Homomorphic Signature-based Witness Encryption and Applications

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

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

March 24, 2025

Abstract

Practical 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. Applications of SWE schemes include voting, auctions, distributed randomness beacons, and more. However, the lack of homomorphism in existing SWE schemes reduces efficiency and hinders deployment. In this work, we introduce the notion of *homomorphic* SWE (HSWE) to improve the practicality of timed-release encryption schemes. We show one can build HSWE using a pair of encryption and signature schemes where the uniqueness of the signature is required when the encryption scheme relies on injective one-way functions. We then build three HSWE schemes in various settings using BLS, RSA, and Rabin signatures and show how to achieve a privacy-preserving variant that only allows extracting the homomorphically aggregated result while keeping the individual plaintexts confidential.

1 Introduction

Timed-release cryptography [RSW96] can be instantiated from computational puzzles (*e.g.,* repeated modular squaring) or obfuscation [BGJ+16] in the *dishonest majority* setting. On the other hand, timed-release crypto is also possible in the *honest majority* setting by assuming (a threshold of) trusted parties that release certain information (*e.g.,* signatures, commitment openings, etc.) at a specified time. In recent years, we have seen significant interest and development in the former category spearheaded by works such as homomorphic time-lock puzzles [MT19], verifiable delay functions [BBBF18], and numerous derived applications: distributed randomness beacons [CATB23], auctions [TAF+23], voting schemes [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 hinder deployment. Next, we highlight some of these hurdles.

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 with repeated squaring in class groups is even more dire. Little is known about the sequentiality of repeated squaring in class 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 $\mathrm{mod}p$ have been successfully attacked in [BFH+24] and thus abandoned.

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

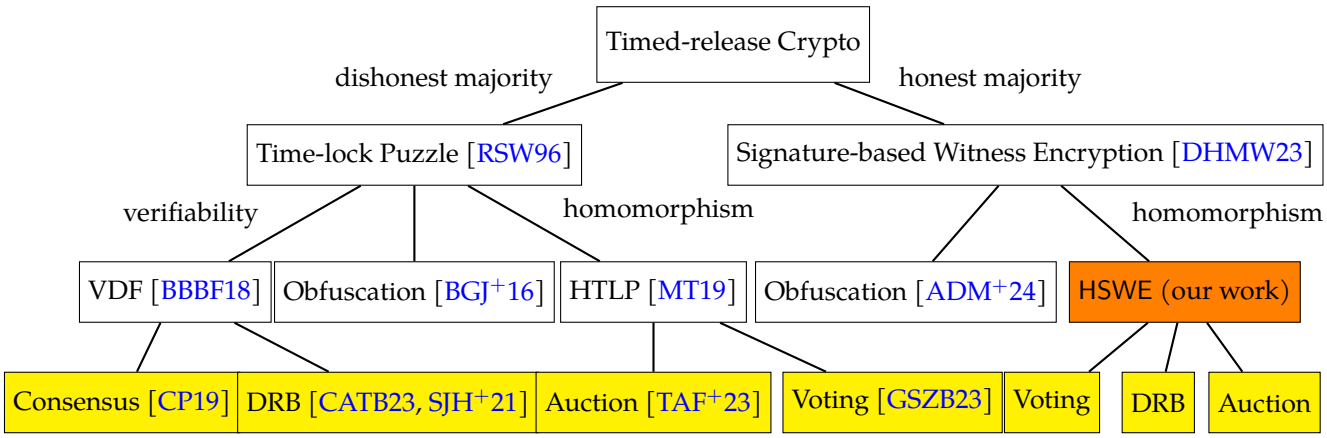

Figure 1: Timed-release cryptography [RSW96] from a bird's-eye view. Only a limited number of papers and schemes are shown due to space constraints. In this paper, we contribute to the honest-majority line of work by extending the applicability and practicability of signature-based witness encryption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

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 the case of 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.

Despite the numerous promising applications enabled by time-lock puzzles and related primitives, we are yet to see their widespread deployment and adoption in practice. We argue that the dishonest majority trust model is essentially overkill in many of the applications we have in mind. For instance, blockchain consensus typically already assumes an honest majority, *e.g.*, Nakamoto consensus [Nak08]. Thus, for the sake of efficiency, it makes sense to build and apply cryptographic primitives from the honest majority branch of timed-release cryptography, see Figure 1. Therefore, in this work, we turn our attention to timed-release cryptography in the honest majority setting.

1.1 Timed-release Cryptography in the Honest Majority Setting

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. Examples include 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 randomness services for randomness generation are the Chainlink VRF [BCC+21] or the Supra dVRF service [GHK+24]. This work focuses on timed-release cryptography

built on signature-based services. One can build 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 on ρ under a (distributed) public key pk. This functionality is enabled by a specific case of witness encryption.

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 [GGSW13]. In general, practical WE scheme for *all NP* seems out of reach, *i.e.,* existing schemes are highly impractical and inefficient. However, there exist several witness encryption schemes that support important NP languages, *e.g.,* [FHAS25]. As a practical 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 [GMR23, DHMW23] (interchangeably referred to as signature-based witness encryption) have seen a resurgence in applied cryptography partly due to the difficulty of deploying time-based cryptography using computational puzzles. SWE foregoes the delicate cryptographic assumptions of timed-release crypto built on computational puzzles and builds efficient timed-release cryptography assuming *threshold trust*. Specifically, the threshold SWE assumes that k participants of a quorum of n parties release a signature σ on a tag ρ (specifying a predefined future date) under some verification key pk.

SWE schemes have significant advantages over common threshold encryption schemes [Des92], making them an excellent choice for use in blockchain settings with dynamic participation. In particular, SWE does not necessarily require an expensive setup phase for the decryption committee, as the ciphertext is generated only with respect to a tag and some verification key pk. This verification key pk could be either a common public key (secret-shared in the threshold setting) or even a set of individual verification keys $\{\mathsf{pk}_i\}_{i=1}^n$ for better liveness. However, the downside of the latter is that the size of the ciphertext depends on the number of verification keys and thus grows linearly [ADM+24]. Moreover, unlike threshold encryption, SWE allows *stateless* decryption, *i.e.,* the decryption can occur independently from and with no prior knowledge about the ciphertexts, as it only requires a signature on some publicly known tag. These properties make SWE a promising 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. SWE schemes hold significant promise for various on-chain applications such as 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.

1.2 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. By incorporating homomorphism, the goal is to significantly boost performance. Existing popular SWE schemes, such as 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 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 (HTLP) [MT19], though without suffering from the inherent issues relevant to time-based cryptography as already outlined.

In a nutshell, we make the following contributions in this paper.

Formal definition for HSWE. We present a formal definition for HSWE and provide its security properties. In particular, we prove that while using a pair of encryption and digital signature schemes with the encryption scheme's key generation algorithm built from an injective one-way function, the accompanying signature should be unique, *i.e.,* there is only one valid signature σ for each pair of (m, pk). The most notable unique signature schemes are BLS and RSA.

HSWE for one-bit messages. We propose a CPA-secure HSWE scheme for single-bit messages (*i.e.,* useful in coin-flipping applications) based on Cocks IBE scheme [Coc01], and Rabin's signature scheme [Rab79].

CPA-secure HSWE. We propose a CPA-secure HSWE scheme based on Boneh-Franklin IBE scheme [BF01] and BLS signature scheme [BLS01]. This scheme has a $\mathsf{poly}(\lambda)$-sized message space and allows for homomorphism

both under common and across distinct tags.

Stateful HSWE**.** We propose a CPA-secure HSWE scheme for exponentially-sized message spaces using a pair of modified Paillier encryption [Pai99], and RSA signature scheme [RSA78]. This scheme is stateful in the sense that the decryption (*i.e.,* the message the trusted parties sign) depends on the set of aggregated ciphertexts.

Privacy-preserving HSWE**.** We show how to build a privacy-preserving HSWE that preserves the confidentiality of individual plaintexts while still offering homomorphism for the corresponding ciphertexts.

1.3 Applications of HSWE Schemes

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

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.

Auctions. In auctions, similarly to voting schemes, user's 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].

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 elaborate on the cryptographic and engineering details of such (on-chain) DRB protocol in a future work.

MEV protection. A random permutation on the block of encrypted transactions turns out to be necessary to protect against the type of maximal extractable value (MEV) attacks that do not depend on the content of transactions [RSKK, KLJD23]. An illustrative example is arbitrage, allowing block proposer to benefit from MEV by taking the top of the block spots. Our HSWE scheme allows generating uniform randomness (*i.e.,* involving at least one honest contribution) with low on-chain communication (*i.e.,* a single signature) and computation (*i.e.,* a single pairing) overheads.

1.4 Technical Overview

A design principle of all current SWE schemes is the following recipe: consider an identity-based encryption scheme (IBE) whose decryption key "is" a valid signature produced by a suitable signature algorithm. Both [DHMW23, GMR23] follow this SWE design paradigm building on the Boneh-Franklin IBE scheme [BF01] and the BLS signature scheme [BLS01]. We follow the same design principle for our HSWE schemes in Section 5. Additionally, we also build an HSWE scheme from the Cocks IBE scheme and the Rabin signature scheme in Section 4.

We extend the aforementioned recipe by considering more than just efficient IBE schemes as the constituent encryption algorithm in our HSWE schemes. In Section 6, we show that one can build HSWE from a Paillier-like encryption scheme, indicating that it is also possible to build SWE schemes from weaker public-key encryption schemes.

1.4.1 HSWE from Cocks IBE and Rabin signature schemes

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

1.4.2 HSWE from Boneh-Franklin IBE and BLS signature schemes

Due to the inherent conflict between homomorphism and the strongest notion of CCA security [AGHV25], the best we can hope for is an efficient CPA-secure HSWE scheme. To do so, we modify the Boneh-Franklin IBE scheme by lifting the messages to the exponent. This allows homomorphic addition of the ciphertext at the expense of limiting the message space to $\mathsf{poly}(\lambda)$-size as the decryptor needs to solve a small discrete logarithm instance. While we consider homomorphism merely under a common tag, we observe that one can homophonically aggregate different ciphertexts under *distinct* tags using this scheme which could be of independent interest. We prove the security of our scheme under the hardness of Decisional Bilinear Diffie-Hellman assumption.

1.4.3 A stateful HSWE from modified Paillier encryption and RSA signature schemes

Our previous schemes support a limited message space, *i.e.*, either $|\mathcal{M}| = 2$ or $|\mathcal{M}| = \mathsf{poly}(\lambda)$. Here, we present a scheme that supports homomorphism for an exponentially large message space, that is, $\mathbb{Z}_N$. The caveat is that the signing parties must be *stateful*: they need to sign a message for decryption that is not independent of the ciphertexts. However, this implies a form of *conditional* decryption of (aggregated) ciphertexts with the benefit of a dynamic decision on the set of ciphertexts to be (homomorphically) decrypted. This is useful in applications where the decision about which ciphertexts are decrypted is non-monotone, such as block building in blockchains [AFP24]. It is a fascinating open problem to devise an efficient scheme that is both *stateless and homomorphic* for a large message space. The high-level technical idea of our stateful scheme is as follows. As we keep the messages in the exponent, we need a subgroup where the discrete logarithm is easy. Observe that working in $\mathrm{mod} N^2$ allows achieving homomorphism for the entire $\mathbb{Z}_N$, since it is easy to compute the discrete logarithm of $(1+N)^m \bmod N^2$ due to the binomial theorem, *i.e.*, $(1+N)^m \bmod N^2 = 1+Nm \bmod N^2$. Hence, we choose to work in $\mathbb{Z}_{N^2}$, and make use of a pair of modified Paillier encryption [Pai99] and RSA signature [RSA78] schemes. Our stateful HSWE ciphertext has the form of $\mathsf{ct} = (U, V) = ((1+N)^m \mathsf{H}(\rho)^{Nr\rho}, \mathsf{H}(\rho)^{Ner\rho})$, where $\mathsf{H}(\cdot)$ is a cryptographic hash function modeled as random oracle.

1.4.4 Privacy-preserving HSWE

An underlying aspect of all the previous HSWE constructions is that they allow the decryption of every ciphertext ct_i once the signature σ has been released. In some applications (*e.g.*, voting, auctions), individual user submissions (*i.e.*, ballots, bids) must remain hidden even after the voting/bidding period ends. The idea of our privacy-preserving variant is to randomize each user i's ciphertext ct_i with some randomness s_i. The blinding factor s_i is secret shared with the signing parties. At the decryption time, the signing parties also reconstruct the sum of the corresponding users' blinding factor s_i, *i.e.*, they obtain $\sum s_i$ to decrypt the aggregated ciphertext. Since no individual s_i is reconstructed, user ciphertexts remain encrypted. This technique is generic and renders the resulting HSWE scheme stateful. In Section 6.1, we describe the full details of our privacy-preserving HSWE scheme.

Organization. The remainder of this paper is organized as follows. In Section 2 we recall the pertinent background knowledge. In Section 3, we formally define homomorphic SWE schemes. In Section 4, we introduce an HSWE scheme for one-bit messages. In Section 5, we introduce our HSWE scheme for small message spaces. Section 6, introduces our stateful HSWE schemes. We analyse the performance of our schemes in Section 7. We review the related works in Section 8. Finally, we conclude our work with open questions and future directions in Section 9.

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 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 computational assumptions that our constructions and their applications rely on for their security.

co-Diffie-Hellman assumption. A bilinear group $\mathsf{bg} = (e, p, \mathbb{G}, \mathbb{G}_T, g, g_T)$ with mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies the co-Diffie-Hellman (co-CDH) assumption if for all PPT adversaries $\mathcal{A}$, it holds

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

Gap Diffie-Hellman assumption. A bilinear group $\mathsf{bg} = (e, p, \mathbb{G}, \mathbb{G}_T, g, g_T)$ with mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies the Gap Diffie-Hellman (GDH) assumption if for all PPT adversaries $\mathcal{A}$, it holds

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

In other words, the GDH assumption is the extension of the DDH assumption to bilinear groups requiring its intractability in the group $\mathbb{G}$.

Bilinear Diffie-Hellman assumption. A bilinear group $\mathsf{bg} = (e, p, \mathbb{G}, \mathbb{G}_T, g, g_T)$ with mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies the Bilinear Diffie-Hellman (BDH) assumption if for all PPT adversaries $\mathcal{A}$, it holds

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

It is helpful to note that the BDH assumption is an extension of the CDH assumption to bilinear groups, and its hardness implies the hardness of the Diffie-Hellman (DH) assumption in both $\mathbb{G}$ and $\mathbb{G}_T$. First, if the DHP in $\mathbb{G}$ can be efficiently solved, then one could solve an instance of the BDH by computing g^{ab} and then $e(g^{\alpha\beta}, g^\gamma) = e(g, g)^{\alpha\beta\gamma}$. Also, if the DH in $\mathbb{G}_T$ can be efficiently solved, then the BDH instance could be solved by having $g_T^{\alpha\beta} = e(g^\alpha, g^\beta), g_T^\gamma = e(g, g^\gamma)$, and then $g_T^{\alpha\beta\gamma}$.

Decisional Bilinear Diffie-Hellman assumption. A bilinear group $\mathsf{bg} = (e, p, \mathbb{G}, \mathbb{G}_T, g, g_T)$ with the bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies the Decisional Bilinear Diffie-Hellman (DBDH) assumption if for all PPT adversaries $\mathcal{A}$, it holds

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

2.4 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 $\mathsf{Share}(\mathsf{s}) \to \{s_1, \ldots, s_n\}$, 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 $l_j(x) = \prod_{\substack{1 \leq m \leq t+1 \\ m \neq j}} \frac{x - \chi_m}{\chi_j - \chi_m}$ allow the reconstruction of the secret as $s = f(0) = \sum_{j=1}^{t+1} s'_{\chi_j} l_j(0)$.

2.5 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 distribted 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.

2.6 Number Theory and 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:

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

Definition 2 (RSA Assumption [RSA78])**.** Informally, the RSA assumption states that no efficient adversary can compute the eth roots of a random group element g. It holds for a group of unknown order with generator algorithm $\mathrm{GGen}(\cdot)$ if for any probabilistic polynomial time adversary $\mathcal{A}$ there exists $\mathsf{negl}(\cdot)$ such that:

$$\Pr\left[u^e = g : \begin{array}{c} \mathbb{G} \xleftarrow{\$} \mathrm{GGen}(\lambda) \\ (g, e) \xleftarrow{\$} \mathbb{G} \times (\mathbb{N} \setminus \{1\}) \\ u \leftarrow \mathcal{A}(\mathbb{G}, g, e) \end{array} \right] \leq \mathsf{negl}(\lambda) \ . \tag{2.1}$$

In other words, computing eth roots in groups of unknown order, *e.g.*, $\mathbb{Z}_N^*$, is hard. It is well-known that if the factoring assumption is easy, then so is the RSA assumption. It is an open problem, however, if the RSA assumption is equivalent to the factoring assumption.

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 3 (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:

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

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

Definition 4 (Decisional Composite Residuosity)**.** Let N be an RSA modulus. Then, for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that:

$$\Pr\left[\begin{array}{c} x \xleftarrow{\$} \mathbb{Z}_N^*; b \xleftarrow{\$} \{0,1\} \\ b \leftarrow \mathcal{A}(N, y): \quad \text{if } b = 0 \text{ then } y \xleftarrow{\$} \mathbb{Z}_{N^2}^* \\ \text{if } b = 1 \text{ then } y := x^N \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda). \tag{2.2}$$

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 5 (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 $0 \leq b < n$. 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[1] 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 $(n, 0)$.

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 argue about the necessity of having unique signatures for an SWE scheme.

3.1 Definition

Definition 6 (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.

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

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

$\mathsf{Dec}(\mathsf{ct}, f(\rho), \sigma) \to m$. Given the ciphertext ct, the tag $f(\rho)$, 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.

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 $f(\rho)$ for which $\Sigma.\mathsf{Verify}(f(\rho), \sigma, \mathsf{pk}) = 1$, the following holds:

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(m, f(\rho), \mathsf{pk}), f(\rho), \sigma) = m] = 1 \ . \tag{3.1}$$

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. [2] In the latter, the adversary further can (adaptively) query 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 2.

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

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{SWE}-\mathsf{IND}} := \Pr[\mathsf{Exp}_{\mathsf{SWE}-\mathsf{IND}}(\mathcal{A}, 1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \ . \tag{3.2}$$

Definition 9 (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 6.

$\mathsf{Eval}_{\mathsf{pk}, f(\rho)}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to \mathsf{ct}$. 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 $f(\rho)$.

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 10 (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:

$$\Pr[\mathsf{Dec}(\mathsf{Eval}_{\mathsf{pk}, f(\rho)}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to \mathsf{ct}) \neq C(m_1, \ldots, m_n)] \leq \mathsf{negl}(\lambda) \ . \tag{3.3}$$

3.2 On the Necessity of 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 a one-way function, but they 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.

Theorem 1 (Unique signatures are necessary)**.** 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. □

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

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

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

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

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

Figure 2: Indistinguishability Security Experiment for SWE

The previous theorem 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 observe that it is enough that the public-key encryption scheme used in the SWE key generation algorithm be an injective function.

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

Proof. The public encryption key of an $\mathcal{E}_\Sigma$ is $f(\rho)$ for a roud number ρ and a function $f(\cdot)$.[3] The corresponding decryption secret key is a valid signature σ on ρ. If there were multiple valid signatures σ_0, σ_1 on ρ, then due to the injectivity of the encryption schemes key generation algorithm, there were different encryption public keys ρ_0, ρ_1 corresponding to the decryption keys σ_0, σ_1. This means that one of the decryptions would fail, violating the correctness of the SWE scheme. □

Remark 1. Note that Theorem 2 is in line with the result of [GKPW24]. Garg et al. show that SWE schemes exist for any signature scheme whose verification is a public linear constraint system; that is, verification equation of the signature scheme is linear in the signature σ. Theorem 2 implies that there cannot be SWE schemes with an injective key generation algorithm for any signature scheme with a higher-degree polynomial verification equation. This is because, in that case, there would be multiple valid signatures for a round number ρ, which is ruled out per Theorem 2.

4 HSWE for One-bit Messages

First, as a feasibility result, we introduce an HSWE scheme for one-bit messages, *i.e.,* $m \in \{\pm 1\}$ supporting XOR-homomorphism. Such an HSWE scheme is useful in coin-flipping protocols. Our scheme is based on the Cocks IBE scheme [Coc01] that can be made homomorphic by a technique developed in [LaV16].

4.1 Correctness and Homomorphism

Recall $r^2 = H(\rho, u)$, *i.e.,* r is a Rabin signature on the message ρ. Moreover,

$$c + 2r = t + \frac{H(\rho, u)}{t} + 2r = t(1 + 2rt^{-1} + H(\rho, u)t^{-2}) = t(1 + rt^{-1})^2 \ . \tag{4.1}$$

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

HSWE **with one-bit message space**

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

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho) \to \pi_\rho = r$ such that $r^2 = H(\rho, u) \mod N$ // Note that r is a Rabin signature on the public tag ρ. Furthermore, u could be published in advance accompanied with ρ.

Encryption: $\mathsf{Enc}(m, \rho, N) \to \mathsf{ct} := (c_1, c_2)$.

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

Return $\mathsf{ct} = c$.

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

1. Parse the ciphertext ct as c.
2. Let $r := \pi_\rho$. // Recall r is a Rabin signature on the public tag ρ satisfying $r^2 = H(\rho, u) \bmod N$.
3. Compute $m = \left(\frac{c + 2r}{N}\right)$.

Return: m.

Figure 3: A CPA-secure HSWE scheme with one-bit message space.

Correctness of the HSWE scheme now follows from Equation (4.1) since:

$$\left(\frac{c + 2r}{N}\right) = \left(\frac{t(1 + rt^{-1})^2}{N}\right) = \left(\frac{t}{N}\right) = m \ . \tag{4.2}$$

Recall that our ciphertext, *cf.* Figure 3, is a Cocks IBE ciphertext for which simple homomorphism exists. For technical details, we refer to [LaV16]. Note that at the expense of doubling the ciphertext size, we could omit u from the public parameters. Specifically, the requirement that $\left(\frac{H(\rho,u)}{N}\right) = 1$ enables us to have a single ciphertext element, although, the signing parties need to publish u as well to every ρ, since without the factorization of n no party can compute quadratic residuosity. If this increase in the public parameters is unwanted, then it could be traded off by doubling the ciphertext size as is classically done in the Cocks IBE scheme. In our applications, we are interested in reducing the ciphertext sizes.

4.2 Security Claims

Theorem 3. The homomorphic SWE scheme in Figure 3 is CPA-secure in the random oracle model, assuming the quadratic residuosity assumption (*cf.* Definition 3).

Proof. $\mathcal{A}$'s challenge plaintexts are $m_0, m_1 \in \{0, 1\}$. Let c^* be the challenge ciphertext encrypting m_b ($b \in_R \{0, 1\}$), *i.e.*, $\left(\frac{c^* + 2r}{N}\right) = m_b$. A straightforward reduction from the quadratic residuosity assumption shows that the adversary cannot distinguish between the encryptions of 0 and 1. For the formal proof, we refer to [Coc01]. □

Remark 2. We highlight that the Rabin signature scheme is not a unique signature, as each quadratic residue $\bmod n (= pq)$ has 4 square roots implying that each message might have 4 different valid Rabin signatures. Thus, we note that our construction in Figure 3 is not captured by the Theorems in Section 3.2. In other words, it is possible to design SWE schemes for non-unique signature schemes, if they are paired with a suitable encryption scheme.

A CPA-secure HSWE with small message space

Public parameters: $\mathbb{G}, \mathbb{G}_T$ elliptic curve groups such that $|\mathbb{G}| = |\mathbb{G}_T| = p$ for some large prime p. $\mathbb{G}$ is the source group and $\mathbb{G}_T$ is the target group of $e(\cdot,\cdot) : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, an efficiently computable, non-degenerate, bilinear pairing. Let $g \in \mathbb{G}, g_T, h_T \in \mathbb{G}_T$ be generators of the respective groups. A cryptographically secure hash-to-curve function is denoted as $\mathsf{H} : \{0,1\}^\lambda \to \mathbb{G}$. Let $P = g^s \in \mathbb{G}$ be a BLS public key, whose secret key is $s \in_R \mathbb{F}_p$. We denote ρ as a public tag (e.g., epoch number or timestamp).

Encryption: $\mathsf{Enc}(m, \rho, P) \to \mathsf{ct} := (U, V)$.

1. Compute an ephemeral key $U = g^r$ for some $r \in_R \mathbb{F}_p$.
2. Compute $g_\rho = e(\mathsf{H}(\rho), P)$
3. Set $c = (g^r, g_T^m {g_\rho}^r) := (U, V)$

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

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho) \to \pi_\rho := \mathsf{H}(\rho)^s$. // This is a BLS signature used to extract the key-pair for ρ with respect to the public key P.

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

1. Parse the ciphertext ct as (U, V).
2. Compute $g_T^m = V e(\pi_\rho, U)^{-1}$
3. Solve the discrete logarithm problem in $\mathbb{G}_T$ for $Z = g_T^m$ to obtain m.

Return: m.

Figure 4: A CPA-secure HSWE scheme with poly(λ)-sized message space.

5 HSWE for Small Message Spaces

5.1 A CPA-secure HSWE Scheme

In this section, we present a CPA-secure HSWE scheme for a poly(λ)-sized message space. We adapt the design of the Boneh-Franklin IBE [BF01] to make it homomorphic. To do so, we work with the message in the exponent. Thus, it is required to solve a discrete logarithm instance for decryption. The full scheme is described in Figure 4.

The correctness of the scheme in Figure 4 is easy to follow. Now, we argue that this scheme is additively homomorphic. Let $\mathsf{ct}_1, \mathsf{ct}_2$ be two distinct ciphertexts:

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

By component-wise aggregating the ciphertexts, we obtain:

$$\mathsf{ct}^* = (g^{r_1+r_2}, g_T^{m_1+m_2} g_\rho^{r_1+r_2}) := (U^*, V^*) \tag{5.1}$$

Then, due to the bilinearity of the pairing map, we correctly obtain the homomorphically added plaintext:

$$g_T^{m_1+m_2} = V^* e(\pi_\rho, U^*)^{-1} = V^* e(\mathsf{H}(\rho), P)^{-(r_1+r_2)} = V^* g_\rho^{-(r_1+r_2)} \ .$$

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

Proof. Assuming the existence of a PPT adversary $\mathcal{A}$ that breaks the CPA-security of our HSWE scheme, we construct a PPT adversary $\mathcal{B}$ that breaks the DBDH assumption. That is, $\mathcal{B}$ internally runs $\mathcal{A}$ and acts as a relay between it and the DBDH challenger to use its advantage in the indistinguishability game $\mathsf{Exp}_{\mathsf{SWE\text{-}IND}}$ to succeed in its own

DBDH game. The proof is implied, given the contradiction. Before we proceed with the proof details, we borrow an argument from Boneh-Franklin IBE scheme (Lemma 4.2, [BF01]) due to the similarity of our designs. That is, querying the signing oracle on arbitrary tags/identities does not help the adversary in its game in comparison to getting challenged in an (identical) public key encryption scheme without such privilege. More formally, given a IND-CPA adversary $\mathcal{A}_1$ with advantage $\epsilon(\lambda)$ that makes at most q_s queries to the signing oracle, there is a IND-CPA adversary $\mathcal{A}_2$ that has an advantage at least $\frac{\epsilon(\lambda)}{e(1+q_s)}$, where e is base of the natural logarithm. We rely on this argument in our security proof and refer the reader to [BF01] for further details.

Suppose there is a PPT adversary $\mathcal{A}$ that wins the CPA-security game of $\mathsf{Exp}_{\mathsf{SWE-IND}}$ with a (non-negligible) advantage $\epsilon(\lambda)$. Then, we show there is a PPT adversary $\mathcal{B}$ that wins the Decisional Bilinear Diffie-Hellman (DBDH) game with a relevant advantage.

- The DBDH challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda)$ and sends the bilinear pairing parameters $\mathsf{bg} = (e, p, \mathbb{G}, \mathbb{G}_T, g, g_T)$ to DBDH adversary $\mathcal{B}$ and it further forwards them to the CPA adversary $\mathcal{A}$ for initialization.
- In the first phase, $\mathcal{A}$ can make as many queries as it would like to get their corresponding ciphertexts. Let's denote them by $m_1, \ldots, m_t$, where t is upper bounded by some polynomial $q(\lambda)$.
- $\mathcal{C}$ randomly samples $\gamma \xleftarrow{\$} \mathbb{F}_p$. For each query $i \in [t]$, it further samples α, β from $\mathbb{F}_p$ and gives back to $\mathcal{B}$ a tuple $(g^\alpha, g^\beta, g^\gamma, T)$ where T is chosen according to its random sampling $b \xleftarrow{\$} \{0,1\}$. If $b = 0$ then $T = e(g^\beta, g^\gamma)^\alpha = e(g,g)^{\alpha\beta\gamma}$, otherwise $T \xleftarrow{\$} \mathbb{G}_T$.
- $\mathcal{B}$ then creates the queried ciphertexts as $\mathsf{ct} = (U, V) = (g^\alpha, g_T^{m_i} T)$ and sends it to the CPA adversary $\mathcal{A}$.
- $\mathcal{A}$ generates a challenge plaintext pair (m_0, m_1) where $|m_0| = |m_1|$ and send them to $\mathcal{B}$ to generate the corresponding challenge ciphertext. To do so, $\mathcal{B}$ randomly samples $b_r \xleftarrow{\$} \{0,1\}$ and sends a query to $\mathcal{C}$. It then generates the challenge ciphertext as $\mathsf{ct}^\star = (g^{\alpha^\star}, g_T^{m_{b_r}} T)$ and sends it to the CPA adversary $\mathcal{A}$.
- In the second phase, $\mathcal{A}$ can make further encryption queries conditioned on the total number of queries is upper bounded by $q(\lambda)$. Then, it outputs the guess $\tilde{b}$ regarding if the challenge ciphertext $\mathsf{ct}^\star$ encrypts m_0 or m_1.
- If $\tilde{b} = b_r$, then the DBDH adversary $\mathcal{B}$ outputs $\hat{b} = 0$ in its own game and else it outputs $\hat{b} = 1$.

Now, we need to analyze the winning probability of the DBDH adversary $\mathcal{B}$ in its own game that is amount to $\Pr[\hat{b} = b]$.

$$\begin{aligned} \Pr[\hat{b} = b] &= \Pr[\hat{b} = b | b = 0] \Pr[b = 0] + \Pr[\hat{b} = b | b = 1] \Pr[b = 1] \\ &= \Pr[\mathsf{Exp}^{\pi,\mathcal{A}}_{\mathsf{SWE-IND}} = 1] 1/2 + \Pr[b' = b | b = 1] 1/2 \end{aligned} \tag{5.2}$$

Note that $\Pr[b' = b | b = 1]$ is equivalent to the probability that the CPA adversary $\mathcal{A}$ fails in its own game when the challenger $\mathcal{C}$ picks a random group element in $\mathbb{G}_T$. Let denote the corresponding (encryption) scheme by π'.[4] So, we have

$$\begin{aligned} \Pr[\hat{b} = b] &= \Pr[\mathsf{Exp}^{\pi,\mathcal{A}}_{\mathsf{SWE-IND}} = 1] 1/2 + \Pr[\mathsf{Exp}^{\pi',\mathcal{A}}_{\mathsf{SWE-IND}} = 0] 1/2 \\ &= \Pr[\mathsf{Exp}^{\pi,\mathcal{A}}_{\mathsf{SWE-IND}} = 1] 1/2 + (1 - \Pr[\mathsf{Exp}^{\pi',\mathcal{A}}_{\mathsf{SWE-IND}} = 1]) 1/2 \\ &= 1/2 + 1/2(\Pr[\mathsf{Exp}^{\pi,\mathcal{A}}_{\mathsf{SWE-IND}} = 1] - \Pr[\mathsf{Exp}^{\pi',\mathcal{A}}_{\mathsf{SWE-IND}} = 1]) \end{aligned} \tag{5.3}$$

Let repeat be an event where the randomness α^* used in challenge ciphertext ct^* appears more than once in encryption queries. Then, we have

[4] This is obviously not a proper encryption algorithm as there is no corresponding decryption. We just use it for the purpose of establishing the proof.

$$
\begin{aligned}
\Pr[\hat{b}=b] &= 1/2 + 1/2(\Pr[\mathsf{Exp}^{\pi,\mathcal{A}}_{\mathsf{SWE-IND}}=1] - (\Pr[\mathsf{Exp}^{\pi',\mathcal{A}}_{\mathsf{SWE-IND}}=1|\mathsf{repeat}]\Pr[\mathsf{repeat}] \\
&\quad + \Pr[\mathsf{Exp}^{\pi',\mathcal{A}}_{\mathsf{SWE-IND}}=1|\bar{\mathsf{repeat}}]\Pr[\bar{\mathsf{repeat}}]) \\
&\geq 1/2 + 1/2(\Pr[\mathsf{Exp}^{\pi,\mathcal{A}}_{\mathsf{SWE-IND}}=1] - (\Pr[\mathsf{repeat}] + \Pr[\mathsf{Exp}^{\pi',\mathcal{A}}_{\mathsf{SWE-IND}}=1|\bar{\mathsf{repeat}}])) \\
&\geq 1/2 + 1/2(\Pr[\mathsf{Exp}^{\pi,\mathcal{A}}_{\mathsf{SWE-IND}}=1] - (\frac{q(\lambda)}{2^\lambda} + 1/2)) \\
&\geq 1/2 + 1/2(1/2 + \epsilon(\lambda) - \frac{q(\lambda)}{2^\lambda} - 1/2) \\
&= 1/2 + \frac{\epsilon(\lambda)}{2} - \frac{q(\lambda)}{2^{\lambda+1}} \ .
\end{aligned}
\tag{5.4}
$$

□

5.2 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 9) 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:

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

By component-wise aggregating the ciphertexts, we obtain:

$$
\mathsf{ct}^* = (g^{r_1+r_2}, g_T^{m_1+m_2} e(g,P)^{\rho_1' r_1 + \rho_2' r_2}) := (U^*, V^*) \tag{5.5}
$$

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:

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

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

Therefore,

$$
\begin{aligned}
g_T^{m_1+m_2} &= V^* e(\pi_{\rho_1}\pi_{\rho_2}, U^*)^{-1} e(U_1, \pi_{\rho_2}) e(U_2, \pi_{\rho_1}) \\
&= V^* e(g^{r_1+r_2}, P^{\rho_1'+\rho_2'})^{-1} e(U_1, \pi_{\rho_2}) e(U_2, \pi_{\rho_1}) \\
&= V^* e(g,P)^{-(\rho_1' r_1 + \rho_1' r_2 + \rho_2' r_1 + \rho_2' r_2)} e(U_1, \pi_{\rho_2}) e(U_2, \pi_{\rho_1}) \\
&= V^* e(g,P)^{-(\rho_1' r_1 + \rho_1' r_2 + \rho_2' r_1 + \rho_2' r_2)} e(g,P)^{\rho_2' r_1} e(g,P)^{\rho_1' r_2} \\
&= V^* e(g,P)^{-(\rho_1' r_1 + \rho_2' r_2)} \ .
\end{aligned}
\tag{5.6}
$$

We note 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 $\mathsf{H}(\rho)$.

5.3 Discussions

We just presented a practical HSWE scheme in Section 5.1. 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 the applications we have in mind, users need to prove the well-formedness of their ciphertexts, *e.g.*, their bid or vote is well-formed, or the plaintext is below a threshold to allow efficient decryption. These proofs would be costly in the target group $\mathbb{G}_T$. However, for example, given a ciphertext $c = (g^r, g_T^m g_\rho{}^r) := (U, V)$ of the scheme described in Figure 4, the prover could map the ciphertext element V to $V' \in \mathbb{G}$ and prove statements efficiently about this element $V' = g^m h^r$. Later, the prover can show the consistency of V and V' using standard techniques [DHMW23].

Efficient decryption. Decryption entails computing a small discrete logarithm instance in $\mathbb{G}_T$. There are well-known lookup table techniques for solving small discrete logarithms. Furthermore, these tables can be significantly compressed and made practical even for mobile phones using the techniques of [CCN21].

Avoiding the $\mathbb{G}_T$ 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.

6 Stateful HSWE Schemes

In this section, we propose another class of HSWE schemes that support exponentially-sized message spaces and is *stateful*. That is, the signing parties need to be aware of the set of ciphertexts to be (homomorphically) decrypted prior to the release of the corresponding signature. This is in contrast to previous *stateless* HSWE schemes where the signing parties only signed off of messages (*e.g.*, timestamps, epoch/block numbers, etc.) that are independent from the applications built on top of the signing parties.

We build our stateful HSWE described in Figure 5 using a pair of *modified*[5] Paillier encryption [Pai99] and RSA signature schemes [RSA78]. The correctness of the scheme is easy to follow. Now, we argue that this scheme is additively homomorphic. Let $\mathsf{ct}_1, \mathsf{ct}_2$ be two ciphertexts on distinct messages:

$$\mathsf{ct}_1 = (U_1, V_1) = ((1+N)^{m_1} \mathsf{H}'(\rho)^{Nr_1\rho}, \mathsf{H}'(\rho)^{Ner_1\rho})$$
$$\mathsf{ct}_2 = (U_2, V_2) = ((1+N)^{m_2} \mathsf{H}'(\rho)^{Nr_2\rho}, \mathsf{H}'(\rho)^{Ner_2\rho})$$

By component-wise aggregating the ciphertexts, we obtain:

$$\mathsf{ct}^* = (U^*, V^*) = ((1+N)^{m_1+m_2} \mathsf{H}'(\rho)^{N(r_1+r_2)\rho}, \mathsf{H}'(\rho)^{Ne(r_1+r_2)\rho})$$

Let $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ be a batch of ciphertexts under tag ρ generated via our proposed HSWE scheme, where $\mathsf{ct}_i = ((1+N)^{m_i} \mathsf{H}'(\rho)^{Nr_i\rho}, \mathsf{H}'(\rho)^{Ner_i\rho}) \bmod N^2$ for $i \in [k]$. One could homomorhically decrypt the corresponding plaintexts $m_1, \ldots, m_k$ by first computing the aggregated ciphertext:

$$\mathsf{ct}^* = (U^*, V^*) = ((1+N)^{\sum m_i} \mathsf{H}'(\rho)^{N\rho \sum r_i}, \mathsf{H}'(\rho)^{Ne\rho \sum r_i}) \tag{6.1}$$

And then use a (modified) RSA signature on V^* as $\pi_\rho := (V^*)^d = \mathsf{H}'(\rho)^{N\rho \sum r_i} \bmod N^2$. This would result in $(1+N)^{\sum m_i} \bmod N^2$ and thus $\sum m_i$ by solving the (easy) discrete logarithm in the subgroup $\langle (1+N) \rangle \subset \mathbb{Z}_{N^2}^*$.

Theorem 5. The HSWE scheme in Figure 5 is CPA-secure in the random oracle model under the Decisional Composite Residuosity (DCR) and RSA assumptions.

Proof. Assuming the existence of a PPT adversary $\mathcal{A}$ that breaks the CPA-security of the scheme, we can construct a PPT adversary $\mathcal{B}$ that breaks the security of the underlying DCR assumption. Before we proceed with the formal proof, we intuitively argue that the CPA security of the ciphertext $\mathsf{ct} = (U, V)$ is concluded from two facts. First, U is a random group element in $\mathbb{Z}_{N^2}^*$ and independent from the message m due to the randomness $r \in_R \mathbb{Z}_N^*$ which essentially works as a one-time pad. Second, due to the RSA assumption, it is computationally infeasible

[5] We essentially use Paillier-like and RSA-like schemes and not the original constructions. This is due to the use of the random oracle in the former and a modulus N^2 in the latter, which, however, do not have security implications as we show in our security proof.

for an adversary to learn $\mathsf{H}'(\rho)^{Nr\rho}$ from observing V that has a uniform distribution in $\mathbb{Z}^*_{N^2}$. Thus, for each pair of messages m_0 and m_1 (of the same sizes) the ciphertexts are computationally indistinguishable, and the distribution of the adversary's views is identical.

Suppose there is a PPT adversary $\mathcal{A}$ that wins the CPA-security game of $\mathsf{Exp}_{\mathsf{SWE}-\mathsf{IND}}$ with a (non-negligible) advantage $\epsilon(\lambda)$. Then, we show there is a PPT adversary $\mathcal{B}$ that wins the DCR game with a relevant advantage.

- The DCR challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda)$ to generate the RSA group public key $\mathsf{pk} = (N, e)$ and secret key $\mathsf{sk} = (p, q, d)$ and only sends $(1^\lambda, N, e)$ to the DCR adversary $\mathcal{B}$ and it further forwards them to the CPA adversary $\mathcal{A}$ for initialization.
- In the first phase, $\mathcal{A}$ can make as many queries as it would like to get their corresponding ciphertexts. Let denote them by $m_1, \ldots, m_t$, where t is upper bounded by some polynomial $q(\lambda)$.
- For each query $i \in [t]$, the DCR challenger $\mathcal{C}$ samples $b \xleftarrow{\$} \{0, 1\}$ and gives back to $\mathcal{B}$ a value y accordingly. If $b = 0$ then $y = x^N$, where $x \xleftarrow{\$} \mathbb{Z}^*_N$. Otherwise, it samples $y \xleftarrow{\$} \mathbb{Z}^*_{N^2}$.
- $\mathcal{B}$ then samples a random value $r_i \xleftarrow{\$} \mathbb{Z}^*_N$ and creates the queried ciphertexts as $\mathsf{ct} = (U, V) = ((1+N)^{m_i} y^{r_i}, y^{er_i})$ and sends it to the CPA adversary $\mathcal{A}$.
- $\mathcal{A}$ generates a challenge plaintext pair (m_0, m_1) where $|m_0| = |m_1|$ and send them to $\mathcal{B}$ to generate the corresponding challenge ciphertext. To do so, $\mathcal{B}$ randomly samples $b_r \xleftarrow{\$} \{0, 1\}$ and $r^\star \xleftarrow{\$} \mathbb{Z}^*_N$ and sends a query to $\mathcal{C}$ to get y. It then generates the challenge ciphertext as $\mathsf{ct}^\star = ((1 + N)^{m_{b_r}} y^{r^\star}, y^{er^\star})$ and sends it to the CPA adversary $\mathcal{A}$.
- In the second phase, $\mathcal{A}$ can make further encryption queries conditioned on the total number of queries is upper bounded by $q(\lambda)$. Then, it outputs the guess $\tilde{b}$ regarding if the challenge ciphertext $\mathsf{ct}^\star$ encrypts m_0 or m_1.
- If $\tilde{b} = b_r$, then the DCR adversary $\mathcal{B}$ outputs $\hat{b} = 0$ in its own game and else it outputs $\hat{b} = 1$.

Note that in case $b = 0$, the ciphertext simulated by $\mathcal{B}$ has an identical distribution to the one expected from $\mathcal{A}$ in its CPA-security game under ρ in the random oracle model. Moreover, applying decryption after encryption produces the original message m as required. Looking forward, we can conduct a similar probabilistic analysis as in Theorem 4 to compute the winning probability of the DCR adversary in its own game that is equivalent to $\Pr[\hat{b} = b] = 1/2 + \frac{\epsilon(\lambda)}{2} - \frac{q(\lambda)}{2^{\lambda+1}}$.

We based our security reduction on the hardness of the DCR assumption and showed a successful CPA-security attack results in a successful attack on the underlying hardness assumption. However, observe that the security of our stateful HSWE also relies on the hardness of the RSA assumption. Given a ciphertext $\mathsf{ct} = (U, V) = ((1 + N)^m \mathsf{H}'(\rho)^{Nr\rho}, \mathsf{H}'(\rho)^{Ner\rho})$, an attacker breaking the RSA assumption can learn $\mathsf{H}'(\rho)^{Nr\rho}$ from V and then retrieve $(1 + N)^m$ from U.

□

Remark 3. It is well-known that the plain RSA signature is forgeable due to its inherent homomorphism. That is, having two signatures $\sigma_1 = m_1^d$ and $\sigma_2 = m_2^d$, one can generate a valid signature on $m_1 m_2$ by simply computing $\sigma_1 \sigma_2 = (m_1 m_2)^d$. A common way to fix the issue is to use *"hash-then-sign"* variant. However, following this would affect the homomorphism of our HSWE scheme in Figure 5. Interestingly, we observe that our scheme maintains its security against an adversary exploiting homomorphism/forgery due to the use of fresh randomness for generating each ciphertext. That is, it essentially does not change the advantage of adversary in decrypting the challenge ciphertext. More accurately, given two signatures $\sigma_1 = V_1^d$ and $\sigma_2 = V_2^d$ the probability of decrypting a challenge ciphertext $\mathsf{ct}^\star$ with an (aggregated) signature $\sigma = \sigma_1 \sigma_2$ (*i.e.,* having a ciphertext $\mathsf{ct}^\star$ where for its underlying randomness it holds $r^\star = r_1 + r_2$) is negligible.

Remark 4. Despite its fast verification, the RSA signature may not be a favorable option for various applications deployed on top of blockchain. This is mainly due to the relatively large size for signature and public keys (*e.g.,* 256 bytes). However, we highlight that it could be an attractive choice for HSWE given that just a single signature is enough for a (homomorphic) batch decryption. Moreover, this single signature only needs to live in CALLDATA, *i.e.,* temporal storage: after the transaction call is finished and necessary state changes are performed (*e.g.,* the winner of the election has been granted with some rights), the signature can be safely discarded for efficiency reasons. This is particularly interesting given that in a blockchain environment permanent storage is extremely expensive

A stateful HSWE scheme

Public parameters: a public tag (e.g., epoch number or timestamp), a hash function $\mathsf{H}'(\mathbb{N}) \to \mathbb{Z}^*_{N^2}$ modeled as random oracle, an RSA group public key $\mathsf{pk} = (N, e)$, with the corresponding secret key $\mathsf{sk} = (p, q, d)$ where $N = pq \wedge ed \equiv 1 \mod \phi(N^2)$.

Encryption: $\mathsf{Enc}(m, \rho, \mathsf{pk}) \to \mathsf{ct} := (U, V)$.

1. Samples the randomness $r \in_R \mathbb{Z}^*_N$ and compute $U = (1+N)^m \mathsf{H}'(\rho)^{Nr\rho} \mod N^2$
2. Let $V = \mathsf{H}'(\rho)^{Ner\rho} \mod N^2$.

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

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho) \to \pi_\rho := V^d \mod N^2$. // This is the RSA signature on V. Hence, this scheme is stateful as the signer needs to be aware of the to-be-decrypted ciphertext.

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

1. Parse the ciphertext ct as (U, V).
2. Let $W := U/\pi_\rho$.
3. Solve the *easy* discrete logarithm instance for $W \mod N^2$ for the base $(1+N)$ to obtain m.

Return: m.

Figure 5: A stateful HSWE scheme with exponentially-sized message space.

(unlike temporary storage). Furthermore, we can consider an RSA-based streaming randomness beacon [BCK+23] as the signing oracle with *history generation*, *i.e.,* one could derive all the previous beacon values (*i.e.,* signatures) from reading the current one. So, the last value suffices to derive all past outputs, demanding only a constant size storage cost.

Remark 5. An established batching technique known as *RSA screening* [BGR98] enables the efficient verification of multiple signatures. This method allows an aggregator to generate a proof of constant size, certifying that a given set of distinct messages has been signed under a public key (N, e). Rather than verifying each signature individually, the verifier can authenticate the entire batch by validating this single proof, thereby optimizing the verification process. This approach not only enhances computational efficiency but also reduces data transmission overhead. A brief introduction to this technique is provided below, with further details available in [BGR98].

Given an RSA public key (N, e), consider a sequence of k distinct messages $m_0, m_1, \ldots, m_k$ accompanied by their corresponding RSA signatures $\sigma_0, \sigma_1, \ldots, \sigma_k$. The aggregator computes the proof, denoted as π_{sig}, simply by computing the product of all signatures modulo N:

$$\pi_{\text{sig}} = \prod_{i \in [k]} \sigma_i \mod N$$

The verifier checks

$$\pi_{\text{sig}}^e \stackrel{?}{=} \prod_{i \in [k]} H(m_i) \mod N$$

where $H(\cdot)$ is the hash function used for the RSA signature.

6.1 Privacy-preserving HSWE

A potential downside of all of our schemes up to this point is that after the signature σ has been released, every ciphertext, and not only the homomorphically aggregated one, is publicly decryptable. In some applications this is unwanted as it translates, for instance in the case of a voting application, to the lack of individual ballot privacy. This motivates us to devise a method to retain the privacy of each individual plaintext, and only allow for the decryption of the final aggregated ciphertext. We define the experiment for a privacy-preserving HSWE in Figure 7.

A privacy-preserving HSWE scheme

Public parameters: a public tag ρ (e.g., epoch number or timestamp) and underlying group description pp.

Encryption: For some message m:

1. Sample a secret $\mathsf{s} \in_R \mathbb{F}_p$.
2. Compute the Shamir shares $\mathsf{Share}(\mathsf{s}) \to s_1, \ldots, s_n$ and send s_i to each (signing) party $i \in [n]$.
3. Run $\mathsf{Enc}(m, \rho, \mathsf{pk}) \to \mathsf{ct}$ and set $\mathsf{ct}' = g^{\mathsf{s}}\mathsf{ct}$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \rho) \to \pi_\rho$.

Decryption: For a batch of (privacy-preserving) HSWE ciphertexts $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$:

1. Each (signing) party $i \in [n]$ computes $s^*_i = \sum_{j=1}^{k} s_{ij}$.
2. For a threshold subset of (aggregated) shares $s^\star_i$, reconstruct the secret in the exponent $\mathsf{Recon}(s^*_1, \ldots, s^*_{t+1}) \to g^{\mathsf{s}^*} = g^{\mathsf{s}_1 + \ldots + \mathsf{s}_k}$.
3. Compute $\mathsf{ct}^* = \prod_{j=1}^{j=k} \mathsf{ct}'_j = g^{\mathsf{s}^*} \prod_{j=1}^{j=k} \mathsf{ct}_j$
4. Run $\mathsf{Dec}(\mathsf{ct}^*, \rho, \pi_\rho)$ and return $m^* = m_1 + \ldots + m_k$

Figure 6: A privacy-preserving HSWE scheme.

We outline our protocol that turns any HSWE encryption scheme into a privacy-preserving variant, *i.e.,* all the plaintexts remain hidden, except the final aggregated one. The underlying trick is to blind each HSWE ciphertext using some randomness and secret share it towards the signing parties. Upon decryption, the signing parties reconstruct the *aggregated* randomness corresponding to the set of HSWE ciphertexts and release it together with the signature. This transformation, however, comes at the cost of rendering the underlying HSWE scheme stateful given that the signing parties need to be aware of the set of HSWE ciphertexts to release the corresponding aggregated randomness. For concreteness, we present our privacy-preserving HSWE scheme in Figure 6 assuming groups of prime order and Shamir Secret sharing while the technique is generally agnostic and can extend to groups of unknown order and/or other variants of (linear) secret sharing.

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$, $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$[a]
$\mathcal{A}^{\Sigma^{\mathcal{O}}}(\mathsf{pp}, \mathsf{pk}) \to (\rho^\star, m_0, m_1)$
$b \xleftarrow{\$} \{0, 1\}$
where for any batch of $[k]$ users/ciphertexts under $\rho^\star$:

$\mathsf{Enc}(m_b^{[k]}, f(\rho^*), \mathsf{pk}) \to \mathsf{ct}_b^{[k]}$

Sample $\mathsf{s}_{[k]} \in_R \mathbb{F}_p$ and set $\mathsf{ct}'_b{}^{[k]} = g^{\mathsf{s}_{[k]}} \mathsf{ct}_b^{[k]}$

$\mathcal{A}^{\Sigma^{\mathcal{O}}}(\mathsf{pp}, \mathsf{pk}, \{\mathsf{ct}'_b{}^{[k]}\}) \to b'$
output 1 if $b' = b$

[a] We implicitly assume the key generation KeyGen is run with less than a threshold corruption, guaranteeing its security and liveness.

Figure 7: The experiment for privacy-preserving HSWE.

Theorem 6. The HSWE scheme in Figure 6 is privacy-preserving, assuming less than a threshold number of signing parties is corrupted.

Proof. The correctness of the scheme is easy to check. Given that each HSWE ciphertext ct is blinded with a uniform randomness s as $\mathsf{ct}' = g^{\mathsf{s}}\mathsf{ct}$ any two ciphertexts ct_1 and ct_2 are statistically indistinguishable. Assuming less than a

threshold corruption (for signing parties), the scheme is privacy-preserving as no PPT adversary can learn about an individual plaintext m (corresponding to some ct') knowing only the aggregated randomness (in the exponent). □

7 Performance

7.1 Theoretical Performance

We compare the theoretical performance of our proposed HSWE schemes in Table 1.

HSWE scheme	Assumption	$\lvert\mathcal{M}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert$	Enc	Dec	Stateless
Section 4	Quadratic-residuosity	2	$\lvert\mathbb{Z}_N\rvert$	$\lvert\mathbb{Z}_N\rvert$	$5\lvert\mathbb{Z}_N\rvert+\mathsf{H}^a$	$\lvert\mathbb{Z}_N\rvert$	●
Section 5.1	Decisional Bilinear Diffie-Hellman	$\mathsf{poly}(\lambda)$	$\lvert\mathbb{G}\rvert$	$\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{G}\rvert+\lvert\mathsf{P}\rvert$	$\lvert\mathbb{G}\rvert+\lvert\mathsf{P}\rvert+\mathsf{DL}$	●
Section 6	DCR and RSA	$\lvert\mathbb{Z}_N^*\rvert$	$\lvert\mathbb{Z}_N\rvert$	$2\lvert\mathbb{Z}_{N^2}\rvert$	$3\lvert\mathbb{Z}_{N^2}\rvert+2\mathsf{H}$	$2\lvert\mathbb{Z}_{N^2}\rvert$	○

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 bit of abuse in notation, the number of applied group $\mathbb{G}, \mathbb{G}_T$ and hashing operations H computed during encryption and decryption are denoted by the size of the corresponding groups. In certain HSWE schemes, a small discrete logarithm computation for decryption is needed that is denoted by DL.

[a] This only holds in expectation as one needs to compute four Jacobi symbols in expectation, to find a quadratic residue $\mod N$.

8 Related Work

Signature-based witness encryption (SWE) allows one to build an encryption scheme from a signature scheme. Intuitively, the ciphertext is created with respect to some statement (*e.g.*, tag) and the decryption is done if and only if there is a valid witness (*i.e.*, signature) on the corresponding statement. The authors in [GMR23, DHMW23] showed how to construct an SWE using (threshold) BLS signature. Their constructions resembles that of Boneh-Franklin identity-based encryption [BF01] where the identity is essentially the statement under which the ciphertext is generated and the secret key is the corresponding BLS signature on the hash of the identity. While the authors in [GMR23] assume the existence of a threshold committee with honest majority to encrypt the plaintext under their common public key, Mcfly [DHMW23] consider a setting where each t-out-of-n share of plaitext is encrypted under some validator's public key so that a threshold number of (multi-)signatures is needed as witness for decryption. This makes the size of ciphertext to grow linearly in the number of public keys (*i.e.*, committee size), an issue recently addressed by the work of Avitabile et al. [ADM+24], though only of theoretical interest due to the use of expensive tools such as indistinguishability obfuscation [KLW15]. However, the approach mentioned above relies on an all-or-nothing decryption mechanism for a batch of ciphertexts under a given identity.

Recent works, such as [CGPP24, AFP24], have explored how to efficiently preserve the confidentiality of individual ciphertexts rather than having all get decrypted at once. This is particularly relevant for applications such as MEV protection, which requires privacy for pending transactions. In particular, Choudhuri et al. [CGPP24] proposed a commitment-based witness encryption where the messages is encrypted towards some (polynomial) commitment and the decryption is done using the corresponding proof of opening. This intriguing design protects the privacy of all the messages whose ciphertexts are not part of interpolating the degree-B polynomial, for a batch of B ciphertexts to be decrypted. On the other hand, our privacy-preserving HSWE offers privacy for all the individual messages except for the aggregated one. The authors in vetkeys [CCN+23] introduced techniques for the private (and verifiable) transfer of BLS signature shares from validators to users. This enables private decryption of ciphertexts by the individuals, rather than making the decryption publicly available. Garg et al. [GKM+24] devised a protocol that generates multi-receiver ciphertexts along with a (constant-size) proof for public verification. Their underlying building block is a scheme called multi-identity based encryption that encrypts multiple messages under multiple identities. We noticed that their underlying technique is similar to that of our CPA-secure HSWE with $\mathsf{poly}(\lambda)$-sized message space, while the motivation and security proofs are different. As the decryption in an SWE scheme is done uniquely using a signature on the current epoch, it does not allow to decrypt any ciphertexts generated under the previous epochs. To tackle this issue that demands a linear storage cost for storing all the

previous epoch keys (*i.e.*, signatures), previous works such as [BMS22] proposed solutions that allow unlocking the (old) ciphertexts under prior epochs with only a logarithmic size on-chain cost. As mentioned, using an RSA-based randomness beacon with history generation enables our stateful HSWE to achieve this property only with a constant size storage.

9 Conclusion and Future Work

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 homomorphism that already turns 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 in the RSA and elliptic curve groups endowed with a bilinear pairing.

Despite our work on HSWE being over-arching, we leave open several important directions for future work.

Homomorphism for larger message spaces. Our proposed *stateless* HSWE schemes support homomorphism for a limited $\mathcal{O}(\mathsf{poly}(\lambda))$-sized message space since the decryptor needs to compute a discrete logarithm problem in the corresponding cryptographic group, *i.e.*, g_T^m for a message m. Future HSWE schemes should be more practical by supporting larger, exponentially-sized message spaces. Moreover, ideally one should have the message in one of the source groups, *i.e.*, $\mathbb{G}_1$ or $\mathbb{G}_2$.

Robust HSWE **schemes.** Similarly to non-homomorphic SWE schemes [DHMW23], it would be beneficial to be able to encrypt not only to a single verification key pk but also to a threshold of multiple verification keys $\{\mathsf{pk}\}_{i=1}^n$. In our current schemes, the signature generation can be optionally thresholdized to avoid a single point of failure. To support a wider range of applications, just like in [DHMW23], future HSWE schemes should allow the encryption of a message under a set of verification keys for better liveness should some parties fail to produce their signatures. It seems reachable to modify the schemes in [DHMW23] to endow their message spaces with homomorphism as well.

Post-quantum secure HSWE **schemes.** Finally, for future applications and deployments, achieving post-quantum security for HSWE schemes will have the utmost importance. For digital signatures and key encapsulation mechanisms, this is already mandated by NIST. In particular, by 2035, NIST dictates that no federal and governmental bodies can use pre-quantum cryptography for digital signatures and key encapsulation. We anticipate a similar post-quantum transition needs to happen for applications using HSWE schemes. To the best of our knowledge, currently, there is no known post-quantum *unique* signature scheme, *cf.* Theorem 1.

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. 4

[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, 3, 19

[AFP24] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. *Cryptology ePrint Archive*, 2024. 5, 19

[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

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018. 1, 2

[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. 2

[BCK+23] Donald Beaver, Konstantinos Chalkias, Mahimna Kelkar, Lefteris Kokoris-Kogias, Kevin Lewi, Ladi de Naurois, Valeria Nikolaenko, Arnab Roy, and Alberto Sonnino. Strobe: Streaming threshold random beacons. In *5th Conference on Advances in Financial Technologies (AFT 2023)*, pages 7–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023. 17

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001. 3, 5, 12, 13, 19

[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. 1

[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. 1, 2

[BGR98] Mihir Bellare, Juan A Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology—EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31–June 4, 1998 Proceedings 17*, pages 236–250. Springer, 1998. 17

[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, 5, 7, 14

[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. 20

[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. 7

[BS07] Daniel Bernstein and Jonathan Sorenson. Modular exponentiation via the explicit chinese remainder theorem. *Mathematics of Computation*, 76(257):443–454, 2007. 1, 2

[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. 1, 2

[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. 15

[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. 19

[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. 19

[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

[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. 3

[DGS20] Samuel Dobson, Steven D Galbraith, and Benjamin Smith. Trustless groups of unknown order with hyperelliptic curves. *IACR Cryptol. ePrint Arch.*, 2020:196, 2020. 2

[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. 2, 3, 5, 15, 19, 20

[DRA20] Team drand, drand project website. `https://drand.love`, 2020. 2

[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. 3

[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. 2

[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. 3

[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. 2

[GKM+24] Sanjam Garg, Aniket Kate, Pratyay Mukherjee, Rohit Sinha, and Sriram Sridhar. Insta-pok3r: Real-time poker on blockchain. *Cryptology ePrint Archive*, 2024. 19

[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. 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. 2, 3, 5, 19

[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. 1, 2, 4

[Kal00] Burt Kaliski. Ieee standard specifications for public-key cryptography. ieee std 1363–2000, 2000. 8

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

[KLJD23] Alireza Kavousi, Duc V Le, Philipp Jovanovic, and George Danezis. Blindperm: Efficient mev mitigation with an encrypted mempool and permutation. *Cryptology ePrint Archive*, 2023. 4

[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. 19

[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. 5, 10, 11

[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. 1

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In *Annual International Cryptology Conference*, pages 620–649. Springer, 2019. 1, 2, 3

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto*, 2008. 2

[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

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999. 4, 5, 15

[Rab79] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. 1979. 3, 8

[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. 1

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 4, 5, 7, 15

[RSKK] Ben Riva, Alberto Sonnino, and Lefteris Kokoris-Kogias. Seahorse: Efficiently mixing encrypted and normal transactions. 4

[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. 2

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996. 1, 2, 3

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

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

[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. 1, 2, 3

[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. 15

[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. 1

[WW20] Benjamin Wesolowski and Ryan Williams. Lower bounds for the depth of modular squaring. *Cryptology ePrint Archive*, 2020. 1