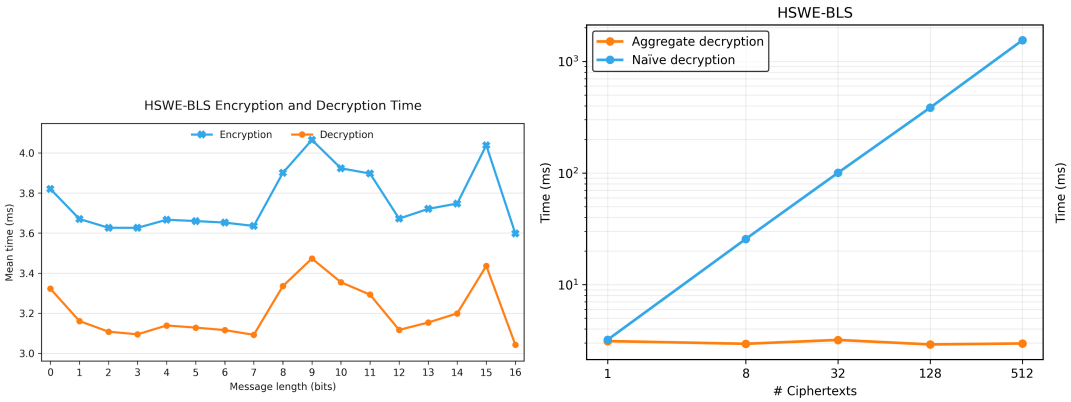
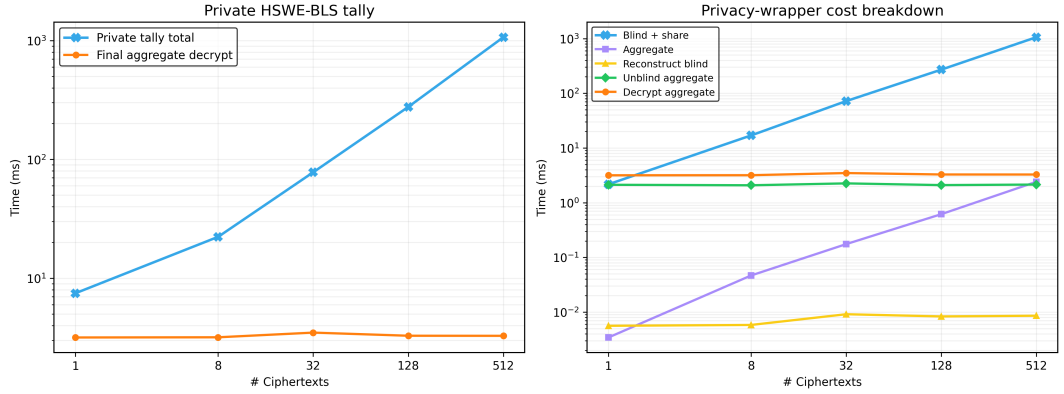


Figure 7: Performance of BLS-based HSWE scheme.



(a) Private aggregate-tally latency

(b) Stateful privacy-wrapper overhead

Figure 8: Performance of the privacy-preserving same-tag HSWE wrapper. Left: end-to-end private-tally latency and final aggregate-decryption time. Right: operation-level cost breakdown. Per-ciphertext blinding and Shamir-share generation dominate the total cost, while unblinding and final aggregate decryption remain approximately constant.

application phases, or externally defined policies. Our measurements show that the present cross-tag implementation has performance essentially equivalent to sequential decryption of the corresponding ciphertexts: at 512 tags, cross-tag decryption requires 1628.70 ms, compared with 1634.36 ms for sequential decryption. Its benefit is therefore not constant-cost decryption, but avoiding the quadratic number of pairing cross-terms required by a direct cross-tag cancellation strategy, yielding linear extraction complexity.

The results in Figure 10 show that aggregation and naïve per-ciphertext decryption scale approximately linearly with batch size in our lattice-based scheme from Section 4.1, increasing from tens of nanoseconds for a single ciphertext to around 15–20 microseconds for 512 ciphertexts. In contrast, aggregate decryption remains essentially constant at roughly 0.03 microseconds, since it operates on one aggregated ciphertext rather than processing every ciphertext individually. This demonstrates the main efficiency benefit of the scheme: after aggregation, the cost of recovering the final binary result is independent of the number of inputs. The primary motivation for lattice-based HSWE is not necessarily to reduce the local asymptotic cost of computing a one-bit aggregate. Rather, HSWE separates inexpensive, publicly executable evaluation from the privileged witness-enabled decryption step. This separation permits an untrusted party to aggregate ciphertexts incrementally while requiring only one final witness-enabled decryption, which is particularly valuable when witness generation or use is threshold-governed, interactive, costly, or subject to an on-chain release policy. Moreover, the lattice instantiation provides a post-quantum foundation and naturally extends to RLWE-based packing, where one ciphertext can encode many message slots and yield meaningful amortized throughput improvements.

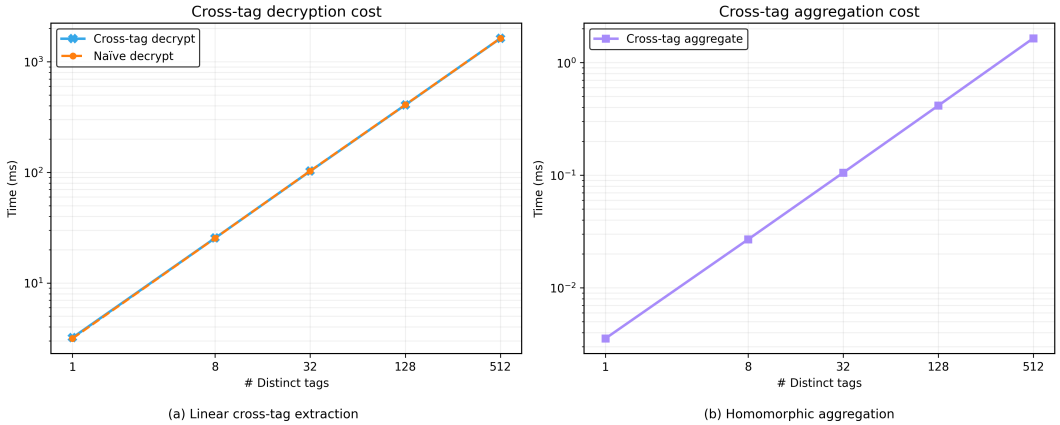


Figure 9: Cross-tag aggregation and decryption performance. Aggregation is inexpensive, while decryption grows linearly with the number of distinct tags. The cross-tag construction avoids quadratic cross-term cancellation.

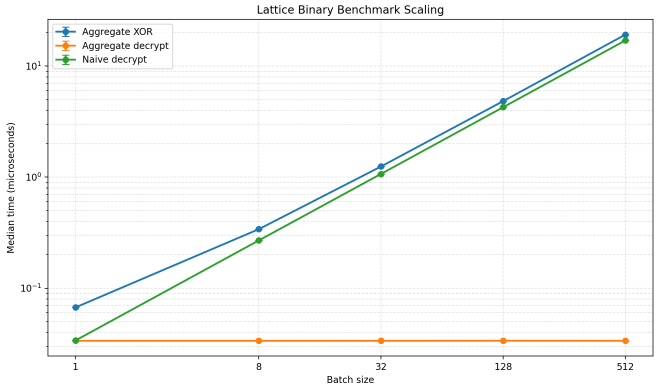


Figure 10: Performance of the lattice-based HSWE scheme.

8 Related Work

Signature-based witness encryption (SWE) enables the construction of encryption schemes directly from digital signature schemes, where a ciphertext is decrypted if and only if a valid witness (e.g., a signature on a specific public tag or epoch) is published. Because general-purpose witness encryption relies on heavy, often impractical cryptographic assumptions such as multilinear maps or indistinguishability obfuscation (iO) [GGSW13, BKP24], the community has recently shifted toward practical, special-purpose WE frameworks [GHK+25] and honest-majority blockchain committees to instantiate social or signature-based WE [SZ23]. Recent works like tlock [GMR23] and McFly [DHMW23] demonstrate how to successfully instantiate SWE using threshold BLS signatures. Their constructions heavily rely on the Boneh-Franklin Identity-Based Encryption (IBE) framework [BF01], where the identity acts as the public statement and the secret extraction key is the corresponding BLS signature on that identity's hash. However, decentralized approaches like McFly require ciphertexts to grow linearly with the committee size. While Avitabile et al. [ADM+24] recently addressed this size blowup, their solution remains primarily of theoretical interest due to its reliance on iO [KLW15]. Furthermore, these baseline SWE approaches inherently rely on an all-or-nothing decryption mechanism for a batch of ciphertexts under a given identity.

This all-or-nothing batch decryption is highly problematic for systems requiring individual ciphertext confidentiality, such as Maximum Extractable Value (MEV) protection [CGPP24, AFP24]. To address this, Choudhuri et al. [CGPP24] proposed a commitment-based witness encryption scheme where decryption requires a corresponding proof of opening, protecting the privacy of messages excluded from the decrypted batch. For private key delivery, Cerulli et al. introduced vetKD [CCN+23], enabling the verifiable transfer of BLS signature shares from validators to users, which complements threshold extraction mechanics. In contrast to these approaches, our privacy-preserving HSWE framework (*cf.* Section 6.1) takes a distinct architectural route: we utilize linear secret sharing to guarantee the privacy of all individual messages, and decrypt *only* the homomorphically aggregated

result. The concept of performing homomorphic operations over identity-based or witness-encrypted ciphertexts is genuinely subtle. While the theoretical notion of *Homomorphic Witness Encryption* (HWE) has been recently explored, existing literature relies on highly impractical building blocks. For instance, Ananth et al. [BKP24] demonstrated that HWE can be constructed, but doing so explicitly requires the use of Homomorphic Indistinguishability Obfuscation (HiO). Garg et al. [GKM+24] devised a protocol generating multi-receiver ciphertexts sharing some algebraic similarities with our $\mathsf{poly}(\lambda)$-sized pairing scheme, though their security model and motivation differ significantly. In this work, rather than relying on iO or obfuscation, we formalize HSWE through a highly practical scaling narrative. We systematically transition from binary message spaces (using Cocks IBE and standard LWE) to polynomial spaces (via BLS) and finally to exponential message spaces (using Ring-LWE). Finally, a known limitation of epoch-based SWE schemes is that storing all previous epoch signatures to decrypt historical ciphertexts incurs a linear storage cost, which prior works like TIRE [BMS22] mitigated using logarithmic-sized tree structures. While our strictly stateless HSWE schemes inherently require specific epoch signatures, our cross-tag homomorphic optimization (*cf.* Section 5.1) natively reduces the computational overhead of aggregating ciphertexts spanning multiple distinct historical epochs, requiring only $\mathcal{O}(k)$ pairings.

9 Conclusion

In this work, we defined the notion of *homomorphic* signature-based witness encryption (HSWE) that allows an encryptor to encrypt a message m towards a tag ρ and a verification key pk. Later, the ciphertext ct can only be decrypted if and only if a valid signature σ with respect to pk on the message ρ is available. Our HSWE schemes allow the computation of simple but useful functions of the plaintexts m. Specifically, they support linear homomorphisms that already turn out to be highly useful in various applications such as voting, sealed-bid auctions, or randomness beacons. We presented four practical HSWE schemes and proved their security assuming standard cryptographic assumptions.

Acknowledgements. István András Seres was supported by the Ministry of Culture and Innovation and the National Research, Development, and Innovation Office within the Quantum Information National Laboratory of Hungary (Grant No. 2022-2.1.1-NL-2022-00004).

References

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008. 3

[ADM+24] Gennaro Avitabile, Nico Döttling, Bernardo Magri, Christos Sakkas, and Stella Wohnig. Signature-based witness encryption with compact ciphertext. *Cryptology ePrint Archive*, 2024. 2, 22

[AFP24] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. *Cryptology ePrint Archive*, 2024. 22

[AGHV25] Adi Akavia, Craig Gentry, Shai Halevi, and Margarita Vald. Achievable cca2 relaxation for homomorphic encryption. *Journal of Cryptology*, 38(1):1–43, 2025. 5

[AW24] Kaya Alpturer and Matthew Weinberg. Optimal randao manipulation in ethereum. *Advances in Financial Technologies*, 2024. 4

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 2004. 10

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018. 2, 27

[BBCE24] Joseph Bonneau, Benedikt Bünz, Miranda Christ, and Yuval Efron. Good things come to those who wait: Dishonest-majority coin-flipping requires delay functions. *Cryptology ePrint Archive*, 2024. 4

[BCC+21] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, 1:1–136, 2021. 1

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001. 3, 4, 10, 14, 15, 22

[BFH+24] Alex Biryukov, Ben Fisch, Gottfried Herold, Dmitry Khovratovich, Gaëtan Leurent, María Naya-Plasencia, and Benjamin Wesolowski. Cryptanalysis of algebraic verifiable delay functions. In *Annual International Cryptology Conference*, pages 457–490. Springer, 2024. 28

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 345–356, 2016. 2, 27

[BGLM24] Daniel Benarroch, Bryan Gillespie, Ying Tong Lai, and Andrew Miller. Sok: Programmable privacy in distributed systems. *Cryptology ePrint Archive*, 2024. 2

[BK25] Renas Bacho and Alireza Kavousi. Sok: Dlog-based distributed key generation. *Cryptology ePrint Archive*, 2025. 17

[BKP24] Kaartik Bhushan, Venkata Koppula, and Manoj Prabhakaran. Homomorphic indistinguishability obfuscation and its applications. In *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 14:1–14:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. 22, 23

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001. 3, 4, 6, 15

[BMS22] Leemon Baird, Pratyay Mukherjee, and Rohit Sinha. i-tire: Incremental timed-release encryption or how to use timed-release encryption on blockchains? In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pages 235–248, 2022. 23

[Bol02] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *International Workshop on Public Key Cryptography*, pages 31–46. Springer, 2002. 6

[BS07] Daniel Bernstein and Jonathan Sorenson. Modular exponentiation via the explicit chinese remainder theorem. *Mathematics of Computation*, 76(257):443–454, 2007. 28

[CATB23] Kevin Choi, Arasu Arun, Nirvan Tyagi, and Joseph Bonneau. Bicorn: An optimistically efficient distributed randomness beacon. In *International Conference on Financial Cryptography and Data Security*, pages 235–251. Springer, 2023. 2, 27

[CCN21] Panagiotis Chatzigiannis, Konstantinos Chalkias, and Valeria Nikolaenko. Homomorphic decryption in blockchains via compressed discrete-log lookup tables. In *International Workshop on Data Privacy Management*, pages 328–339. Springer, 2021. 17

[CCN+23] Andrea Cerulli, Aisling Connolly, Gregory Neven, Franz-Stefan Preiss, and Victor Shoup. vetkeys: How a blockchain can keep many secrets. *Cryptology ePrint Archive*, 2023. 22

[CD17] Ignacio Cascudo and Bernardo David. Scrape: Scalable randomness attested by public entities. In *International Conference on Applied Cryptography and Network Security*, pages 537–556. Springer, 2017. 29

[CGPP24] Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. *Cryptology ePrint Archive*, 2024. 22

[CMB23] Kevin Choi, Aathira Manoj, and Joseph Bonneau. SoK: Distributed randomness beacons. Cryptology ePrint Archive, Paper 2023/728, 2023. 4

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding: 8th IMA International Conference Cirencester, UK, December 17–19, 2001 Proceedings 8*, pages 360–363. Springer, 2001. 3, 10, 11

[CP92] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Annual international cryptology conference*, pages 89–105. Springer, 1992. 29

[CP19] Bram Cohen and Krzysztof Pietrzak. The chia network blockchain. *White Paper, Chia. net*, 9, 2019. 2

[Des92] Yvo Desmedt. Threshold cryptosystems. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 1–14. Springer, 1992. 2

[DGS20] Samuel Dobson, Steven D Galbraith, and Benjamin Smith. Trustless groups of unknown order with hyperelliptic curves. *IACR Cryptol. ePrint Arch.*, 2020:196, 2020. 28

[DHMW23] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. Mcfly: verifiable encryption to the future made practical. In *International Conference on Financial Cryptography and Data Security*, pages 252–269. Springer, 2023. 1, 2, 3, 4, 16, 22

[DRA20] Team drand, drand project website. `https://drand.love`, 2020. 1

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th annual symposium on foundations of computer science (sfcs 1987)*, pages 427–438. IEEE, 1987. 29

[FHAS25] Nils Fleischhacker, Mathias Hall-Andersen, and Mark Simkin. Extractable witness encryption for kzg commitments and efficient laconic ot. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 423–453. Springer, 2025. 1

[FLOP18] Tore Kasper Frederiksen, Yehuda Lindell, Valery Osheter, and Benny Pinkas. Fast distributed rsa key generation for semi-honest and malicious adversaries. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*, pages 331–361. Springer, 2018. 28

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476, 2013. 22

[GHK+24] Jacob Gorman, Lucjan Hanzlik, Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Pratik Sarkar, and Sri AravindaKrishnan Thyagarajan. Vraas: Verifiable randomness as a service on blockchains. *Cryptology ePrint Archive*, 2024. 1

[GHK+25] Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru-Vamsi Policharla. A framework for witness encryption from linearly verifiable snarks and applications. In *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part III*, pages 504–539. Springer, 2025. Also available as IACR Cryptol. ePrint Arch. 2025:1364. 22

[GKM+24] Sanjam Garg, Aniket Kate, Pratyay Mukherjee, Rohit Sinha, and Sriram Sridhar. Insta-pok3r: Real-time poker on blockchain. *Cryptology ePrint Archive*, 2024. 23

[GKPW24] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In *Annual International Cryptology Conference*, pages 352–386. Springer, 2024. 9, 10

[GM19] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, pages 173–201. 2019. 7

[GMR23] Nicolas Gailly, Kelsey Melissaris, and Yolan Romailler. tlock: Practical timelock encryption from threshold bls. *Cryptology ePrint Archive*, 2023. 1, 3, 4, 22, 27

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008. 3, 4, 8, 10, 11, 13, 18

[GSZB23] Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. Cicada: A framework for private non-interactive on-chain auctions and voting. *Cryptology ePrint Archive*, 2023. 2, 3, 27

[Kal00] Burt Kaliski. Ieee standard specifications for public-key cryptography. ieee std 1363–2000, 2000. 7

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007. 9

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 419–428, 2015. 22

[KWJ24] Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. Sok: Public randomness. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 216–234, 2024. 4

[LaV16] Rio LaVigne. Simple homomorphisms of cocks ibe and applications. *Cryptology ePrint Archive*, 2016. 4, 10, 11

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer Berlin Heidelberg, 2010. 8

[LSS20] Esteban Landerreche, Marc Stevens, and Christian Schaffner. Non-interactive cryptographic timestamping based on verifiable delay functions. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, pages 541–558. Springer, 2020. 27

[MLQ23] Liam Medley, Angelique Faye Loe, and Elizabeth A Quaglia. Sok: Delay-based cryptography. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 169–183. IEEE, 2023. 27

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In *Annual International Cryptology Conference*, pages 620–649. Springer, 2019. 2, 3, 27

[NTSL25] Ábel Nagy, János Tapolcai, István András Seres, and Bence Ladóczki. Forking the randao: Manipulating ethereum's distributed randomness beacon. *Cryptology ePrint Archive*, 2025. 4

[Rab79] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. 1979. 3, 4, 7

[RG22] Mayank Raikwar and Danilo Gligoroski. Sok: Decentralized randomness beacon protocols. In *Australasian Conference on Information Security and Privacy*, pages 420–446. Springer, 2022. 4

[RS20] Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In *Annual International Cryptology Conference*, pages 481–509. Springer, 2020. 28

[RSS20] Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 155–180. Springer, 2020. 28

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996. 1, 2, 27

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. 7, 18

[SJH+21] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar Weippl. Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness. 2021. 2

[SZ23] Schwinn Saereesitthipitak and Dionysis Zindros. Cassiopeia: Practical on-chain witness encryption. In *Financial Cryptography and Data Security. FC 2023 International Workshops - CoDecFin, DeFi, Voting, and WTSC, Bol, Brač, Croatia, May 5, 2023, Revised Selected Papers*, volume 13968 of *Lecture Notes in Computer Science*, pages 385–404. Springer, 2023. 22

[TAF+23] Nirvan Tyagi, Arasu Arun, Cody Freitag, Riad Wahby, Joseph Bonneau, and David Mazières. Riggs: Decentralized sealed-bid auctions. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1227–1241, 2023. 2, 27

[TLN25] János Tapolcai, Bence Ladóczki, and Ábel Nagy. Slot a la carte: Centralization issues in ethereum's proof-of-stake protocol. *Cryptology ePrint Archive*, 2025. 4

[VOSS20] A Vlasov, K Olson, A Stokes, and A Sanso. eip-2537: Precompile for bls12-381 curve operations [draft]. *ethereum improvement proposals*, (2537), 2020. 17

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*, pages 379–407. Springer, 2019. 28

[WW20] Benjamin Wesolowski and Ryan Williams. Lower bounds for the depth of modular squaring. *Cryptology ePrint Archive*, 2020. 28

A Limitations of Timed-release Cryptography in the Dishonest Majority Setting

Timed-release cryptography is a fascinating branch of cryptography that allows integrating the notion of time into the typical security and privacy considerations such as confidentiality [MLQ23]. It fundamentally revolves on the principle that the computation should demand a minimum amount of wall clock time to complete. Timed-release cryptography can be instantiated from computational puzzles [RSW96] or obfuscation [BGJ+16] in the *dishonest majority* setting. On the other hand, it is also possible in the *honest majority* setting by assuming (a threshold of) trusted parties releasing certain piece of information (*e.g.*, signatures) at a specified time [RSW96, GMR23]. In recent years, we have seen significant interest in the former category spearheaded by works such as homomorphic time-lock puzzles [MT19], verifiable delay functions [BBBF18], and numerous derived applications including distributed randomness beacons [CATB23], auctions [TAF+23], voting [GSZB23], cryptographic time-stamping services [LSS20], and more. However, practical deployments of timed-release cryptography from computational puzzles are almost nonexistent due to the severe technical challenges that we highlight a few as follows.

Inherent sequentiality of delay functions. Repeated squaring is the most popular and well-understood candidate as the computational puzzle in timed-release cryptography. However, little is known about the sequentiality of repeated squaring. In RSA groups, it was shown that one cannot speed up *generically* repeated squaring unless factoring is easy [RS20]. Lower bounds for circuit depth computing modular squaring are known due to Wesolowski and Williams [WW20]. However, speeding up repeated squaring is possible, though at the expense of massive parallelism [BS07]. The situation in class groups is even more dire with little being known about the sequentiality of repeated squaring in these groups [Wes19]. Importantly, it is not clear how these theoretical results on circuit-depth lower bounds translate to lower bounds on practical running time that would be necessary for any real-world deployment. Recently proposed alternative delay functions based on root finding $\mod p$ have been successfully attacked in [BFH+24] and thus abandoned.

Instantiating the cryptographic group. *Generic* group delay functions require groups of unknown order [RSS20], *e.g.,* RSA or class groups of imaginary quadratic fields. Timed-release cryptography is further hindered by the challenges of efficiently and securely instantiating groups of unknown order. In the case of RSA groups, one typically needs to perform a multi-party distributed modulus generation protocol [FLOP18]. Securely implementing and running such a distributed modulus generation protocol for hundreds of parties turned out to be an extremely difficult and error-prone endeavour. In class groups, one can transparently generate a large random prime $\Delta = -p$ as a secure discriminant for the group $Cl(\Delta)$. However, class groups are considered inefficient as they must be instantiated in significantly larger groups than RSA groups for a given security parameter λ, *e.g.,* for $\lambda = 128$ one needs a 3072-bit semiprime as modulus, while a class group must be instantiated in a 6784-bit discriminant Δ [DGS20]. In resource-constrained environments, *e.g.,* blockchains, these parameters are considered highly inefficient.

Hardware assumptions. An implicit non-cryptographic assumption of every computational puzzle-based timed-release cryptography is the adversary's inability to compute the puzzle faster than the currently available best specialized hardware (*e.g.,* GPU, FPGA, ASIC) can do. Even minor speedups achieved in computing the puzzle can endanger the security of the applications built on the assumption that the adversary cannot speed up the puzzle computation. Moreover, algorithms exist that allow the parallelization of repeated squaring assuming numerous processors [BS07]. However, we note that this algorithm's practicality is still disputed.

B Formal Definitions for SWE

Definition 7 (Correctness)**.** The perfect correctness of SWE requires that for every pp output by Setup, and for every $m \in \mathcal{M}$, for every $\rho \in \mathcal{R}$, and for every signature σ on ρ for which $\Sigma.\mathsf{Verify}(\rho, \sigma, \mathsf{pk}) = 1$, the following holds:

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(m, \rho, \mathsf{pk}), \rho, \sigma) = m] = 1 \tag{B.1}$$

The indistinguishability security experiments for SWE resemble the well-known semantic security against adaptive chosen-plaintext and chosen-ciphertext attacks, namely $\mathsf{SWE} - \mathsf{CPA}$ and $\mathsf{SWE} - \mathsf{CCA}$. In the former, the adversary should not distinguish a ciphertext encrypting one of the two messages (of equal length) of its choice while only having (adaptive) oracle access to the signatures on arbitrary tag/identity, except for the one used for generating the challenge ciphertext. [8] In the latter, the adversary can further (adaptively) query the decryption oracle for any ciphertexts of its choice, except for the one corresponding to the challenge ciphertext. More formally, the indistinguishability security game is defined in Figure 11.

Definition 8 (SWE indistinguishability)**.** Indistinguishable security for a SWE scheme $\mathcal{E}_\Sigma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ requires that no PPT adversary $\mathcal{A}$ has more than a negligible advantage in the experiment $\mathsf{Exp}_{\mathsf{SWE}-\mathsf{IND}}$. Thus,

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{SWE}-\mathsf{IND}} := \Pr[\mathsf{Exp}_{\mathsf{SWE}-\mathsf{IND}}(\mathcal{A}, 1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \ . \tag{B.2}$$

C Privacy-Preserving HSWE with Verifiability

In our baseline privacy-preserving framework (*cf.* Section 6.1), we implicitly assume each user blinds their ciphertext correctly—namely, that the randomness s used to blind the ciphertext is exactly the same value that is

[8] In the standard experiment the adversary is challenged on a random public key instead of the one of its choice as here.

Experiment $\mathsf{Exp}_{\mathsf{SWE\text{-}IND}}$

$\underline{\mathsf{Exp}_{\mathsf{SWE\text{-}IND}}}$
$Q_{\mathsf{so}} := \emptyset, Q_{\mathsf{do}} := \emptyset$
$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}, \mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$
$\mathcal{A}^{\Sigma^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}}}(\mathsf{pp}, \mathsf{pk}) \to (\rho^*, m_0, m_1)$
$b \xleftarrow{\$} \{0,1\}$
$\mathsf{Enc}(m_b, \rho^*, \mathsf{pk}) \to \mathsf{ct}_b^{\rho^*}$
$\mathcal{A}^{\Sigma^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}}}(\mathsf{pp}, \mathsf{pk}, \mathsf{ct}_b^{\rho^*}) \to b'$
$b_0 := (b = b')$
$b_1 := (\rho^* \notin Q_{\mathsf{so}})$
$b_2 := (\{\rho^*, \mathsf{ct}_b^{\rho^*}\} \notin Q_{\mathsf{do}})$
return $b_0 \wedge b_1 \wedge b_2$

$\underline{\Sigma^{\mathcal{O}}(\rho)}$: Signing Oracle
$Q_{\mathsf{so}} := Q_{\mathsf{so}} \cup \{\rho\}$
$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho) \to \tilde{\sigma}$
return $\tilde{\sigma}$

$\underline{\mathsf{Decrypt}^{\mathcal{O}}(\rho, \mathsf{ct}^\rho)}$: Decryption Oracle
$Q_{\mathsf{do}} := Q_{\mathsf{do}} \cup \{\rho, \mathsf{ct}^\rho\}$
return $\mathsf{Dec}(\mathsf{ct}, \rho, \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho))$

Figure 11: Indistinguishability Security Experiment for SWE

secret-shared among the committee. If a malicious user deviates, the aggregated blinding factor will be incorrect, and the final decryption will fail. To ensure robustness, we require users to attach a proof demonstrating the well-formedness of their blinded ciphertexts.

To optimize this proof and avoid expensive operations in the target group $\mathbb{G}_T$ (as discussed in Section 5.2), the user specifically blinds the source-group ephemeral key U rather than the target-group component V. Given the baseline ciphertext $\mathsf{ct} = (U, V) = (g^r, g_T^m g_\rho{}^r)$, the blinded ciphertext is constructed as $\mathsf{ct}' = (U', V) = (h^{\mathsf{s}} g^r, g_T^m g_\rho{}^r)$, where $h \in \mathbb{G}$ is a secondary generator.

The encryptor utilizes verifiable secret sharing [Fel87] to commit to their shares $v_i = g_1^{s_i}$ and broadcasts the commitment vector $\mathbf{v} = \{v_i\}_{i \in [n]}$. The encryptor must prove that the shares are consistent with a degree-t polynomial using standard techniques [CD17]. To prove that the blinding factor embedded in U' matches the VSS commitment, we employ a Sigma protocol that proves simultaneous knowledge of two witnesses, where one witness is shared across two statements. This is a blinded variant of the Discrete Logarithm Equality (BDLEQ) proof, derived from Chaum and Pedersen [CP92]:

$$\mathcal{L}_{\mathsf{BDLEQ}} = \{(g_1, v, h, g, U') \mid \exists(\mathsf{s}, r) : v = g_1^{\mathsf{s}} \wedge U' = h^{\mathsf{s}} g^r\}$$

The interactive protocol proceeds as follows (and can be made non-interactive via the standard Fiat-Shamir heuristic):

1. **Commit:** The prover (user) chooses two random scalars $u_1, u_2 \xleftarrow{\$} \mathbb{F}_p$, computes $a_1 = g_1^{u_1}$ and $a_2 = h^{u_1} g^{u_2}$, and sends (a_1, a_2) to the verifier.
2. **Challenge:** The verifier sends back a uniformly random challenge $c \xleftarrow{\$} \mathbb{F}_p$.
3. **Respond:** The prover computes the masked responses $r_1 = u_1 + c\mathsf{s}$ and $r_2 = u_2 + cr$, and sends (r_1, r_2) to the verifier.
4. **Verify:** The verifier accepts the proof if and only if both $g_1^{r_1} = a_1 x^c$ and $h^{r_1} g^{r_2} = a_2 (U')^c$ hold.

By providing this proof, the aggregator and the signing committee are mathematically guaranteed that the secret-shared blinding factor perfectly corresponds to the mask applied to the ciphertext, ensuring that the final homomorphic unblinding step succeeds.

D Setup for Lattice-based HSWE

Setup and notation. Let encryption of a bit $b \in \{0,1\}$ under identity ρ be

$$c = (u, v) = (A^\top s + e_1 \ , \ \langle y, s \rangle + e_2 + \lfloor q/2 \rfloor b) \in \mathbb{Z}_q^m \times \mathbb{Z}_q,$$

with the secret $s \leftarrow \mathcal{D}_s \subset \mathbb{Z}_q^n$ (short), and error terms $e_1 \in \mathbb{Z}^m$ and $e_2 \in \mathbb{Z}$ drawn from subgaussian distributions of parameters σ_1, σ_2 respectively (e.g., discrete Gaussians). For the identity hash $y = H(\rho)$, the private key x satisfies $Ax \equiv y \pmod q$ and is sampled from a discrete Gaussian with parameter $\sigma_x > 0$ over the coset lattice. Thus, with probability $1 - \delta_x$, we have:

$$\|x\|_2 \leq \ B_x \ := \ \sigma_x(\sqrt{m} + \sqrt{2\ln(1/\delta_x)}).$$

Decryption computes

$$v - \langle u, x\rangle \equiv \lfloor q/2 \rfloor \, b + \eta \pmod q, \qquad \eta := e_2 - \langle e_1, x\rangle.$$

Correctness holds if $|\eta| < q/4$ (using the standard rounding interval).

Single-ciphertext tail bound. Conditioned on $\|x\|_2 \leq B_x$, the error η is subgaussian with parameter

$$\sigma_{\text{eff}} \ := \ \sqrt{\sigma_2^2 \ + \ \sigma_1^2 \, \|x\|_2^2} \ \leq \ \sqrt{\sigma_2^2 \ + \ \sigma_1^2 \, B_x^2}\,.$$

Hence, for any $t > 0$,

$$\Pr\left[\, |\eta| > t \mid \|x\|_2 \leq B_x \right] \ \leq \ 2\exp\Big(-\frac{t^2}{2\sigma_{\text{eff}}^2}\Big).$$

Aggregation of k ciphertexts (same identity). Let $c_i = (u_i, v_i)$ be encryptions of b_i, and define the componentwise sum $c_\oplus = \sum_{i=1}^k c_i = (\sum_i u_i, \ \sum_i v_i)$. Decryption error aggregates as

$$\eta_{\text{tot}} = \sum_{i=1}^{k} (e_{2,i} - \langle e_{1,i}, x\rangle).$$

Assuming independence across ciphertexts (standard in ROM/LWE analyses), η_{tot} is subgaussian with parameter

$$\sigma_{\text{tot}} \ = \ \sqrt{k}\,\sigma_{\text{eff}} \ \leq \ \sqrt{k}\,\sqrt{\sigma_2^2 + \sigma_1^2 B_x^2}\,.$$

Therefore, with failure probability at most δ_{dec},

$$|\eta_{\text{tot}}| \leq \ \sqrt{2\ln\Big(\frac{2}{\delta_{\text{dec}}}\Big)}\ \sigma_{\text{tot}} \ = \ \sqrt{2k\ln\Big(\frac{2}{\delta_{\text{dec}}}\Big)}\ \sqrt{\sigma_2^2 + \sigma_1^2 B_x^2}\,.$$

Correct decryption after summing k ciphertexts is guaranteed whenever

$$\boxed{\sqrt{2k\ln\Big(\frac{2}{\delta_{\text{dec}}}\Big)}\ \sqrt{\sigma_2^2 + \sigma_1^2 B_x^2} \ < \ \frac{q}{4}\,.}$$