Practical Silent Threshold Signatures and Silent Threshold Encryption for Dynamic Committees*

Yifei He
School of Computer Science
Peking University
Beijing, Beijing, China
heyifei@pku.edu.cn

Zheng Zhou
School of Computer Science
Peking University
Beijing, Beijing, China
zhengzhoum@stu.pku.edu.cn

Yu Chen†‡
School of Cyber Science and Technology
Shandong University
Qingdao, Shandong, China
yuchen.prc@gmail.com

Zhi Guan
National Engineering Research Center For Software Engineering
Peking University
Beijing, Beijing, China
guan@pku.edu.cn

Zhong Chen§
School of Computer Science
Peking University
Beijing, Beijing, China
zhongchen@pku.edu.cn

ABSTRACT

Silent threshold signatures (STS) and encryption (STE) enable threshold cryptography without interactive distributed key generation, allowing a group of N parties to non-interactively generate a joint public signature verification key or an encryption key. However, modern distributed systems (such as Ethereum) rely on small, dynamically changing committees of size $n \ll N$ for efficiency, and existing silent threshold schemes either fail to support this dynamic setting or suffer from severe scalability issues. To the best of our knowledge, the only prior STS construction for dynamic committees, Dyna-hinTS, requires an aggregation time of $O(N \log N)$ per epoch, tightly coupling the cost to the global system size rather than the small active committee. Furthermore, no STE scheme for dynamic committees has been proposed yet.

In this work, we present practical silent threshold signature and encryption schemes for dynamic committees, reducing the aggregation cost so that it depends only on the committee size n. For signatures, we redesign the Dyna-hinTS framework by replacing its Plonk-style SNARKs with linear pairing checks and a new polynomial commitment for representing the committee, yielding an aggregation time of $O(n \log^2 n)$. We also introduce the first silent threshold encryption scheme for dynamic committees with matching efficiency. We further significantly optimize the silent setup phase common to prior STS and STE schemes, reducing each party's one-time setup (i.e., generating the setup data, referred to as a "hint") cost from $O(N^2)$ to $O(N)$.

We implement our schemes in Rust, and the results demonstrate practicality at scale. For a system parameterized with $N = 2^{20}$ and $n = 2^{10}$, the per-party hint generation takes 97.5 seconds, and signature aggregation takes 0.145 seconds, achieving a $> 140\times$ improvement over Dyna-hinTS. At the same time, our aggregated signature size, verification key size, and verification time (with access to the epoch committee representation) remain constant.

CCS CONCEPTS

• **Security and privacy** → **Digital signatures**; **Public key encryption**.

KEYWORDS

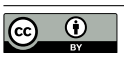
Threshold Cryptography, Silent Threshold Signature, Silent Threshold Encryption, Dynamic Committees, Non-interactive Setup

1 INTRODUCTION

Threshold signatures and threshold encryption are important primitives in modern cryptography. Given a set of N parties and a threshold $t \leq N$, any subset of at least t parties can jointly sign a message or decrypt a ciphertext, while fewer than t parties cannot forge signatures or learn any information from the ciphertext. Traditionally, constructing such threshold cryptography schemes relies on distributed key generation (DKG) as the setup phase. This process requires interaction among all parties to jointly generate and share a single global secret key, and to publish the corresponding public key for signature verification or encryption.

However, the interactive nature of DKG incurs high communication and computational overhead, and typically assumes a synchronous setting. Moreover, jointly generating the shared key makes it difficult to dynamically adjust the threshold or the set of parties, and these limitations become increasingly severe as the system scales up. This has motivated recent works [9, 20, 26, 27, 35] on *silent* threshold cryptography, which replaces DKG with a non-interactive setup. Garg et al. [26, 27] introduced silent threshold signature (STS) and encryption (STE), where each party provides an additional "hint" that implicitly encodes information about its private key, and the aggregator can use these "hints" to generate

*This is the full version of the paper accepted to the 2026 ACM Conference on Computer and Communications Security (CCS '26, Cycle B).
†Also with, State Key Laboratory of Cryptography and Digital Economy SecurityShandong University.
‡Also with State Key Laboratory of Cryptology.
§Also with, School of AI and Liberal ArtsBeijing Normal-Hong Kong Baptist University.



This work is licensed under a Creative Commons Attribution 4.0 International License.

a succinct proof, demonstrating the validity of the aggregated signature or performing decryption on the ciphertext. These schemes also supported dynamic threshold adjustment and achieved constant-size verification keys and signatures, as well as constant verification time.

On the other hand, large-scale distributed systems [3–5, 19, 28] are gradually shifting to **dynamic committee settings** for efficiency and security considerations. In such a setting, the system operates in recurring epochs, and only a randomly chosen small subset of parties Com_e of size $n \ll N$ is authorized to sign or decrypt in each epoch e. This significantly reduces the cost without sacrificing security, because t and n can be much smaller than N to ensure that an adversary controlling f parties has negligible probability of controlling $\geq t$ parties within a randomly sampled committee.

However, existing silent threshold schemes described above cannot restrict authorization to any specific subset of parties: their silent setup is performed globally by all parties, which can only guarantee that $\geq t$ of all N (instead of n) parties generated the signature or decrypted the ciphertext. To address this problem, Kate et al. [30] further proposed the notion of silent threshold signature for dynamic committees (c-STS) and a construction called DynahinTS, which shared a common construction with [26]. Despite its conceptual breakthrough, the signature aggregation of DynahinTS suffers from an $O(N \log N)$ computational cost: to additionally prove that all partial signatures used for aggregation are from the required committee, the aggregator must interpolate and evaluate dense global state polynomials of degree $O(N)$ to generate several Plonk-style proofs. This forces the aggregation cost to scale with the massive global system rather than the small active committee, which may become a bottleneck in settings such as Ethereum, where $N \approx 2^{20}, n \approx 2^{10}$ and the committee changes every 12 seconds. Intuitively, if signing or encryption involves only a small committee, the associated overhead should depend only on the size of this committee rather than the size of the entire system. In consequence, we ask the following question:

Can we construct a silent threshold signature scheme for dynamic committees whose aggregation cost only depends on the size of the committee?

Furthermore, we observe that silent threshold signature and encryption schemes share similar structures, but only the former has been extended to support dynamic committees. Considering that threshold encryption schemes also have unique applications in decentralized systems, bridging this gap is highly desirable. A typical use case is the encrypted mempool, where users can encrypt their transactions before submitting them to mitigate Miner Extractable Value (MEV), and these transactions can be threshold-decrypted by consensus nodes only after their order is fixed. To deploy such a mechanism on large blockchains such as Ethereum, the decryption authority must be restricted to the specific committee of the current epoch, which changes dynamically. Therefore, a practical STE scheme for dynamic committees could be an important building block toward encrypted mempools. This leads to our second question:

Can we construct a silent threshold encryption scheme for dynamic committees?

In this work, we answer both of them in the affirmative: we demonstrate how to build a practical silent threshold signature and encryption scheme for dynamic committees based on existing techniques [26, 27, 30], with aggregation cost quasilinear in the committee size n and independent of the system size N.

1.1 Our Contributions

Practical STS scheme for dynamic committees. We propose a practical STS scheme for dynamic committees, whose signature aggregation time dynamically adapts to each epoch and scales only quasilinearly in the corresponding committee size n (rather than the system size N). Inspired by the degree-check techniques from [27], we redesign the polynomial commitment representing the epoch committee to strictly enforce that the signers form a subset of this actively rotating committee, which allows us to remove the Plonk-style proofs in [30] and prove the subset relation purely through polynomial divisibility, yielding fully linear verification equations over pairing groups. Furthermore, to ensure the system remains oblivious to the system size N even as the committee changes, we mathematically bypass full-domain polynomial interpolation in signature aggregation and propose a method to compute all required group elements strictly over the dynamically updated subset of roots of unity (corresponding to the epoch committee). As a result, the complexity of signature aggregation is decoupled from the global setup, reducing from $O(N \log N)$ (which was due to generating the Plonk-style proofs) to $O(n \log^2 n)$, making our scheme suitable for much larger systems with dynamic committees.

Silent threshold encryption scheme for dynamic committees. We introduce the notion of silent threshold encryption for dynamic committees (c-STE), extending STE to support dynamic committee settings in which the decryption authority rotates to a specific committee for epoch e. During the one-time silent setup phase, each party generates a public/secret key pair independently and a public hint that contributes to a compact encryption key. When the system transitions to a new epoch e, only a subset Com_e (with size n and threshold t) of all parties is authorized for decryption without any re-setup. Messages encrypted to epoch e can be decrypted by any $\geq t$ parties in Com_e in a non-interactive manner. The security of c-STE guarantees that an adversary cannot distinguish encryptions of two chosen messages as long as $< t$ parties in Com_e are corrupted (there is no limitation on the adversary for corrupting parties beyond Com_e). Furthermore, we present a concrete c-STE construction that inherits the same efficiency properties as our c-STS scheme: the costs of epoch representation and decryption aggregation scale only with the committee size n, making it highly practical for large-scale systems.

Optimized silent setup. We significantly improve the setup phase of silent threshold cryptography schemes based on the idea proposed in [26], by reducing the per-party hint generation complexity from $O(N^2)$ to $O(N)$. To compute $O(N)$ hint elements, existing schemes natively compute multi-scalar multiplications (MSMs) of size $O(N)$ for each of them. In comparison, we exploit the algebraic properties of Lagrange basis polynomials over roots of unity and demonstrate how to mathematically express these complex quotient polynomials as sparse linear combinations of precomputed

base commitments, which allows us to compute each element using only $O(1)$ group operations on average. This optimization can be applied to [26, 27, 30] and this work since all of these schemes share a common structure and the same setup phase. Considering that the related technique does not correspond to any specific silent threshold signature or encryption schemes, this part may be of independent interest.

Implementation and evaluation. We implement our schemes in Rust and evaluate them on realistic system scales. For a system parameterized with $N = 2^{20}$ and $n = 2^{10}$, generating a single hint takes only 97.5 seconds, while the naive approach used in prior schemes becomes infeasible. Signature aggregation takes only about 0.145 seconds, yielding a $> 140\times$ improvement over Dyna-hinTS. Partial signing and aggregated-signature verification (with access to the epoch committee representation) both run in constant time (1.4 ms and 7.3 ms, respectively), with a constant-size aggregated signature and verification key. We also analyze and estimate the cost of the one-time preprocessing, demonstrating its practicality in large-scale systems.

1.2 Related Work

Silent setup for threshold cryptography. There are two main approaches to constructing a silent threshold signature (STS) scheme. The first delegates the public key aggregation to the signature verifier. A typical example is the BLS multi-signature [9, 13], in which the aggregator must provide the list of signers as part of the aggregated signature, and the verifier aggregates all corresponding public keys. As a result, both the signature size and verification cost extend linearly with the number of signers, leading to high communication and storage overhead. While recent works optimize either signature size [6, 7] or verification key size [35], they fail to minimize both simultaneously, resulting in massive signatures.

The second approach delegates the public key aggregation to the aggregator. This yields succinct aggregated signatures, but the aggregator must additionally provide a correctness proof. Succinct non-interactive arguments (SNARKs) naturally meet this requirement, as their succinctness and efficiency greatly reduce communication costs and verification overhead. However, the aggregation cost and parameter choices in such schemes are restricted by the underlying proof system. Building on this approach, several STS schemes [20, 26] design customized SNARKs to reduce aggregation costs further. A typical example is hinTS [26], which represents the set of signers as a binary vector. The aggregator commits to this vector and uses a set of customized succinct Plonk [23]-style proofs to ensure that the number of signers meets the threshold. The aggregated public key is computed as the inner product between this vector and the list of public keys, and its correctness is guaranteed using an inner-product argument called generalized sumcheck [37]. With these techniques, hinTS achieved a constant signature size, verification key size, and verification complexity.

Constructing silent threshold encryption (STE) schemes is significantly more challenging, and no practical solutions were known for a long time. A closely related notion is distributed broadcast encryption, which can be viewed as a special case of STE with $t = 1$. However, even this special case has only feasible constructions [16, 17, 22, 32, 40] in the past several years. Before the work of Garg et al. [27], it was widely believed that building STE necessarily required heavy cryptographic primitives such as indistinguishability Obfuscation [10] (e.g., [38]) or universal witness encryption [24, 25]. The work of Garg et al. [27] first proposed the notion of STE and provided a concretely efficient construction based on pairing-friendly groups. Specifically, they proposed a paradigm that derives signature-based witness encryption from signature schemes with linear verification, and applied this paradigm to a linear-verification variant of hinTS [26] to obtain the corresponding STE scheme.

Threshold cryptography schemes for dynamic committees. Since traditional DKGs do not support changes to the committee, prior work [11, 21, 33, 39, 41] has long focused on dynamic secret sharing to change parties while keeping the secret value (public key) fixed. However, this incurs higher complexity and tolerates lower adversarial thresholds. Recently, Baird et al. [8] proposed Multiverse Threshold Signature (MTS), enabling arbitrary universes (subsets) of signers to hold distinct thresholds. Although parties hold constant-size keys and the size of partial signatures is independent of the number of universes, the signers must interact among themselves and the aggregator to set up the universe's verification key when they are assigned to a particular universe. This interaction was eliminated by the STS scheme [26] via a one-time global setup.

Building on this, Kate et al. [30] proposed Dyna-hinTS to support dynamic committees. Dyna-hinTS allows the committee to change per epoch using the same global verification key from a one-time setup. It yields an improved MTS scheme that bypasses any per-universe setup requirement, though its aggregation efficiency remains constrained by the global system size.

2 PRELIMINARIES

Notions. We use λ for the security parameter, and $\mathsf{negl}(\lambda)$ for the negligible function, i.e., for any polynomial f, there exists a λ_0 such that for all $\lambda > \lambda_0, \mathsf{negl}(\lambda) < 1/f(\lambda)$. We use $[n]$ for the set $\{1, 2, \ldots, n\}$. For a set S, we use S' for $S \cup \{0\}$, and use $[n]'$ for $[n] \cup \{0\} = \{0, 1, \ldots, n\}$.

2.1 Bilinear Groups

We use $(\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_T)$ for a pairing-friendly group, where $\mathbb{Z}_p$ is a prime field of order $p = \exp(\Omega(\lambda))$, and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p with a non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ are uniformly sampled generators such that $e(g_1, g_2) = g_T$. For $x \in \mathbb{Z}_p$, we use $[x]_s$ to denote $x \cdot g_s$, and we denote the group operation additively, namely $[x]_s + [y]_s = [x + y]_s$ for $s \in \{1, 2, T\}$. We denote the pairing operation $e([x]_1, [y]_2)$ by $[x]_1 \circ [y]_2 = [y]_2 \circ [x]_1 = [x \cdot y]_T$. We also use this notion for matrices of group elements, i.e., for a $k_1 \times k_2$ matrix $\boldsymbol{A} = \{[a_{i,k}]_{\alpha_k}\}$ and a $k_2 \times k_3$ matrix $\boldsymbol{B} = \{[b_{k,j}]_{\beta_k}\}$, if $\alpha_k + \beta_k = 3$ for all $k = 1, 2, \ldots, k_2$, then $\boldsymbol{A} \circ \boldsymbol{B}$ is well-defined, and its result is a $k_1 \times k_3$ matrix $\boldsymbol{C} = \{[c_{i,j}]_T\}$ that satisfies $[c_{i,j}]_T = \sum_{k=1}^{k_2} [a_{i,k}]_{\alpha_k} \circ [b_{k,j}]_{\beta_k}$. It is obvious that $c_{i,j} = \sum_{k=1}^{k_2} a_{i,k} b_{k,j}$.

2.2 BLS Signature

Let $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{G}_2$ be a random oracle. The BLS signature scheme [14] consists of the following algorithms:

- $\mathsf{BLS.Gen}$: It samples a random $\mathsf{sk} \leftarrow \mathbb{Z}_p$ and outputs a public/secret key pair as $(\mathsf{pk} = [\mathsf{sk}]_1, \mathsf{sk})$.
- $\mathsf{BLS.Sign}(\mathsf{sk}, \mathsf{msg})$: It outputs a signature $\sigma = \mathsf{sk} \cdot \mathcal{H}_2(\mathsf{msg}) \in \mathbb{G}_2$.
- $\mathsf{BLS.Verify}(\mathsf{pk}, \mathsf{msg}, \sigma)$: It verifies the validity of the signature by checking $\mathsf{pk} \circ \mathcal{H}_2(\mathsf{msg}) \stackrel{?}{=} [1]_1 \circ \sigma$.

2.3 Proofs of Equality of Discrete Logarithms

Let $\mathcal{H}_0 : \{0,1\}^* \to \mathbb{Z}_p$ be a random oracle independent of $\mathcal{H}_2$. We use the Fiat-Shamir transform of the Chaum–Pedersen protocol [18] to prove knowledge of a scalar k such that $X = kA, Y = kB$, where $A, X \in \mathbb{G}_a$, $B, Y \in \mathbb{G}_b$, and $a, b \in \{1, 2\}$. In our applications, $A = [1]_1$.

The Eq-DL protocol [29] consists of the following algorithms:

- $\mathsf{EqProve}(A, X, B, Y; k)$:
  (1) Sample $r \leftarrow_\$ \mathbb{Z}_p$.
  (2) Compute $\alpha = rA$ and $\beta = rB$.
  (3) Compute $c = \mathcal{H}_0(A, X, B, Y, \alpha, \beta)$, and $s = r + ck$.
  (4) Output $\pi = (c, s)$.
- $\mathsf{EqVrfy}(A, X, B, Y; \pi)$:
  (1) Reject malformed inputs and group elements of incorrect types.
  (2) Parse $\pi = (c, s)$ and compute $\alpha = sA - cX, \beta = sB - cY$.
  (3) Accept if and only if $c = \mathcal{H}_0(A, X, B, Y, \alpha, \beta)$.

This protocol is used to require users to prove knowledge of their sk during registration and partial signing (see Construction 1).

2.4 Polynomials over a Prime Field

Let $\omega \in \mathbb{Z}_p^*$ be an M-th primitive root of unity, and the multiplicative subgroup of roots of unity generated by ω is denoted by $\mathbb{H} = \{\omega, \omega^2, \ldots, \omega^M\}$. Let $L_i(X)$ be the i-th Lagrange basis polynomial on $\mathbb{H}$, namely the unique degree $M-1$ polynomial which satisfies $L_i(\omega^j)$ is 1 when $i = j$ and 0 when $i \neq j$. Let $Z(X) = \prod_{i \in [M]} (X - \omega^i)$ be the vanishing polynomial on $\mathbb{H}$. Since $\mathbb{H}$ is a multiplicative subgroup, we have $Z(X) = X^M - 1$, and $L_i(X) = \frac{\omega^i}{M} \cdot \frac{X^M - 1}{X - \omega^i}$ with $L_i(0) = \frac{1}{M}$ for $i \in [M]$. Sometimes we also write ω^M as ω^0 and L_M as L_0 since they are equivalent.

For any vector $B = (b_1, b_2, \ldots, b_M)$[1], let $B(X) = \sum_{i \in [M]} b_i \cdot L_i(X)$ be the unique polynomial that satisfies $\deg(B) \leq M-1$ and $B(\omega^i) = b_i$ for all $i \in [M]$.

2.5 Polynomial Commitment

We use the KZG polynomial commitment [31] as the vector commitment scheme. Given a degree bound d, the common reference string derived by a trapdoor τ is denoted by $\mathsf{CRS} = ([1], [\tau], [\tau^2], \ldots, [\tau^d])$. For any polynomial $B(X)$ that satisfies $\deg(B) \leq d$, its commitment is $[B(\tau)]$.

2.6 Generalized Sumcheck

Our construction relies on the following lemma, known as generalized sumcheck, whose proof can be found in [37].

[1]If B has length $l_B < M$, we consider that $b_i = 0$ for all $i = l_B + 1, \ldots, M$.

Lemma 2.1 (Generalized Sumcheck). *Let* $A(X) = \sum_{i=1}^{|\mathbb{H}|} a_i \cdot L_i(X)$, $B(X) = \sum_{i=1}^{|\mathbb{H}|} b_i \cdot L_i(X)$, *then*

$$A(X) \cdot B(X) = \frac{\sum_i a_i b_i}{|\mathbb{H}|} + Q_x(X) \cdot X + Q_Z(X) \cdot Z(X)$$

where Q_x, Q_Z *are polynomials with degree* $\leq |\mathbb{H}| - 2$ *defined as*

$$Q_x(X) = \sum_i a_i b_i \frac{L_i(X) - L_i(0)}{X}$$

$$Q_Z(X) = \sum_i a_i b_i \frac{L_i^2(X) - L_i(X)}{Z(X)} + \sum_{i \neq j} a_i b_j \frac{L_i(X) L_j(X)}{Z(X)}$$

3 REVISITING THE DEFINITION OF C-STS

To better demonstrate our construction, we first revisit the definition of c-STS in [30] and introduce several adjustments in this section. These adjustments include:

- The maximum number of parties in the system N is added as an input to the Setup algorithm since it determines the size of CRS.
- We replace the $\mathsf{ComSelect}$ algorithm from [30] with a new abstraction, $\mathsf{EpochRepr}$, to decouple our cryptographic construction from the underlying committee election mechanism. Instead of relying on a trusted third party, we define $\mathsf{EpochRepr}$ as a deterministic, publicly computable function that extracts the mathematical representation of the committee (e.g., the polynomial commitment $[B_{\mathsf{Com}_e}(\tau)]_2$) directly from the agreed-upon on-chain state at the start of epoch e. Cryptographic unforgeability and confidentiality require $|A \cap \mathsf{Com}_e| < t$, where A is the corrupted-party set. Completing signing or decryption using only honest parties additionally requires at least t available honest members with accepted registrations. For a randomly sampled committee, suitable system parameters and a secure election mechanism (e.g., a randomness beacon [30]) must make violations of these respective conditions sufficiently unlikely; committee sampling does not guarantee them unconditionally. Vulnerabilities in the selection mechanism itself are orthogonal to our cryptographic design. Under this abstraction, $\mathsf{EpochRepr}$ is computed from the agreed-upon committee and is publicly available to Verify (and Enc of Definition 7.1). The related discussion can be found in Section 6.2.
- We add additional efficiency requirements to $\mathsf{EpochRepr}$ and Aggr. Considering the important role of $\mathsf{EpochRepr}$, it is natural to set the formal requirement, while the latter is based on our goal of constructing practical schemes as described in Section 1.

Definition 3.1 (c-STS, revisited). A silent threshold signature scheme for dynamic committees consists of a tuple of algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{HintGen}, \mathsf{Preprocess}, \mathsf{EpochRepr}, \mathsf{Sign}, \mathsf{PartVerify}, \mathsf{Aggr}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, N)$: On input the security parameter λ and an upper bound N on the maximum number of users, the Setup algorithm outputs a common reference string CRS.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$: On input the security parameter λ, the KeyGen algorithm outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{hint}_i \leftarrow \mathsf{HintGen}(\mathsf{CRS}, sk, N, i)$: On input the CRS, the secret key sk, the number of parties N, and a position $i \in [N]$, the HintGen algorithm outputs a hint hint_i.
- $(\mathsf{ak}, \mathsf{vk}) \leftarrow \mathsf{Preprocess}(\mathsf{CRS}, \{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]})$: On input the CRS, all pairs $\{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]}$, the Preprocess algorithm outputs an aggregation key ak and a verification key vk.
- $B_e \leftarrow \mathsf{EpochRepr}(e)$: On input epoch e, the EpochRepr algorithm outputs a mathematical representation B_e of the epoch e's committee $\mathsf{Com}_e \subseteq [N]$.
- $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m, e)$: On input a secret key sk_i, a message m, and epoch e, the Sign algorithm outputs a partial signature σ_i.
- $1/0 \leftarrow \mathsf{PartVerify}(\mathsf{pk}_i, \sigma_i, m, e)$: On input a message m, a partial signature σ_i, a public key pk_i, and epoch e, the PartVerify algorithm outputs 1 iff the partial signature verifies and corresponds to epoch e.
- $\sigma \leftarrow \mathsf{Aggr}(\mathsf{CRS}, \mathsf{ak}, m, \{\sigma_i\}_{i\in S}, t, e)$: On input the CRS, the aggregation key ak, a message m, a set of partial signatures $\{\sigma_i\}_{i\in S}$, a threshold t, and epoch e, the Aggr algorithm outputs an aggregated signature σ.
- $1/0 \leftarrow \mathsf{Verify}(\mathsf{vk}, m, \sigma, t, e)$: On input the verification key vk, a message m, an aggregated signature σ, a threshold t, and epoch e, the Verify algorithm outputs 1 iff σ is verified.

Moreover, the c-STS scheme must have the following efficiency requirements:

- The size of the verification key vk and the aggregated signature σ should be $O(1)$.
- For epoch e with the committee size $n = |\mathsf{Com}_e|$, the complexity of EpochRepr should be $O(n \cdot \mathrm{polylog}(n))$.
- Given the commitment of a committee B_e, the complexity of Verify should be $O(1)$, and the complexity of Aggr should be $O(n \cdot \mathrm{polylog}(n))$.

Based on the modification above, we also revisit the correctness and unforgeability properties of c-STS. We defer the formal definition to Appendix A.

4 LINEARLY VERIFIABLE C-STS SCHEME

In this section, we first recall the c-STS scheme Dyna-hinTS proposed in [30], and discuss how to use the linear verification equations to replace the Plonk-style weight proofs.

4.1 Revisiting Dyna-hinTS

Dyna-hinTS is a silent threshold signature scheme for dynamic committees based on hinTS [26], but the authorized committee is a subset of the entire set of parties that changes dynamically over epochs. Assume there are N parties in the system and $|\mathbb{H}| = N + 1$. During the silent setup, each party will sample an independent BLS key pair $\{\mathsf{pk}_i = [\mathsf{sk}_i]_1, \mathsf{sk}_i\}$. At epoch e, the authorized committee Com_e is randomly chosen[2]. Given a message m to sign, parties compute an epoch-specified[3] $\mathsf{tag} = \mathcal{H}_2(m, e)$ and sign it using the BLS signature by $\sigma_i = \mathsf{sk}_i \cdot \mathsf{tag}$. Suppose a subset $B \subseteq [N]$ of parties have signed tag; the aggregator will proceed to aggregate the partial signatures as follows. It first shall aggregate the partial signatures $\{\sigma_i\}_{i\in B}$ and public keys $\{\mathsf{pk}_i\}_{i\in B}$ as σ^* and aPK. Furthermore, to prove the honest aggregation of aPK, the aggregator should commit to B and perform the following computation:

1. It computes proof $\pi_1 = ([Q_Z(\tau)]_1, [Q_x(\tau)]_1, [\hat{Q}_x(\tau)]_1)$ which relies on the generalized sumcheck protocol to prove that the inner product of public keys and B is the claimed aggregate public key aPK:
$$\begin{aligned}[SK(\tau)]_1 \circ [B(\tau)]_2 &= [1]_2 \circ \mathsf{aPK} + [Z(\tau)]_2 \circ [Q_Z(\tau)]_1 \\ &\quad + [\tau]_2 \circ [Q_x(\tau)]_1 \\ [\tau]_2 \circ [Q_x(\tau)]_1 &= [1]_2 \circ [\hat{Q}_x(\tau)]_1\end{aligned}$$
where $SK(X) = \sum_{i\in[N]} \mathsf{sk}_i \cdot L_i(X)$ and $B(X) = \sum_{i\in[N]} b_i \cdot L_i(X)$. The first equation checks that the polynomial identity holds at the random point $x = \tau$, and the second equation constrains the maximum degree of $Q_x(X)$ as required in the sumcheck protocol.
2. It computes a Plonk-style proof π_2 to demonstrate that (1) B is a binary vector, (2) the inner product between B and $(1, 1, \ldots, 1)$ equals t, and (3) B is a subset of the authorized committee Com_e. In particular, (3) is achieved in the same way as (1) and (2): By proving that $\bar{B} = \mathsf{Com}_e \setminus B$ is also a binary vector and the inner product between $\bar{B}$ and $(1, 1, \ldots, 1)$ equals $|\mathsf{Com}_e| - t$, it is sufficient to show that $B \subseteq \mathsf{Com}_e$ (notice that given the commitment $[B_{\mathsf{Com}_e}(\tau)]_2$ of Com_e, we have $[\bar{B}(\tau)]_2 = [B_{\mathsf{Com}_e}(\tau)]_2 - [B(\tau)]_2$).

The final signature is $(\sigma^*, \mathsf{aPK}, [B(\tau)]_2, \pi_1, \pi_2)$. For verification, one verifies π_1, π_2 and verifies the aggregated signature σ^* by checking $\mathsf{tag} \circ \mathsf{aPK} = [1]_1 \circ \sigma^*$.

4.2 Removing Plonk-style Proofs

Since the computation of $[B(\tau)]_2, \mathsf{aPK}, [Q_Z(\tau)]_1, [Q_x(\tau)]_1, [\hat{Q}_x(\tau)]_1$ only relates to $i \in B$, it has complexity $O(n)$. However, generating Plonk-style proofs for proving B and $\bar{B}$ have sufficient weight requires $O(N \log N)$ (below we use B, and the same also applies for $\bar{B}$) because it contains the following steps (slightly adapted and rephrased here for consistency with our notation; note that $L_{N+1} = L_0$):

- Compute the partial sum vector $parsum$ defined by
$$\begin{aligned}parsum_1 &= 0 \\ parsum_{i+1} &= parsum_i + b_i \qquad i = 1, \ldots, N,\end{aligned}$$
the corresponding polynomial
$$Parsum(X) = \sum_{i=1}^{N+1} parsum_i \cdot L_i(X)$$
and its commitment $[Parsum(\tau)]_1$.

[2]Here we treat Com_e as a binary vector $\mathsf{Com}_e = \{b_1, \ldots, b_N\} \in \{0, 1\}^N$ where $b_i = 1$ if and only if $i \in \mathsf{Com}_e$. The same applies to B and $\bar{B}$, which we describe later.

[3]Embedding epoch e into the signature prevents the signature from being used for other epochs.

- To prove the correctness of $[Parsum(\tau)]_1$ and $[B(\tau)]_2$, i.e.,

$$\begin{aligned} Parsum(X) &= 0 && X = \omega \\ Parsum(\omega X) - Parsum(X) - B(X) &= 0 && X \in \mathbb{H} \setminus \{\omega^{N+1}\} \\ Parsum(X) &= t && X = \omega^{N+1} \\ B(X) \cdot (1 - B(X)) &= 0 && X \in \mathbb{H}, \end{aligned}$$

  compute the corresponding quotient polynomial $Q_i(X)$ for $i = 1, 2, 3, 4$ where

$$\begin{aligned} L_1(X) \cdot Parsum(X) &= Q_1(X) \cdot Z(X) \\ (X - \omega^{N+1})(Parsum(\omega X) - Parsum(X) - B(X)) &= Q_2(X) \cdot Z(X) \\ L_{N+1}(X) \cdot (Parsum(X) - t) &= Q_3(X) \cdot Z(X) \\ B(X) \cdot (1 - B(X)) &= Q_4(X) \cdot Z(X) \end{aligned}$$

  and their (linearly combined) commitment

$$[Q(\tau)]_1 = \Big[\sum_{i=1}^{4} Q_i(\tau) \cdot v^{i-1} \Big]_1$$

  for some random challenge v.
- For some random challenge r, compute $B(r), Q(r), Parsum(r), Parsum(\omega r)$ with the corresponding KZG opening proofs.

Therefore, to reduce the complexity of aggregating signatures, we must avoid generating these proofs and use alternative methods to provide a proof of weight. [27] proposed a technique to achieve this goal: They compute $B(X)$ by constructing a polynomial vanishing at the non-participating positions $\{(\omega^i, 0)\}_{i \in [N] \setminus B}$, and a valid B will satisfy $\deg(B) \leq N - t$ which can be checked linearly via $[\tau^t]_1 \circ [B(\tau)]_2 = [1]_2 \circ [\hat{B}(\tau)]_1$ with an additional commitment of $\hat{B}(X) = B(X) \cdot X^t$. Intuitively, the degree bound ensures at least t nonzero participation coefficients, so aPK combines at least t public keys even when B is not binary. This intuition alone does not establish security for maliciously related registered keys or adversarially chosen coefficients. The formal unforgeability argument, including malicious registrations and adversarially chosen aggregation coefficients, is given in Appendix B.7.

However, this trick cannot be used directly to replace π_2: π_2 proves that the weight of B (resp. $\bar{B}$) **equals** t (resp. $n - t$), and the degree check is limited to proving the weight $\geq t$. Even if it is used for proving the weight of $B \geq t$, the corresponding statement for $\bar{B}$ becomes "the weight of $\bar{B} \leq n - t$" (not $\geq$), which is also beyond the ability of the degree check.

Our key observation for addressing this technical challenge is that **the vector Com_e does not have to be binary either**. One possible approach is to format $B_{\mathsf{Com}_e}(X)$ the same as $B(X)$, i.e., to compute $B_{\mathsf{Com}_e}(X)$ by constructing a polynomial vanishing at the non-participating positions $\{(\omega^i, 0)\}_{i \in [N] \setminus \mathsf{Com}_e}$. In this way, "$B \subseteq \mathsf{Com}_e$" is equivalent to "the zero set of $B_{\mathsf{Com}_e}(X)$ is a subset of the zeros of $B(X)$", and is further equivalent to the existence of the quotient polynomial $\bar{B}(X) = \frac{B(X)}{B_{\mathsf{Com}_e}(X)}$. Therefore, we obtain the following equation for the subset check:

$$[1]_1 \circ [B(\tau)]_2 = [B_{\mathsf{Com}_e}(\tau)]_2 \circ [\bar{B}(\tau)]_1$$

The aggregator can prove $B \subseteq \mathsf{Com}_e$ by providing the commitment $[\bar{B}(\tau)]_1$ of the quotient polynomial. This also allows us to verify the weight of B via the degree check, where the aggregator should provide the degree proof $[\hat{B}(\tau)]_1$:

$$[\tau^t]_1 \circ [B(\tau)]_2 = [1]_2 \circ [\hat{B}(\tau)]_1 \tag{1}$$

Finally, to ensure that $B(X)$ is non-zero, a dummy party P_0 is always required in B (as was done in [27]), which can also be checked by a KZG opening equation

$$[1]_1 \circ \boxed{[B(\tau)]_2} = [\tau - 1]_2 \circ \boxed{[Q_0(\tau)]_1} + [1]_T.$$

Notice that the minimum weight of B should be $t + 1$ after introducing P_0 to B, so we have $\hat{B}(X) = X^{t+1}B(X)$, and the first pairing input of Equation 1 should also be replaced by $[\tau^{t+1}]_1$.

So far, we have replaced the Plonk-style weight proofs by introducing a new representation of Com_e and new polynomials with their commitments. Now the aggregated signature $\boldsymbol{w}$ consists of

$$[B(\tau)]_2, \mathsf{aPK}, [Q_Z(\tau)]_1, [Q_x(\tau)]_1, [\hat{Q}_x(\tau)]_1, \sigma^*, [\hat{B}(\tau)]_1, [\bar{B}(\tau)]_1, [Q_0(\tau)]_1$$

which should satisfy the verification that consists of the following six pairing equations (the group elements from the signature are shaded, and the rest are public group elements):

$$\begin{aligned} [SK(\tau)]_1 \circ \boxed{[B(\tau)]_2} &= [1]_2 \circ \boxed{\mathsf{aPK}} + [Z(\tau)]_2 \circ \boxed{[Q_Z(\tau)]_1} \\ &\quad + [\tau]_2 \circ \boxed{[Q_x(\tau)]_1} \\ [\tau]_2 \circ \boxed{[Q_x(\tau)]_1} &= [1]_2 \circ \boxed{[\hat{Q}_x(\tau)]_1} \\ \mathsf{tag} \circ \boxed{\mathsf{aPK}} &= [1]_1 \circ \boxed{\sigma^*} \\ [\tau^{t+1}]_1 \circ \boxed{[B(\tau)]_2} &= [1]_2 \circ \boxed{[\hat{B}(\tau)]_1} \\ [1]_1 \circ \boxed{[B(\tau)]_2} &= [B_{\mathsf{Com}_e}(\tau)]_2 \circ \boxed{[\bar{B}(\tau)]_1} \\ [1]_1 \circ \boxed{[B(\tau)]_2} &= [\tau - 1]_2 \circ \boxed{[Q_0(\tau)]_1} + [1]_T \end{aligned} \tag{2}$$

4.3 Our Construction

Now we give our construction based on the above discussion. Let N be the maximum size of the entire set of parties. Without loss of generality, we set $|\mathbb{H}| = N + 1$ where the index 0 is reserved for a dummy party with $\mathsf{sk}_0 = 1$ and $\mathsf{pk}_0 = [1]_1$.

We use the Eq-DL proof in Section 2.3 as a knowledge proof for sk at two different steps:

- In $\mathsf{HintGen}$, prove knowledge of one scalar k satisfying both $\mathsf{pk}_i = k[1]_1$ and $h_1 = k[L_i(\tau)]_1$, where h_1 is the first hint component, and check the proof in $\mathsf{HintVerify}$. This binds the hint relation to the same scalar as the public key. Without this proof, a malicious member can construct a hint from an existing $\mathbb{G}_1$ element (e.g., another member's public key or an element related to CRS) without knowing the corresponding scalar and pass the original hint-consistency checks.
- In Sign, prove knowledge of one scalar k satisfying both $\mathsf{pk}_i = k[1]_1$ and $s_i = k\mathsf{tag}$, where s_i is the BLS share in σ_i, and check the proof in $\mathsf{PartVerify}$. It certifies the same BLS verification relation, with negligible soundness error in the random-oracle model, while replacing pairings in partial verification with group operations.

Construction 1 (Linearly Verifiable c-STS). *We specify each algorithm of linearly verifiable c-STS as follows.*

- $\mathsf{Setup}(1^\lambda, N)$: *Sample* $\tau \leftarrow_\$ \mathbb{Z}_p$ *and output*

$$\mathsf{CRS} = ([1]_1, [\tau^1]_1, \ldots, [\tau^{N+1}]_1, [1]_2, [\tau^1]_2, \ldots, [\tau^{N+1}]_2).$$

- $\mathsf{KeyGen}(1^\lambda)$: *Sample* $x \leftarrow_\$ \mathbb{Z}_p$ *and output* $\mathsf{pk} = [x]_1, \mathsf{sk} = x$.
- $\mathsf{HintGen}(\mathsf{CRS}, \mathsf{sk}_i, N, i)$: *Output*

$$\mathsf{hint}_i = \Big([\mathsf{sk}_i L_i(\tau)]_1, [\mathsf{sk}_i(L_i(\tau) - L_i(0))]_1, \Big[\mathsf{sk}_i \frac{L_i^2(\tau) - L_i(\tau)}{Z(\tau)}\Big]_1, \Big[\mathsf{sk}_i \frac{L_i(\tau) - L_i(0)}{\tau}\Big]_1, \Big\{\Big[\mathsf{sk}_i \frac{L_i(\tau)L_j(\tau)}{Z(\tau)}\Big]_1\Big\}_{j\in[N]', j\neq i}, \pi_i^{\mathsf{hint}}\Big),$$

where the Eq-DL proof

$$\pi_i^{\mathsf{hint}} = \mathsf{EqProve}([1]_1, [\mathsf{sk}_i]_1, [L_i(\tau)]_1, [\mathsf{sk}_i L_i(\tau)]_1; \mathsf{sk}_i).$$

- $\mathsf{Preprocess}(\mathsf{CRS}, \{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]})$: *Verify each hint and let* $V \subseteq [N]$ *be the set of indices with valid hints. The dummy party is included unconditionally, and has* $\mathsf{sk}_0 = 1, \mathsf{pk}_0 = [1]_1$ *with the corresponding (publicly computable) hint. Set* $\mathsf{sk}_i = 0$ *for* $i \notin V'$. *Output*

$$\mathsf{ak} = \Big(V, \{\mathsf{pk}_i\}_{i\in V'}, \{\mathsf{hint}_i^{(2)}\}_{i\in V'}, \{\mathsf{hint}_i^{(3)}\}_{i\in V'}, \{\mathsf{hint}_i^{(4)}\}_{i\in V'}, \Big\{\sum_{j\in V', j\neq i} \mathsf{hint}_{j,i}^{(5)}\Big\}_{i\in V'}\Big),$$

$$\mathsf{vk} = \Big(\sum_{i\in V'} \mathsf{hint}_i^{(1)}, [Z(\tau)]_2\Big) = (C, Z).$$

- $\mathsf{EpochRepr}(e)$: *Obtain committee* $\mathsf{Com}_e \subseteq [N]$ *of epoch* e, *compute* $B_{\mathsf{Com}_e}(X)$ *by constructing the normalized vanishing polynomial*

$$B_{\mathsf{Com}_e}(X) = \prod_{i\notin \mathsf{Com}_e'} \frac{X - \omega^i}{1 - \omega^i},$$

and output $B_e = [B_{\mathsf{Com}_e}(\tau)]_2$.

- $\mathsf{Sign}(\mathsf{sk}_i, m, e)$: *Let* $\mathsf{tag} = \mathcal{H}_2(m, e)$ *and output* $\sigma_i = (s_i, \pi_i^{\mathsf{sign}})$, *where*

$$s_i = \mathsf{sk}_i \cdot \mathsf{tag},$$
$$\pi_i^{\mathsf{sign}} = \mathsf{EqProve}([1]_1, [\mathsf{sk}_i]_1, \mathsf{tag}, s_i; \mathsf{sk}_i).$$

- $\mathsf{PartVerify}(\mathsf{pk}_i, \sigma_i, m, e)$: *Parse* $\sigma_i = (s_i, \pi_i^{\mathsf{sign}})$. *Let* $\mathsf{tag} = \mathcal{H}_2(m, e)$ *and output 1 iff* $\mathsf{EqVrfy}([1]_1, \mathsf{pk}_i, \mathsf{tag}, s_i; \pi_i^{\mathsf{sign}})$.
- $\mathsf{Aggr}(\mathsf{CRS}, \mathsf{ak}, m, \{\sigma_i\}_{i\in S}, t, e)$: *Let* $\mathsf{tag} = \mathcal{H}_2(m, e)$. *Parse each* $\sigma_i = (s_i, \pi_i^{\mathsf{sign}})$. *Discard signatures from those whose hints did not verify (under* $\mathsf{HintVerify}$*), whose signatures did not verify (under* $\mathsf{PartVerify}$*), and who do not belong to* Com_e. *Let* S *be the set of indices of the remaining signers, and return* $\perp$ *unless* $1 \le t \le |\mathsf{Com}_e|$ *and* $|S| \ge t$. *Proceed as follows:*
  1. *Compute* $B(X)$ *by constructing the normalized vanishing polynomial*

$$B(X) = \prod_{i\notin S'} \frac{X - \omega^i}{1 - \omega^i},$$

  and compute $B = [\sum_{i=0}^N B(\omega^i) L_i(\tau)]_2$.
  2. *Compute* $\mathsf{aPK} = \frac{1}{N+1}(\sum_{i\in S} B(\omega^i)\mathsf{pk}_i + [1]_1)$.
  3. *Compute*

$$Q_Z = [Q_Z(\tau)]_1 = \sum_{i\in S'} B(\omega^i) \cdot (\mathsf{ak}_i^{(4)} + \mathsf{ak}_i^{(6)}),$$
$$Q_x = [Q_x(\tau)]_1 = \sum_{i\in S'} B(\omega^i) \cdot \mathsf{ak}_i^{(5)},$$
$$\hat{Q}_x = [Q_x(\tau)\cdot\tau]_1 = \sum_{i\in S'} B(\omega^i) \cdot \mathsf{ak}_i^{(3)}.$$

  4. *Compute* $\sigma^* = \frac{1}{N+1}(\sum_{i\in S} B(\omega^i) s_i + \mathsf{tag})$.
  5. *Compute* $\hat{B}(X) = X^{t+1}B(X)$ *and set* $\hat{B} = [\hat{B}(\tau)]_1$.
  6. *Compute* $\bar{B}(X) = \prod_{i\in \mathsf{Com}_e\setminus S} \frac{X-\omega^i}{1-\omega^i}$ *and set* $\bar{B} = [\bar{B}(\tau)]_1$.
  7. *Compute* $Q_0(X) = \frac{B(X)-1}{X-1}$ *and set* $Q_0 = [Q_0(\tau)]_1$.
  8. *Output* $\sigma = (B, -\mathsf{aPK}, -Q_Z, -Q_x, \hat{Q}_x, \sigma^*, -\hat{B}, -\bar{B}, -Q_0)^\top$.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma, t, e)$: *If* $1 \le t \le |\mathsf{Com}_e|$ *does not hold, output 0. Otherwise, let* $\mathsf{tag} = \mathcal{H}_2(m, e)$ *and set* $Z_0 = [\tau - 1]_2$, *compute*

$$\boldsymbol{A} = \begin{pmatrix} C & [1]_2 & Z & [\tau]_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & [\tau]_2 & [1]_2 & 0 & 0 & 0 & 0 \\ 0 & \mathsf{tag} & 0 & 0 & 0 & [1]_1 & 0 & 0 & 0 \\ [\tau^{t+1}]_1 & 0 & 0 & 0 & 0 & 0 & [1]_2 & 0 & 0 \\ [1]_1 & 0 & 0 & 0 & 0 & 0 & 0 & B_e & 0 \\ [1]_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Z_0 \end{pmatrix}$$

$$\boldsymbol{b} = ([0]_T, [0]_T, [0]_T, [0]_T, [0]_T, [1]_T)^\top$$

and output 1 iff $\boldsymbol{A} \circ \sigma = \boldsymbol{b}$. *This corresponds to Equation 2 since it is fully linear, i.e., each equation only consists of pairing computation, while each pairing has at least one public input (we use colors to demonstrate this).*

To verify the well-formedness of each hint, we define the $\mathsf{HintVerify}$ algorithm as follows.

$\mathsf{HintVerify}(\mathsf{CRS}, i, \mathsf{hint}_i, \mathsf{pk}_i) \to 1/0$: Parse $\mathsf{hint}_i = (h_1, h_2, h_3, h_4, \{h_5^{(j)}\}_{j\in[N]', j\neq i}, \pi_i^{\mathsf{hint}})$. Output 1 iff $\mathsf{EqVrfy}([1]_1, \mathsf{pk}_i, [L_i(\tau)]_1, h_1; \pi_i^{\mathsf{hint}}) = 1$ and

$$\begin{aligned} h_1 \circ [1]_2 &= \mathsf{pk}_i \circ [L_i(\tau)]_2 \\ h_2 \circ [1]_2 &= \mathsf{pk}_i \circ [L_i(\tau) - L_i(0)]_2 \\ h_3 \circ [1]_2 &= \mathsf{pk}_i \circ \Big[\frac{L_i^2(\tau) - L_i(\tau)}{Z(\tau)}\Big]_2 \\ h_4 \circ [1]_2 &= \mathsf{pk}_i \circ \Big[\frac{L_i(\tau) - L_i(0)}{\tau}\Big]_2 \\ h_{5,j} \circ [1]_2 &= \mathsf{pk}_i \circ \Big[\frac{L_i(\tau)L_j(\tau)}{Z(\tau)}\Big]_2, \forall j \in [N]' \wedge j \neq i \end{aligned} \tag{3}$$

Remark 1. *We assume that the* Setup *algorithm also outputs the commitments of the Lagrange basis polynomials* $[L_i(\tau)]_{1,2}$. *This is because almost all steps of our scheme use these values, and preprocessing them can significantly reduce the related computational cost. Since these values can be easily computed and publicly verified by* CRS, *this assumption will not affect the security of our scheme.*

5 MAKING C-STS PRACTICAL

In this section, we analyze the efficiency of Construction 1. We first review the time complexities of several fundamental polynomial operations:

- Given a polynomial $f(X)$ with $f(0) \neq 0$, its multiplicative inverse modulo X^n can be computed in $O(n \log n)$ time by Newton's iteration and the Fast Fourier Transform (FFT).
- For a polynomial $f(X)$ of degree n, evaluating $f(X)$ at n arbitrary distinct points requires $O(n \log^2 n)$ time by a divide-and-conquer approach. Alternatively, if the evaluation points are the N-th roots of unity ($N \geq n$), the evaluation can be computed in $O(N \log N)$ time by FFT.
- Let ω be an N-th root of unity and $S \subseteq [N]$ with $|S| = n$. Computing the coefficients of $f_S(X) = \prod_{i \in S}(X - \omega^i)$ by divide-and-conquer requires $O(n \log^2 n)$ time. However, $f_S(X)$ can also be obtained by computing the inverse of $-\frac{Z(X)}{f_S(X)}$ modulo X^{n+1} when $n < N$, which takes $O((N - n) \log^2(N - n) + n \log n)$ time and is asymptotically faster when $N - n \ll n$.

Let $n = |\mathsf{Com}_e|$ be the size of the epoch e's committee and $s = |S|$ be the number of valid partial signatures; then for any valid S we have $t \leq s \leq n$. Under the assumption that $N \gg n$, the efficiency of Construction 1 can be analyzed as follows:

- The complexity of $\mathsf{EpochRepr}$ is $O(N \log N)$, since obtaining $[B_{\mathsf{Com}_e}(\tau)]_2$ requires to compute

$$B_{\mathsf{Com}_e}(X) = \prod_{i \in [N] \setminus \mathsf{Com}_e} \frac{X - \omega^i}{1 - \omega^i},$$

  which has complexity $O(N \log N)$ when $N \gg n$.
- The complexity of Aggr is also $O(N \log N)$:
  1. Computing $B(X)$ has complexity $O(N \log N)$ (similar to computing $B_{\mathsf{Com}_e}(X)$).
  2. Evaluating $B(\omega^i)_{i \in S}$ from $B(X)$ has complexity $O(N \log N)$ since $\deg(B) = N - s$.
  3. Computing $[B(\tau)]_2 = [\sum_{i=0}^{N} B(\omega^i) L_i(\tau)]_2$ is $O(n)$ since there are only $O(n)$ non-zero $B(\omega^i)$. Similarly, the complexity of computing $aPK, \sigma^*, [Q_Z(\tau)]_1, [Q_x(\tau)]_1, [\hat{Q}_x(\tau)]_1$ is also $O(n)$.
  4. Since $\hat{B}(X) = X^{t+1} B(X)$, $[\hat{B}(\tau)]_1$ can be computed directly from the coefficients of $B(X)$, which has complexity $O(N)$.
  5. Computing $\bar{B}(X)$ has complexity $O(n \log^2 n)$, and computing $[\bar{B}(\tau)]_1$ has complexity $O(n)$.
  6. Computing the coefficients of $Q_0(X)$ and $[Q_0(\tau)]_1$ has complexity $O(N)$.

The analysis above demonstrates that although we have successfully removed the Plonk-style proofs in Dyna-hinTS, the efficiency of Construction 1 cannot meet our requirements in large-scale systems, where the global system size is much larger than that of each epoch's committee. Therefore, we propose improvements to Construction 1 to make it practical even if N is very large. Specifically, we reduce the complexity of $\mathsf{EpochRepr}$ and Aggr from $O(N \log N)$ to $O(n \log^2 n)$, which significantly improves their efficiency when $N \gg n$. We also reduce the complexity of $\mathsf{HintGen}$ from $O(N^2)$ to $O(N)$, which is of independent interest. We describe the details of all optimizations below.

5.1 Improving the Epoch Commitment

Recall the equation of the subset check

$$[1]_1 \circ [B(\tau)]_2 = [B_{\mathsf{Com}_e}(\tau)]_2 \circ [\bar{B}(\tau)]_1 \tag{4}$$

Since $B_{\mathsf{Com}_e}(X)$ has $N-n$ zeros, we must compute it in $O(N \log N)$. In contrast, its "complementary" polynomial $C_{\mathsf{Com}_e}(X)$ with respect to the vanishing polynomial $Z(X)$, i.e.,

$$C_{\mathsf{Com}_e}(X) = \prod_{i \in \mathsf{Com}'_e} (X - \omega^i),$$

has only $n + 1$ zeros and can be computed in $O(n \log^2 n)$. Thus, multiplying $C_{\mathsf{Com}_e}(\tau)$ on both sides of Equation 4 we have

$$[C_{\mathsf{Com}_e}(\tau)]_1 \circ [B(\tau)]_2 = [k \cdot Z(\tau)]_2 \circ [\bar{B}(\tau)]_1 \tag{5}$$

where k is some constant value (we compute it below) and can be moved into the computation of $[\bar{B}(\tau)]_1$. Notice that Equation 5 is still linear because the commitment of the vanishing polynomial $[Z(\tau)]_2$ is public. This allows us to replace the output of $\mathsf{EpochRepr}(e)$ by $[C_{\mathsf{Com}_e}(\tau)]_1$ without compromising the security properties of the scheme. The modified $\bar{B}(X)$ can be computed as follows and remains $O(n \log^2 n)$:

$$\begin{aligned}\bar{B}(X) &= \frac{C_{\mathsf{Com}_e}(X) \cdot B(X)}{Z(X)} = \frac{\prod_{i \in \mathsf{Com}'_e}(X - \omega^i) \cdot \prod_{i \notin S, i \neq 0} \frac{X - \omega^i}{1 - \omega^i}}{\prod_i (X - \omega^i)} \\ &= \frac{\prod_{i \in \mathsf{Com}_e \setminus S}(X - \omega^i)}{\prod_{i \notin S, i \neq 0}(1 - \omega^i)} = \prod_{i \in \mathsf{Com}_e \setminus S} (X - \omega^i) \cdot \frac{\prod_{i \in S}(1 - \omega^i)}{N + 1}\end{aligned}$$

5.2 Improving Signature Aggregation

There are three steps with complexity $\Omega(N)$ in the algorithm Aggr of Section 4, namely computing B, $\hat{B}$, and Q_0. We optimize these steps separately as follows.

Computing B. Since the purpose of computing $B(X)$ is to obtain $\{B(\omega^i)\}_{i \in S}$, which has only $O(n)$ non-zero elements, it is possible to reduce the complexity to $O(n)$ if we can bypass interpolating $B(X)$, i.e., to compute $\{B(\omega^i)\}_{i \in S}$ without computing $B(X)$ itself.

Let $f_k(X) = \prod_{i \neq k}(X - \omega^i) = \frac{X^{N+1} - 1}{X - \omega^k}$, then by $L_k(\omega^k) = 1$ we have $f_k(\omega^k) = \frac{N+1}{\omega^k}$. Since $B(X) = \prod_{i \notin S, i \neq 0} \frac{X - \omega^i}{1 - \omega^i}$, for each $\{\omega^k\}_{k \in S}$ we want to compute

$$B(\omega^k) = \prod_{i \notin S, i \neq 0} \frac{\omega^k - \omega^i}{1 - \omega^i} = \frac{\prod_{i \neq 0, i \neq k} \frac{\omega^k - \omega^i}{1 - \omega^i}}{\prod_{i \in S, i \neq k} \frac{\omega^k - \omega^i}{1 - \omega^i}}.$$

For the numerator, we have

$$\begin{aligned}\prod_{i \neq 0, i \neq k} \frac{\omega^k - \omega^i}{1 - \omega^i} &= \frac{\prod_{i \neq 0, i \neq k}(\omega^k - \omega^i)}{\prod_{i \neq 0, i \neq k}(1 - \omega^i)} = \frac{f_k(\omega^k)/(\omega^k - 1)}{f_0(1)/(1 - \omega^k)} \\ &= \frac{N + 1}{\omega^k(\omega^k - 1)} \cdot \frac{1 - \omega^k}{N + 1} = -\omega^{-k}.\end{aligned}$$

For the denominator, let $g(X) = \prod_{i \in S}(X - \omega^i)$, then $g'(\omega^k) = \prod_{i \in S, i \neq k}(\omega^k - \omega^i)$ and

$$\prod_{i \in S, i \neq k} \frac{\omega^k - \omega^i}{1 - \omega^i} = \frac{\prod_{i \in S, i \neq k}(\omega^k - \omega^i)}{\prod_{i \in S, i \neq k}(1 - \omega^i)} = (1 - \omega^k) \cdot \frac{g'(\omega^k)}{g(1)}.$$

In conclusion, we can compute $B(\omega^k)$ by

$$B(\omega^k) = \frac{g(1)}{\omega^k \cdot (\omega^k - 1) \cdot g'(\omega^k)}.$$

Since the complexity of computing $g(1)$ is $O(n)$, the complexity of computing and evaluating $g'(X)$ on S is $O(n \log^2 n)$, all $B(\omega^k)$ can be computed with complexity $O(n \log^2 n)$. Finally, using $\{B(\omega^k)\}_{k \in S}$ to compute $B = [\sum_{i \in S'} B(\omega^i) L_i(\tau)]_2$ has complexity $O(n)$.

Computing $\hat{B}$. $\hat{B}$ can be computed in a similar way to computing B since we already know that $\hat{B}(1) = 1, \hat{B}(\omega^i)_{i \notin S, i \neq 0} = 0$ and $\hat{B}(\omega^i)_{i \in S} = \omega^{i(t+1)} B(\omega^i)$. Since $\deg(\hat{B}) = (N - s) + (t + 1) \leq N$ when $s > t$, computing $\hat{B} = [\hat{B}(\tau)]_1 = [\sum_{i \in S'} \hat{B}(\omega^i) L_i(\tau)]_1$ has complexity $O(n)$.

The only exception is $\deg(\hat{B}) = N + 1$ when $s = t$, where $\hat{B}(X)$ cannot be directly represented by $L_i(X)$. Let $\hat{B}'(X) = \sum_{i \in S'} \hat{B}(\omega^i) L_i(X)$; then $\hat{B}(X) = \hat{B}'(X)$ for all ω^i, which implies $Z(X) | \hat{B}(X) - \hat{B}'(X)$. Since $\deg(\hat{B}') \leq N$ and $\deg(Z) = N + 1$, it must be

$$\hat{B}(X) = \sum_{i \in S'} \hat{B}(\omega^i) L_i(X) + k \cdot Z(X),$$

where k equals the leading coefficient of $B(X)$ and can be computed as

$$k = \prod_{i \notin S, i \neq 0} \frac{1}{1 - \omega^i} = \frac{g(1)}{N + 1}.$$

Therefore, computing

$$\hat{B} = [\hat{B}(\tau)]_1 = \Big[\sum_{i \in S'} \hat{B}(\omega^i) L_i(\tau) \Big]_1 + \frac{g(1)}{N + 1} \cdot [Z(\tau)]_1$$

also has complexity $O(n)$.

Computing Q_0. The method above cannot be used again for computing $Q_0 = [Q_0(\tau)]_1$: for every $i \notin S'$, the original quotient satisfies $Q_0(\omega^i) = -1/(\omega^i - 1) \neq 0$. Thus, when $s \ll N$, its evaluation vector is dense, with at least $N - s$ nonzero entries. The direct coefficient-based and evaluation-based commitment methods (i.e., compute $Q_0 = [Q_0(\tau)]_1 = [\sum_{i=0}^{N} Q_0(\omega^i) L_i(\tau)]_1$) considered here therefore require $\Omega(N)$ work.

To address this challenge, we should review the role of Q_0: it is the KZG opening proof of $B(\omega^0) = 1$, which ensures that the dummy party $\mathsf{P}_0 \in B$. Our approach here is to **regard the 1 on the right-hand side as the evaluation of some polynomial $h(X)$ at ω^0**, i.e., we prove $B(\omega^0) = h(\omega^0)$ for a polynomial $h(X)$ with $h(\omega^0) = 1$ publicly verifiable. This clearly does not affect the statement we need to prove, but by carefully choosing $h(X)$, the computation of $Q_0 = [Q_0(\tau)]_1$ may be significantly simplified. Since the vast majority of $\mathbb{H}$ are zeros of $B(X)$, we would like $h(X)$ to have the same property, and $h(X) = L_0(X)$ becomes the most natural choice. By defining

$$Q_0(X) = \frac{B(X) - L_0(X)}{X - 1}, \tag{6}$$

we have $Q_0(\omega^i) = 0$ for $i \notin S, i \neq 0$, i.e., there are only $O(n)$ nonzero $Q_0(\omega^i)$.

For $i \in S$, we know that $Q_0(\omega^i) = \frac{B(\omega^i)}{\omega^i - 1}$ which can be computed in $O(n)$, and for $Q_0(1)$ we have $Q_0(1) = B'(1) - L_0'(1)$. To compute $B'(1)$ and $L_0'(1)$, notice that for any $p(X) = \prod_i p_i(X)$ we have

$$p'(X) = p(X) \cdot \sum_i \frac{p_i'(X)}{p_i(X)}.$$

Since $B(X) = \prod_{i \notin S, i \neq 0} p_i(X)$ and $L_0(X) = \prod_{i \neq 0} p_i(X)$ where $p_i(X) = \frac{X - \omega^i}{1 - \omega^i}$ satisfies $p_i(1) = 1$ and $p_i'(1) = \frac{1}{1 - \omega^i}$, we have

$$B'(1) = B(1) \cdot \sum_{i \notin S, i \neq 0} \frac{p_i'(1)}{p_i(1)} = \sum_{i \notin S, i \neq 0} \frac{1}{1 - \omega^i}$$

$$L_0'(1) = L_0(1) \cdot \sum_{i \neq 0} \frac{p_i'(1)}{p_i(1)} = \sum_{i \neq 0} \frac{1}{1 - \omega^i}$$

which concludes that

$$Q_0(1) = \sum_{i \in S} \frac{1}{\omega^i - 1}$$

and can also be computed in $O(n)$. The result above allows us to compute $Q_0 = [\sum_{i \in S'} Q_0(\omega^i) L_i(\tau)]_1$ in $O(n)$ since there are only $O(n)$ non-zero $Q_0(\omega^i)$.

Finally, the verification equation corresponding to Equation 6 is

$$[1]_1 \circ \boxed{[B(\tau)]_2} = [\tau - 1]_2 \circ \boxed{[Q_0(\tau)]_1} + [L_0(\tau)]_1 \circ [1]_2,$$

which remains linear.

By applying the optimizations described above, our full signature verification equations can be summarized as

$$\begin{aligned}
[SK(\tau)]_1 \circ \boxed{[B(\tau)]_2} &= [1]_2 \circ \boxed{\mathsf{aPK}} + [Z(\tau)]_2 \circ \boxed{[Q_Z(\tau)]_1} \\
&\quad + [\tau]_2 \circ \boxed{[Q_x(\tau)]_1} \\
[\tau]_2 \circ \boxed{[Q_x(\tau)]_1} &= [1]_2 \circ \boxed{[\hat{Q}_x(\tau)]_1} \\
\mathsf{tag} \circ \boxed{\mathsf{aPK}} &= [1]_1 \circ \boxed{\sigma^*} \\
[\tau^{t+1}]_1 \circ \boxed{[B(\tau)]_2} &= [1]_2 \circ \boxed{[\hat{B}(\tau)]_1} \\
[C_{\mathsf{Com}_e}(\tau)]_1 \circ \boxed{[B(\tau)]_2} &= [Z(\tau)]_2 \circ \boxed{[\tilde{B}(\tau)]_1} \\
[1]_1 \circ \boxed{[B(\tau)]_2} &= [\tau - 1]_2 \circ \boxed{[Q_0(\tau)]_1} + [L_0(\tau)]_1 \circ [1]_2
\end{aligned} \tag{7}$$

while the aggregation has complexity $O(n \log^2 n)$.

5.3 Improving the Silent Setup

Although this work shares a common silent setup phase with [26, 27, 30], we notice that the optimizations to this phase are rarely mentioned. This is likely because the setup is one-time and trivially parallelizable; hence, its cost does not critically affect the usability of the underlying schemes.

Nevertheless, we provide some improvements to the silent setup phase in this section, which do not relate to specific threshold signatures or encryption schemes. Therefore, they can also be applied to [26, 27, 30] and may be of independent interest.

The silent setup phase in these schemes consists of the Setup, KeyGen, HintGen, Preprocess algorithms, among which we focus on the optimization of HintGen since the other algorithms are hard to optimize at the mathematical level. To generate hint_i, one must compute $H_i = (h_1, h_2, h_3, h_4, \{h_{5,j}\}_{j\in[N]',j\neq i})$ in advance, where

$$h_1 = [L_i(\tau)]_1, h_2 = [L_i(\tau) - L_i(0)]_1,$$
$$h_3 = \left[\frac{L_i^2(\tau) - L_i(\tau)}{Z(\tau)}\right]_1, h_4 = \left[\frac{L_i(\tau) - L_i(0)}{\tau}\right]_1,$$
$$h_{5,j} = \left[\frac{L_i(\tau)L_j(\tau)}{Z(\tau)}\right]_1, \forall j \in [N]' \wedge j \neq i.$$

In practice, each party may compute its own H_i by itself, or the aggregator may compute all of them and deliver each H_i to the i-th party. In any case, the i-th party multiplies its own sk_i with all the values in H_i and sends them to the aggregator. Therefore, we only focus on how to compute H_i efficiently below.

We denote the multi-scalar multiplication of k $\mathbb{G}_i$ $(i = 1, 2)$ points by $\mathrm{msm}_i(k)$. The naive approach computes each polynomial in H_i and its commitment:

- $h_1 = [L_i(\tau)]_1$ is already available. Since $[L_i(0)]_1 = [\frac{1}{N+1}]_1$ is a constant for all $i \in [N]'$, we also consider it publicly available, hence computing h_2 requires 1 $\mathbb{G}_1$ addition.
- Computing h_3 and h_4 requires to obtain the coefficients of $h_3(X)$ and $h_4(X)$ first, and $\mathrm{msm}_1(N)$ each for their commitments since they have degree $N-1$.
- Similarly, computing $h_{5,j}$ requires to obtain the coefficients of $h_{5,j}(X)$ first, and then $\mathrm{msm}_1(N)$. These steps apply to all $j \in [N]', j \neq i$.

In conclusion, the total complexity of computing H_i is $O(N^2)$, and can be reduced to $O(N^2/\log N)$ when applying a faster MSM algorithm (e.g., Pippenger's method [36]).

Our observation is that this complexity can be further reduced to $O(N)$. Specifically, we compute h_3, h_4, h_5 as follows:

- By $\sum_{j\in[N]'} L_j(X) = 1$, we have
$$\sum_{j\in[N]',j\neq i} \frac{L_i(X)L_j(X)}{Z(X)} + \frac{L_i^2(X)}{Z(X)} = \frac{L_i(X)}{Z(X)},$$
namely $h_3(X) = -\sum h_{5,j}(X)$. Therefore, we can compute $h_3 = -\sum h_{5,j}$ by $(N-1)\mathbb{G}_1$ additions after we obtain h_5.
- For $h_4(X)$, we have
$$\begin{aligned}\frac{L_i(X) - L_i(0)}{X} &= \frac{1}{X}\left(\frac{\omega^i}{N+1}\cdot\frac{X^{N+1}-1}{X-\omega^i} - \frac{1}{N+1}\right)\\ &= \frac{1}{X}\cdot\frac{\omega^i(X^{N+1}-1) - (X-\omega^i)}{(N+1)(X-\omega^i)}\\ &= \frac{\omega^i X^N - 1}{(N+1)(X-\omega^i)}\\ &= \frac{(X^{N+1}-1) - (X^{N+1} - \omega^i X^N)}{(N+1)(X-\omega^i)}\\ &= \frac{L_i(X)}{\omega^i} - \frac{X^N}{N+1}\end{aligned}$$
Therefore, $h_4 = \frac{[L_i(\tau)]_1}{\omega^i} - \frac{[\tau^N]_1}{N+1}$ and can be computed by 2 $\mathbb{G}_1$ multiplications and 1 $\mathbb{G}_1$ addition.
- For $h_{5,j}(X)$, since
$$\frac{L_i(X)L_j(X)}{Z(X)} = \frac{\omega^i\omega^j}{(N+1)^2}\cdot\frac{X^{N+1}-1}{(X-\omega^i)(X-\omega^j)},$$
we have
$$h_{5,j}(\omega^i) = \frac{1}{N+1}\cdot\frac{\omega^j}{\omega^i-\omega^j}$$
$$h_{5,j}(\omega^j) = \frac{1}{N+1}\cdot\frac{\omega^i}{\omega^j-\omega^i}$$
and $h_{5,j}(\omega^k) = 0$ for all $k \neq i, j$. Therefore,
$$\begin{aligned}[h_{5,j}(\tau)]_1 &= \sum_{k=0}^{N} h_{5,j}(\omega^k)[L_k(\tau)]_1\\ &= h_{5,j}(\omega^i)[L_i(\tau)]_1 + h_{5,j}(\omega^j)[L_j(\tau)]_1.\end{aligned}$$
Notice that $h_{5,j}(\omega^i) + h_{5,j}(\omega^j) = -\frac{1}{N+1}$, we can further write
$$[h_{5,j}(\tau)]_1 = -\frac{[L_i(\tau)]_1}{N+1} + h_{5,j}(\omega^j)([L_j(\tau)]_1 - [L_i(\tau)]_1)$$
where the first term is constant across all $h_{5,j}$. This allows us to compute each $h_{5,j}$ by 1 $\mathbb{G}_1$ multiplication and 2 $\mathbb{G}_1$ additions, with a single multiplication for computing $-\frac{[L_i(\tau)]_1}{N+1}$.

In conclusion, the total complexity of our approach to compute H_i is $(N+3)\mathbb{G}_1$ multiplications and $(3N+1)\mathbb{G}_1$ additions, providing a significant improvement over the naive approach.

6 THE FINAL C-STS CONSTRUCTION

In this section, we propose our core contribution, i.e., the practical silent threshold signature scheme for dynamic committees. Our final construction is demonstrated in Section 6.1, and we analyze its security properties in Appendix B.

6.1 Construction of a Practical c-STS Scheme

Our final construction combines the construction in Section 4 and the improvements in Section 5. For clarity, differences from Construction 1 are shown in orange.

Construction 2 (Practical c-STS). *We specify each algorithm of practical c-STS as follows.*

- *The* Setup, KeyGen, HintGen *and* Preprocess *algorithms are the same as Construction 1.*
- $\mathsf{EpochRepr}(e)$*: Obtain committee* $\mathsf{Com}_e \subseteq [N]$ *of epoch* e*, compute* $C_{\mathsf{Com}_e}(X) = \prod_{i\in\mathsf{Com}_e'}(X-\omega^i)$ *and output* $B_e = [C_{\mathsf{Com}_e}(\tau)]_1$.
- *The* Sign *and* PartVerify *algorithms are also the same as Construction 1.*
- $\mathsf{Aggr}(\mathsf{CRS}, \mathsf{ak}, m, \{\sigma_i\}_{i\in S}, t, e)$*: Let* $\mathsf{tag} = \mathcal{H}_2(m, e)$*. Parse each* $\sigma_i = (s_i, \pi_i^{\mathsf{sign}})$*. Discard signatures from those whose hints did not verify (under* HintVerify*), whose signatures did not verify (under* PartVerify*), and who do not belong to* Com_e*. Let* S *be the set of indices of the remaining signers, and return* $\perp$ *unless* $1 \le t \le |\mathsf{Com}_e|$ *and* $|S| \ge t$*. Proceed as follows (recall that* $S' = S \cup \{0\}$*):*

1. *Define* $B(1) = 1$ *and* $B(\omega^i) = 0$ *for* $i \notin S'$. *Compute* $g(X) = \prod_{i\in S}(X - \omega^i)$ *and* $\{g'(\omega^i)\}_{i\in S}$. *For* $i \in S$, *compute*

$$B(\omega^i) = \frac{g(1)}{\omega^i \cdot (\omega^i - 1) \cdot g'(\omega^i)}$$

and set $B = [\sum_{i\in S'} B(\omega^i) L_i(\tau)]_2$.
2. *Compute* $\mathsf{aPK} = \frac{1}{N+1}(\sum_{i\in S'} B(\omega^i)\mathsf{pk}_i)$.
3. *Compute*

$$Q_Z = [Q_Z(\tau)]_1 = \sum_{i\in S'} B(\omega^i) \cdot (\mathsf{ak}_i^{(4)} + \mathsf{ak}_i^{(6)}),$$

$$Q_x = [Q_x(\tau)]_1 = \sum_{i\in S'} B(\omega^i) \cdot \mathsf{ak}_i^{(5)},$$

$$\hat{Q}_x = [Q_x(\tau)\cdot\tau]_1 = \sum_{i\in S'} B(\omega^i) \cdot \mathsf{ak}_i^{(3)}.$$

4. *Compute* $\sigma^* = \frac{1}{N+1}(\sum_{i\in S} B(\omega^i) s_i + \mathsf{tag})$.
5. *Let* $\hat{B}(1) = 1$; *for* $i \in S$, *compute* $\hat{B}(\omega^i)_{i\in S} = \omega^{i(t+1)} B(\omega^i)$ *and* $\hat{B}' = [\sum_{i\in S'} \hat{B}(\omega^i) L_i(\tau)]_1$. *Set*

$$\hat{B} = [\hat{B}(\tau)]_1 = \begin{cases} \hat{B}', & |S| > t \\ \hat{B}' + \dfrac{g(1)}{N+1} \cdot [Z(\tau)]_1, & |S| = t \end{cases}$$

6. *Compute* $\bar{B}(X) = \frac{g(1)}{N+1} \prod_{i\in \mathsf{Com}_e \setminus S}(X - \omega^i)$ *and set* $\bar{B} = [\bar{B}(\tau)]_1$.
7. *Let* $Q_0(X) = \frac{B(X) - L_0(X)}{X-1}$ *(but do not compute* $Q_0(X)$ *and* $Q_0 = [Q_0(\tau)]_1$ *directly). For* $i \in S$, *compute* $Q_0(\omega^i)_{i\in S} = \frac{B(\omega^i)}{\omega^i - 1}$ *and* $Q_0(1) = \sum_{i\in S} \frac{1}{\omega^i - 1}$. *Finally, set* $Q_0 = [Q_0(\tau)]_1 = [\sum_{i\in S'} Q_0(\omega^i) L_i(\tau)]_1$.
8. *Output* $\sigma = (B, -\mathsf{aPK}, -Q_Z, -Q_x, \hat{Q}_x, \sigma^*, -\hat{B}, -\bar{B}, -Q_0)^\top$.

- $\mathsf{Verify}(\mathsf{vk}, m, \sigma, t, e)$: *If* $1 \le t \le |\mathsf{Com}_e|$ *does not hold, output 0. Otherwise, let* $\mathsf{tag} = \mathcal{H}_2(m, e)$, *set* $Z_0 = [\tau - 1]_2$ *and* $E = [L_0(\tau)]_T$, *compute*

$$\boldsymbol{A} = \begin{pmatrix} C & [1]_2 & Z & [\tau]_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & [\tau]_2 & [1]_2 & 0 & 0 & 0 & 0 \\ 0 & \mathsf{tag} & 0 & 0 & 0 & [1]_1 & 0 & 0 & 0 \\ [\tau^{t+1}]_1 & 0 & 0 & 0 & 0 & 0 & [1]_2 & 0 & 0 \\ B_e & 0 & 0 & 0 & 0 & 0 & 0 & Z & 0 \\ [1]_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Z_0 \end{pmatrix}$$

$$\boldsymbol{b} = ([0]_T, [0]_T, [0]_T, [0]_T, [0]_T, E)^\top$$

and output 1 iff $\boldsymbol{A} \circ \sigma = \boldsymbol{b}$.

6.2 Remarks on Epoch Setup and Verification Complexity

In our scheme, achieving $O(1)$ complexity for signature verification relies on access to an authenticated B_e and committee size n. Although we assume this in Section 3, establishing this epoch metadata still requires care. Methods that explicitly read the committee list and directly establish its relation to the commitment require $\Omega(n)$ work just to read the list. This is not a lower bound for all proof systems: when the committee is represented by an authenticated digest, a verifier of a suitable succinct argument need not read every member index.

To avoid imposing this epoch setup cost on each signature verification, we follow the paradigm established in [30], decoupling the computation or verification of B_e from per-message signature verification. Establishing and caching (B_e, n) is a one-time operation per epoch. This can be instantiated in two ways:

- At the beginning of a new epoch, the verifier computes B_e from the agreed-upon committee list, which requires $O(n \log^2 n)$ field operations + $\mathrm{msm}_1(n)$, and caches (B_e, n) locally. The setup cost is amortized over all signatures within the epoch, each of which can then be verified in $O(1)$ time.
- In on-chain environments, a smart contract can verify a sound succinct argument that the authenticated committee consists of n distinct indices in $[N]$ and that

$$B_e = \left[\prod_{i\in \mathsf{Com}'_e} (\tau - \omega^i)\right]_1.$$

An external prover computes B_e and generates the argument. The contract checks it against the authenticated epoch metadata and stores (B_e, n) once per epoch, enabling subsequent signature verification in $O(1)$ time.

In summary, while the end-to-end process inherently includes a committee-dependent epoch-boundary cost, our design isolates this cost to the epoch boundary. Consequently, the signature verification complexity remains $O(1)$.

7 DEFINITIONS OF C-STE

As another contribution of this work, we formally define a new primitive, namely the silent threshold encryption scheme for dynamic committees (c-STE), in this section, and give a practical construction in Section 8. Our definition adheres to the structure of [26, 27, 30]. We extend silent threshold encryption to support dynamic committees, in which the authorized committee is a subset of the entire set of parties that changes dynamically over epochs, and messages encrypted for a specific epoch can be decrypted only by the corresponding committee.

Definition 7.1 (c-STE). A silent threshold encryption scheme for dynamic committees consists of a tuple of algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{HintGen}, \mathsf{Preprocess}, \mathsf{EpochRepr}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{PartVerify}, \mathsf{DecAggr})$ with the following syntax:

- $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, N)$: On input the security parameter λ and an upper bound N on the maximum number of users, the Setup algorithm outputs a common reference string CRS.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$: On input the security parameter λ, the KeyGen algorithm outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{hint}_i \leftarrow \mathsf{HintGen}(\mathsf{CRS}, sk, N, i)$: On input the CRS, the secret key sk, the number of parties N, and a position $i \in [N]$, the $\mathsf{HintGen}$ algorithm outputs a hint hint_i.
- $(\mathsf{ak}, \mathsf{ek}) \leftarrow \mathsf{Preprocess}(\mathsf{CRS}, \{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]})$: On input the CRS, all pairs $\{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]}$, the $\mathsf{Preprocess}$ algorithm outputs an aggregation key ak and an encryption key ek.
- $B_e \leftarrow \mathsf{EpochRepr}(e)$: On input epoch e, the $\mathsf{EpochRepr}$ algorithm outputs a mathematical representation B_e of the epoch e's committee $\mathsf{Com}_e \subseteq [N]$.

- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{msg}, t, e)$: On input an encryption key ek, a message msg, a threshold t, and epoch e, the Enc algorithm outputs a ciphertext ct.
- $\sigma \leftarrow \mathsf{PartDec}(\mathsf{sk}, \mathsf{ct}, e)$: On input a secret key sk, a ciphertext ct, and epoch e, the $\mathsf{PartDec}$ algorithm outputs a partial decryption σ.
- $1/0 \leftarrow \mathsf{PartVerify}(\mathsf{pk}, \sigma, \mathsf{ct}, e)$: On input a ciphertext ct, a partial decryption σ, a public key pk, and epoch e, the $\mathsf{PartVerify}$ algorithm outputs 1 iff the partial decryption verifies and corresponds to epoch e.
- $\mathsf{msg} \leftarrow \mathsf{DecAggr}(\mathsf{CRS}, \mathsf{ak}, \mathsf{ct}, \{\sigma_i\}_{i\in S}, t, e)$: On input the CRS, an aggregation key ak, a ciphertext ct, a set of partial decryptions $\{\sigma_i\}_{i\in S}$, a threshold t, and epoch e, the $\mathsf{DecAggr}$ algorithm outputs a message msg.

Moreover, the c-STE scheme must have the following efficiency requirements:

- The size of the encryption key ek and the ciphertext ct should be $O(1)$.
- The complexity of $\mathsf{PartDec}$ should be $O(1)$.
- For epoch e with the committee size $n = |\mathsf{Com}_e|$, the complexity of $\mathsf{EpochRepr}$ should be $O(n \cdot \mathrm{polylog}(n))$.
- Given the commitment of a committee B_e, the complexity of Enc should be $O(1)$, and the complexity of $\mathsf{DecAggr}$ should be $O(n \cdot \mathrm{polylog}(n))$.

We also define the correctness and semantic security of c-STE through two games between a challenger and an adversary $\mathcal{A}$. We defer the formal definition to Appendix C.

8 C-STE CONSTRUCTION

By eliminating the Plonk-style proofs, our c-STS scheme obtains a fully linear signature verification. This allows us to leverage the compiler proposed by [27] to construct a c-STE scheme from our c-STS scheme, and the c-STE scheme naturally inherits all the efficiency advantages of our c-STS scheme. We propose this construction of the practical c-STE scheme in this section, and analyze its security properties in Appendix D.

8.1 Construction of a Practical c-STE Scheme

Construction 3 (Practical c-STE). *We specify each algorithm of practical c-STE as follows, and we simplify the description of the algorithms that remain unchanged from Construction 2.*

- *The* $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{HintGen}, \mathsf{Preprocess}$ *and* $\mathsf{EpochRepr}$ *algorithms are the same as Construction 2.*
- $\mathsf{Enc}(\mathsf{ek}, \mathsf{msg}, t, e)$*: If* $1 \le t \le |\mathsf{Com}_e|$ *does not hold, output* $\perp$*. Otherwise, sample* $u \leftarrow_\$ \mathbb{Z}_p$ *and let* $\mathsf{tag} = \mathcal{H}_2(u, e)$*. Compute the matrix* $\boldsymbol{A}$ *and vector* $\boldsymbol{b}$ *as the* Verify *algorithm of Construction 2. Sample a vector* $\boldsymbol{s} \leftarrow_\$ \mathbb{Z}_p^6$ *and output* $\mathsf{ct} = (u, \boldsymbol{s}^\top \cdot \boldsymbol{A}, \boldsymbol{s}^\top \cdot \boldsymbol{b} + \mathsf{msg})$.
- $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, e)$*: Parse* $\mathsf{ct} = (u, \boldsymbol{ct}_2, ct_3)$ *and output* $\sigma_i = \mathsf{Sign}(\mathsf{sk}_i, u, e)$ *using the* Sign *algorithm of Construction 2.*
- $\mathsf{PartVerify}(\mathsf{pk}_i, \sigma_i, \mathsf{ct}, e)$*: Parse* $\mathsf{ct} = (u, \boldsymbol{ct}_2, ct_3)$ *and output 1 iff* $\mathsf{PartVerify}(\mathsf{pk}_i, \sigma_i, u, e) = 1$ *using the* $\mathsf{PartVerify}$ *algorithm of Construction 2.*
- $\mathsf{DecAggr}(\mathsf{CRS}, \mathsf{ak}, \mathsf{ct}, \{\sigma_i\}_{i\in S}, t, e)$*: Parse* $\mathsf{ct} = (u, \boldsymbol{ct}_2, ct_3)$ *and compute* $\boldsymbol{w} = \mathsf{Aggr}(\mathsf{CRS}, \mathsf{ak}, u, \{\sigma_i\}_{i\in S}, t, e)$ *using the* Aggr *algorithm of Construction 2. If the aggregation returns* $\perp$*, output* $\perp$*. Otherwise, output* $\mathsf{msg}^* = ct_3 - \boldsymbol{ct}_2 \circ \boldsymbol{w}$.

9 IMPLEMENTATION AND EVALUATION

We implement our practical c-STS and c-STE schemes in Rust and evaluate their performance.

Experimental setup. We use the arkworks library [1] for elliptic curve and other mathematical operations (including the implementation of Pippenger's method [36] for the multi-scalar multiplication of curve points). We choose BLS12-381 as the underlying pairing-friendly elliptic curve, where an element in $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2$ is of size 32, 48, 96 bytes, respectively. For the total number of parties N[4], we use $N \approx 2^{20}$ as a benchmark upper bound for evaluation[5]. We always set the committee threshold $t = n/2$ for simplicity.

For the sake of fairness, we only implement the single-thread version of our schemes (although there are obvious opportunities for parallelism) and run it on a machine equipped with an Intel Core Ultra 7 265K processor (3.90 GHz) and 64 GB of RAM, running Windows 11 with WSL2 and Ubuntu 20.04 LTS.

9.1 Evaluation of the c-STS Scheme

We evaluate the performance of our c-STS scheme and compare it with the Rust implementation of Dyna-hinTS [2].

Epoch committee representation. The representation of epoch committees has a constant size in both Dyna-hinTS (1 $\mathbb{G}_2$, 96 bytes) and our scheme (1 $\mathbb{G}_1$, 48 bytes). As a comparison, a direct representation of a committee of size n in the BLS multi-signature scheme can use $\min(N, \lceil n \log_2 N \rceil)$ bits (2560 bytes under the scale of Ethereum).

Computing the representation of epoch committees in Dyna-hinTS only requires n $\mathbb{G}_2$ point additions. In comparison, our scheme incurs a higher cost for computing the corresponding polynomial and commitment (although they are in $\mathbb{G}_1$), which has complexity $O(n \log^2 n)$. We compare the time in Table 1 and argue that this overhead is acceptable when n is within a reasonable range. In return, this representation enables a substantial optimization of the aggregation time, which we demonstrate below.

Table 1: Time to compute the representation of epoch committees (ms)

n	Dyna-hinTS	This work	Overhead
128	0.218	3.364	15.45×
256	0.468	6.101	13.02×
512	0.913	11.209	12.28×
1024	1.815	20.544	11.32×
2048	3.803	40.478	10.64×
4096	7.469	82.111	10.99×

Partial sign and verification time. Computing and verifying a partial signature both take about 1.4 ms. The signing cost consists of one BLS signature computation and one Eq-DL proof generation,

[4]For ease of reading, in this section we include the dummy party in N so that N is a power of 2.

[5]This is close to the total number of validators in Ethereum; a system with larger N is hard to find in practice.

while verification computes and checks the Eq-DL proof without pairing operations. The partial signature contains one $\mathbb{G}_2$ element and two $\mathbb{Z}_p$ elements, i.e., 160 bytes.

Signature aggregation time. According to the analysis in 5, the cost of signature aggregation can be summarized as follows:

- Computing $g(X), g'(\omega^i)$ and $B(\omega^i)$ require $O(n \log^2 n)$ field operations, and computing B requires $\mathrm{msm}_2(n)$.
- Computing aPK requires $\mathrm{msm}_1(n)$.
- Computing $Q_Z, Q_x, \hat{Q}_x$ requires $\mathrm{msm}_1(n)$ each.
- Computing σ^* requires $\mathrm{msm}_2(n)$.
- Computing $\hat{B}(\omega^i)$ requires $O(n)$ field operations, and computing $\hat{B}$ requires $\mathrm{msm}_1(n)$.
- Computing $\bar{B}(X)$ requires $O(n \log^2 n)$ field operations, and computing $\bar{B}$ requires $\mathrm{msm}_1(n)$.
- Computing $Q_0(\omega^i)$ requires $O(n)$ field operations, and computing Q_0 requires $\mathrm{msm}_1(n)$.

We conclude that our total aggregation cost is $O(n \log^2 n)$ field operations + 7 $\mathrm{msm}_1(n)$ + 2 $\mathrm{msm}_2(n)$.

We compare the time of signature aggregation of two schemes in Table 2, which excludes all verification of the partial signature and assumes $t = n/2$ valid partial signatures are provided. The results show that the aggregation time of Dyna-hinTS increases substantially with N, which is consistent with our analysis in Section 4. In contrast, the aggregation time only scales quasilinearly with n in our scheme and is not affected by N. This advantage becomes particularly significant in large-scale systems where N is large and n is small: when $N = 1048576$ and $n = 1024$, our scheme only requires 0.145 seconds to compute the aggregated signature, achieving more than 140× improvement over Dyna-hinTS. In small-scale systems ($N \leq 4096$), our scheme also brings improvements up to about 8×. Overall, the results demonstrate that our scheme provides high efficiency and scalability across different system sizes, making it well-suited for latency-sensitive aggregation scenarios.

Table 2: Time to compute the aggregated signature (s)

N	n	Dyna-hinTS	This work	Improvement
256	64	0.059	0.021	2.887×
1024	64	0.083	0.021	4.012×
	256	0.098	0.050	1.985×
4096	64	0.165	0.021	8.036×
	256	0.169	0.050	3.408×
	1024	0.205	0.145	1.417×
16384	64	0.407	0.021	19.782×
	256	0.424	0.050	8.554×
	1024	0.492	0.145	3.400×
65536	64	1.315	0.021	63.921×
	256	1.366	0.050	27.527×
	1024	1.433	0.145	9.914×
1048576	64	20.349	0.021	988.848×
	256	20.690	0.050	417.070×
	1024	20.643	0.145	142.780×
	4096	20.769	0.545	38.139×

Signature size. We compare the size of the aggregated signature, the verification key, and the number of pairing operations in the verification algorithm of three schemes in Table 3. Although the BLS multi-signature scheme requires the fewest pairings, the aggregated signature must represent the set of signers (denoted by *Repr*), so the signature size scales with n when the epoch committee is given, and scales with $\min(N, \lceil n \log_2 N \rceil)$ otherwise. Moreover, verifying a BLS multi-signature requires storing all public keys and computing the verification key, which is a significant storage and computational overhead. In comparison, both Dyna-hinTS and our scheme achieve constant signature size, verification key size, and number of pairing operations. By removing Plonk-style proofs in Dyna-hinTS, our scheme reduces the signature size to 528 bytes (compared to 896 bytes in Dyna-hinTS). On the other hand, our scheme adds four pairings in verification because we check the corresponding relations by three additional linear pairing equations. Nevertheless, the verification time is only about 7.3 ms (with access to the epoch committee representation).

Table 3: Complexity of signature verification

Scheme	Signature size	Verification key size	Pairings	Time
BLS	1 $\mathbb{G}_2$, *Repr*	N $\mathbb{G}_1$	2	~1.5 ms + Computing vk
Dyna-hinTS	8 $\mathbb{G}_1$, 2 $\mathbb{G}_2$, 10 $\mathbb{Z}_p$	1 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	9	5.2 ms
This work	7 $\mathbb{G}_1$, 2 $\mathbb{G}_2$	1 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	13	7.3 ms

9.2 Evaluation of the c-STE Scheme

While our c-STE construction closely parallels the c-STS scheme (and thus shares similar performance in committee representation and polynomial aggregation), we provide its evaluation to demonstrate its practical efficiency for dynamic committees.

Table 4 gives the computational time for the c-STE operations. The time to generate a ciphertext (with access to the epoch committee representation) and to compute/verify a partial decryption remains constant. The decryption time equals the signature aggregation time in the c-STS scheme (Table 2) plus 9 additional pairings (which take about 3.2 ms) to compute $\boldsymbol{ct}_2 \circ \boldsymbol{w}$. Table 5 gives the sizes of the corresponding components, showing that the communication overhead of our c-STE is succinct.

Table 4: Running times of c-STE operations

Operation	Time	Remarks
Encryption	5.05 ms	Constant time
Partial decryption	1.37 ms	Constant time (BLS with Eq-DL)
Partial verification	1.42 ms	Constant time (BLS with Eq-DL)
Decryption	Table 2 + 3.2 ms	Aggregation + 9 pairings

9.3 Evaluation of Silent Setup

We also evaluate the one-time setup cost of our scheme.

Hint generation. We evaluate the impact of the optimization proposed in Section 5.3 on the time required for a single party to generate its hint during the setup phase. As analyzed in Section 5.3, our optimization reduces this computational complexity from

Table 5: Sizes of c-STE components

Component	Size
Encryption key	1 $\mathbb{G}_1$, 1 $\mathbb{G}_2$
Plaintext	1 $\mathbb{G}_T$
Ciphertext	1 $\mathbb{Z}_p$, 2 $\mathbb{G}_1$, 7 $\mathbb{G}_2$, 1 $\mathbb{G}_T$
Partial decryption	1 $\mathbb{G}_2$, 2 $\mathbb{Z}_p$

about $O(N^2/\log N)$ to $O(N)$, which is consistent with the results in Table 6. The execution time of the naive setup becomes experimentally infeasible for $N \geq 4096$, and we do not report it therefore (denoted by '-'); in contrast, our optimized algorithm only requires about 1.6 minutes to generate the hint when $N = 2^{20}$, demonstrating that our optimization is necessary and practically efficient.

Table 6: Time to compute a hint for a single party (s)

N	Naive approach	Our approach	Improvement
64	0.143	0.007	21.60×
256	1.634	0.024	67.48×
1024	27.16	0.092	295.11×
4096	-	0.375	-
16384	-	1.477	-
65536	-	5.965	-
262144	-	23.745	-
1048576	-	97.503	-

System preprocessing phase. While the Preprocess algorithm will be executed only once, it is crucial to ensure its feasibility when the system scale is large. The following analysis assumes that (1) the terms required for hint verification (the public commitment terms on the right-hand sides of Equation 3) are cached during system initialization, and (2) all hints are valid, namely $|V| = N$. The Preprocess algorithm consists of two parts:

- Hint verification: Executing HintVerify naively needs $2N+8$ pairings per hint, which can be reduced to $\mathrm{msm}_1(N) + \mathrm{msm}_2(N) + 2$ pairings by performing random linear combinations to each side [30][6].
- Computing ak and vk: Requiring N^2-1 and N $\mathbb{G}_1$ additions, respectively.

Therefore, the average computational overhead per hint is closely approximated by a HintVerify together with $N + 1$ $\mathbb{G}_1$ additions, and we denote its time by T.

As the system scales, evaluating this setup sequentially on a single machine becomes impractical. However, this setup is fully parallelizable in practice because these computations are highly repetitive and independent. We consider the parallelism level $P = P_1P_2$ from two dimensions:

- Node parallelism (P_1): The workload can be evenly distributed among multiple participating parties, which is highly feasible when $N \gg P_1$. The system could even assign multiple groups to execute the Preprocess algorithm redundantly to enhance trust in the results.
- Hardware parallelism (P_2): Individual parties can leverage multithreading to process their assigned batches. The workload distribution can also be adjusted dynamically based on heterogeneous hardware capabilities.

Therefore, the execution time of Preprocess can be reasonably estimated by $C = T \times N/P$, which assumes ideal load balancing and linear speedup and excludes communication, storage I/O, and coordination overhead. Table 7 shows this theoretical estimation across different system scales assuming an accessible distributed configuration of $P_1 = 128$ and $P_2 = 4$ (i.e., total parallelism $P = 512$). Under this setting, the overall execution time for an extraordinarily large scale of $N = 1048576$ drops to approximately 24.8 hours. Since this is a strict one-time setup, we consider this computational overhead acceptable for large-scale distributed systems.

Table 7: Estimated setup time at $P = 512$

N	T (ms)	C (s)
1024	87.41	0.17
4096	271.84	2.17
16384	935.30	29.93
65536	3219.64	412.11
262144	11829.61	6056.75
1048576	43617.72	89329.08

Dynamic party additions. Assume the system is initialized with a domain size of $|\mathbb{H}| = 2^k$. When a new party joins the system, there are two possible scenarios based on the number of existing active parties:

- If the system is full, i.e., already has 2^k parties (including the dummy party), the current $\mathbb{H}$ cannot accommodate new evaluations. All parties must therefore re-execute the setup using a larger multiplicative subgroup $\mathbb{H}$.
- If the system is not full, the join process is efficient. The new party first computes and submits its own hint with $O(N)$ complexity (Table 6). For existing parties, since their original hints depend solely on their own sk and the global $\mathbb{H}$ remains unchanged, they do not need to perform any operations. For the Preprocess algorithm, the system incrementally updates the parameters: it verifies the new party's hint, adds $\mathsf{hint}^{(1)}$ to vk, adds the new party's $\mathsf{hint}^{(5)}$ cross-terms to the corresponding existing entries in ak, and computes the new party's accumulated $\mathsf{hint}^{(5)}$ using the original raw hints provided by the existing parties (assuming the aggregator archives these original components). The time cost of this incremental update is about T in Table 7 (about 43.6 s at $N = 2^{20}$) in total.

In summary, our scheme achieves highly practical dynamic party integration for the vast majority of scenarios.

10 DISCUSSION

Our work achieves a significant leap in aggregation efficiency by replacing the Plonk-style Hamming weight proofs for B and $\bar{B}$ in

[6]When the same party verifies multiple hints, the pairing cost can be further reduced by applying an additional random linear combination across the hints since their left-hand sides share the same $[1]_2$.

Dyna-hinTS with linear pairing checks. However, this efficiency gain inevitably introduces a trade-off in expressiveness: our scheme no longer natively supports non-trivial party weights or general access structures. The fundamental reason lies in the mathematical nature of our polynomial subset checks. Our current linear pairing checks can only verify inclusion relationships among the roots of polynomials (i.e., whether a party is present), but cannot verify non-zero numerical attributes without destroying the underlying algebraic structure. In contrast, a Plonk-style SNARK can easily support weighted structures by computing the inner product of B and a public weight vector w instead of $(1, 1, \ldots, 1)$. It can even support arbitrary access structures by replacing the inner product with custom circuit predicates. However, the full-domain Plonk construction considered here strictly requires the aggregator to perform $O(N \log N)$ operations over the global domain of size N, which inherently contradicts our goal of $O(n \cdot \text{polylog}(n))$ aggregation efficiency. Consequently, designing a c-STS scheme that achieves both $O(n \cdot \text{polylog}(n))$ aggregation efficiency and support for complex access structures remains a highly challenging open question.

The situation is even more challenging for STE. Currently, the only known method for constructing a practical STE is to leverage the compiler proposed in [27], which transforms a public linearly verifiable STS into an STE. Therefore, the original scheme in [27] is inherently limited to unweighted threshold settings. Moreover, whether there exists any STE scheme supporting weighted structures—let alone c-STE schemes—remains unknown. We consider solving this an important and valuable direction for future research.

11 CONCLUSION

We presented a practical construction of the silent threshold signature scheme for dynamic committees (c-STS). For system size N and committee size n, we reduced the signature aggregation complexity from $O(N \log N)$ in prior work to $O(n \log^2 n)$ by introducing fully linear verification equations, together with a new committee representation and several polynomial-level optimizations. We further optimized the one-time silent setup phase, reducing the complexity of per-party setup from $O(N^2)$ to $O(N)$, while the signature size, verification key size, and verification time (with access to the epoch committee representation) of our scheme remained constant. Our Rust implementation confirms the practical impact of these improvements: with system parameters comparable to Ethereum, signature aggregation completes in about 0.145 seconds, and one-time setup becomes feasible even for systems with millions of parties.

Furthermore, we introduced the notion and practical construction of the silent threshold encryption scheme for dynamic committees (c-STE). This construction inherits the same efficiency and scalability benefits as our c-STS scheme. In conclusion, we demonstrate that silent threshold cryptography can be made highly practical for dynamic committee settings and deployed in real-world large-scale distributed systems.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 62272269), Scientific Research Innovation Capability Support Project for Young Faculty (Grant No. ZYGXQNJSKYCXNLZCXM-I21), and Primary Research and Development Plan of Jiangsu Province (Grant No. BE2023025). We thank Professor Jun Shao (Zhejiang Gongshang University) for sharing his independent findings concerning the security issue and for suggesting an alternative approach to addressing it.

REFERENCES

[1] [n. d.]. arkworks — arkworks.rs. https://arkworks.rs/. [Accessed 04-02-2026].
[2] [n. d.]. GitHub - dynaHintsRepo/dynaHints — github.com. https://github.com/dyna{H}ints{R}epo/dyna{H}ints.git. [Accessed 31-01-2026].
[3] [n. d.]. Proof of Stake Consensus - Polkadot Developer Docs — docs.polkadot.com. https://docs.polkadot.com/reference/polkadot-hub/consensus-and-security/pos-consensus/. [Accessed 05-02-2026].
[4] [n. d.]. Upgrading Ethereum | 2.9.5 Committees — eth2book.info. https://eth2book.info/capella/part2/building_blocks/committees/. [Accessed 05-02-2026].
[5] Inc. Algorand Foundation US. [n. d.]. Pure Proof-of-Stake - how Algorand's consensus mechanism works — algorand.co. https://algorand.co/technology/pure-proof-of-stake. [Accessed 05-02-2026].
[6] Thomas Attema and Ronald Cramer. 2020. Compressed-protocol theory and practical application to plug & play secure algorithmics. In *Annual International Cryptology Conference*. Springer, 513–543.
[7] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. 2021. Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 526–556.
[8] Leemon Baird, Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. 2023. Threshold signatures in the multiverse. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1454–1470.
[9] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Francois Garillot, Jonas Lindstrøm, Ben Riva, Arnab Roy, Mahdi Sedaghat, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. 2024. Subset-optimized bls multi-signature with key aggregation. In *International Conference on Financial Cryptography and Data Security*. Springer, 188–205.
[10] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. 2001. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*. Springer, 1–18.
[11] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. 2015. Communication-optimal proactive secret sharing for dynamic groups. In *International conference on applied cryptography and network security*. Springer, 23–41.
[12] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. 2005. Hierarchical identity based encryption with constant size ciphertext. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 440–456.
[13] Dan Boneh, Manu Drijvers, and Gregory Neven. 2018. Bls multi-signatures with public-key aggregation. *URL: https://crypto. stanford. edu/dabo/pubs/papers/BLSmultisig. html* (2018).
[14] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short signatures from the Weil pairing. In *International conference on the theory and application of cryptology and information security*. Springer, 514–532.
[15] Xavier Boyen. 2008. The uber-assumption family: A unified complexity framework for bilinear groups. In *International Conference on Pairing-Based Cryptography*. Springer, 39–56.
[16] Jeffrey Champion, Yao-Ching Hsieh, and David J Wu. 2025. Registered ABE and adaptively-secure broadcast encryption from succinct LWE. In *Annual International Cryptology Conference*. Springer, 3–34.
[17] Jeffrey Champion and David J Wu. 2024. Distributed broadcast encryption from lattices. In *Theory of Cryptography Conference*. Springer, 156–189.
[18] David Chaum and Torben Pryds Pedersen. 1992. Wallet databases with observers. In *Annual international cryptology conference*. Springer, 89–105.
[19] Tarun Chitra and Uthsav Chitra. 2019. Committee selection is more similar than you think: Evidence from avalanche and stellar. *arXiv preprint arXiv:1904.09839* (2019).
[20] Sourav Das, Philippe Camacho, Zhuolun Xiang, Javier Nieto, Benedikt Bünz, and Ling Ren. 2023. Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 356–370.
[21] Yvo Desmedt and Kirill Morozov. 2015. Parity check based redistribution of secret shares. In *2015 IEEE International Symposium on Information Theory (ISIT)*.

IEEE, 959–963.
[22] Cody Freitag, Brent Waters, and David J Wu. 2023. How to use (plain) witness encryption: registered ABE, flexible broadcast, and more. In *Annual International Cryptology Conference*. Springer, 498–531.
[23] Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. 2019. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive* (2019).
[24] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. 2016. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.* 45, 3 (2016), 882–929.
[25] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. 2013. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 467–476.
[26] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. 2024. hints: Threshold signatures with silent setup. In *2024 IEEE symposium on security and privacy (SP)*. IEEE, 3034–3052.
[27] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. 2024. Threshold encryption with silent setup. In *annual international cryptology conference*. Springer, 352–386.
[28] Timo Hanke, Mahnush Movahedi, and Dominic Williams. 2018. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548* (2018).
[29] Aniket Kate, Easwar Vivek Mangipudi, Siva Maradana, and Pratyay Mukherjee. 2023. Flexirand: Output private (distributed) vrfs and application to blockchains. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 1776–1790.
[30] Aniket Kate, Pratyay Mukherjee, Samipa Samanta, and Pratik Sarkar. 2025. Dyna-hinTS: Silent Threshold Signatures for Dynamic Committees. *Cryptology ePrint Archive* (2025).
[31] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. 2010. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*. Springer, 177–194.
[32] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. 2023. Distributed broadcast encryption from bilinear groups. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 407–441.
[33] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. 2019. CHURP: dynamic-committee proactive secret sharing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2369–2386.
[34] Ueli Maurer. 2005. Abstract models of computation in cryptography. In *IMA International Conference on Cryptography and Coding*. Springer, 1–12.
[35] Silvio Micali, Leonid Reyzin, Georgios Vlachos, Riad S Wahby, and Nickolai Zeldovich. 2021. Compact certificates of collective knowledge. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 626–641.
[36] Nicholas Pippenger. 1980. On the evaluation of powers and monomials. *SIAM J. Comput.* 9, 2 (1980), 230–250.
[37] Carla Ràfols and Arantxa Zapico. 2021. An algebraic framework for universal and updatable SNARKs. In *Annual International Cryptology Conference*. Springer, 774–804.
[38] Leonid Reyzin, Adam Smith, and Sophia Yakoubov. 2021. Turning HATE into LOVE: compact homomorphic ad hoc threshold encryption for scalable MPC. In *International Symposium on Cyber Security Cryptography and Machine Learning*. Springer, 361–378.
[39] David Schultz, Barbara Liskov, and Moses Liskov. 2010. MPSS: mobile proactive secret sharing. *ACM Transactions on Information and System Security (TISSEC)* 13, 4 (2010), 1–32.
[40] Hoeteck Wee and David J Wu. 2025. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In *Annual International Cryptology Conference*. Springer, 204–235.
[41] Lidong Zhou, Fred B Schneider, and Robbert Van Renesse. 2005. APSS: Proactive secret sharing in asynchronous systems. *ACM transactions on information and system security (TISSEC)* 8, 3 (2005), 259–286.

A DEFINITIONS OF CORRECTNESS AND UNFORGEABILITY FOR C-STS

Following the idea of [26, 30], we still define the correctness and unforgeability of our c-STS scheme through two games between a challenger who controls the honest parties and an adversary $\mathcal{A}$ who may send arbitrary messages (for the hints and partial signatures) on behalf of the corrupted parties. However, we only focus on the number of parties that the adversary can control **within the epoch committee**: the protocol satisfies the security properties as long as this number is smaller than the threshold t. In contrast, our definition does not relate to the number of parties f that the adversary can control in the entire system, nor to the probability that the adversary fully controls a randomly sampled committee, which relates to the system-level parameter selection strategy and is logically independent of the cryptographic security of the protocol.

In both games below, all corrupted registrations are verified during $\mathsf{Preprocess}$ before the signing oracle is enabled. Random-oracle queries remain available during registration.

Definition A.1 (Correctness, revisited). The c-STS scheme satisfies correctness if, for any PPT adversary $\mathcal{A}$ and any $N = \mathrm{poly}(\lambda)$, the output of the correctness game defined below is 1 with probability $\leq \mathsf{negl}(\lambda)$.

1. The challenger runs $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives CRS to $\mathcal{A}$.
2. $\mathcal{A}$ picks a subset A of corrupt parties such that $A \subseteq [N]$, and A remains fixed throughout the game.
3. For all $i \in [N] \setminus A$, the challenger samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, computes $\mathsf{hint}_i \leftarrow \mathsf{HintGen}(\mathsf{CRS}, \mathsf{sk}_i, N, i)$, and gives only $(\mathsf{pk}_i, \mathsf{hint}_i)$ to $\mathcal{A}$.
4. For all $i \in A$, the adversary returns a public key pk_i and a hint hint_i to the challenger.
5. The challenger invokes the preprocessing as $(\mathsf{ak}, \mathsf{vk}) \leftarrow \mathsf{Preprocess}(\mathsf{CRS}, \{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]})$ and the outputs are given to $\mathcal{A}$. Let $V \subseteq [N]$ be the set of parties whose hints are valid.
6. $\mathcal{A}$ picks an epoch e.
7. The challenger obtains the committee Com_e with its commitment by $B_e \leftarrow \mathsf{EpochRepr}(e)$ and gives it to $\mathcal{A}$.
8. $\mathcal{A}$ picks a message m, a threshold $1 \leq t \leq |\mathsf{Com}_e|$, and prepares the partial signatures $\{\sigma_i\}_{i\in S_1}$ for some subset of malicious parties $S_1 \subseteq A$. Let $S_1' \subseteq S_1$ be the subset of indices whose maliciously generated partial signatures verify under $\mathsf{PartVerify}$.
9. $\mathcal{A}$ may also request a subset of honest parties $S_2 \subseteq \mathsf{Com}_e \setminus A$ for partial signatures, which are returned by computing $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m, e)$.
10. Let $S = (S_1' \cup S_2) \cap V \cap \mathsf{Com}_e$, and the game outputs 0 and terminates if $|S| < t$. Otherwise, the challenger computes the aggregated signature as $\sigma \leftarrow \mathsf{Aggr}(\mathsf{CRS}, \mathsf{ak}, m, \{\sigma_i\}_{i\in S}, t, e)$.
11. The output of the game is 1 if aggregation returns $\perp$ or $\mathsf{Verify}(\mathsf{vk}, m, \sigma, t, e) = 0$. Otherwise, the output is 0.

Definition A.2 (Unforgeability, revisited). The c-STS scheme satisfies unforgeability if, for any PPT adversary $\mathcal{A}$ and any $N = \mathrm{poly}(\lambda)$, the output of the forgery game defined below is 1 with probability $\leq \mathsf{negl}(\lambda)$.

1. The challenger runs $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives CRS to $\mathcal{A}$.
2. $\mathcal{A}$ picks a subset A of corrupt parties such that $A \subseteq [N]$, and A remains fixed throughout the game.
3. For all $i \in [N] \setminus A$, the challenger samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, computes $\mathsf{hint}_i \leftarrow \mathsf{HintGen}(\mathsf{CRS}, \mathsf{sk}_i, N, i)$, and gives only $(\mathsf{pk}_i, \mathsf{hint}_i)$ to $\mathcal{A}$.
4. For all $i \in A$, the adversary returns a public key pk_i and a hint hint_i to the challenger.

5. The challenger invokes the preprocessing as $(\mathsf{ak}, \mathsf{vk}) \leftarrow \mathsf{Preprocess}(\mathsf{CRS}, \{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]})$ and the outputs are given to $\mathcal{A}$. Let $V \subseteq [N]$ be the set of parties whose hints are valid.
6. $\mathcal{A}$ picks an epoch e.
7. The challenger obtains the committee Com_e with its commitment by $B_e \leftarrow \mathsf{EpochRepr}(e)$ and gives it to $\mathcal{A}$.
8. $\mathcal{A}$ may make a partial signature query with a message m and an honest party $i \in [N] \setminus A$, and the query is returned by computing $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m, e)$. This can be run polynomially many times by $\mathcal{A}$.
9. Finally, $\mathcal{A}$ shall output a challenge message m^* and an aggregated signature σ. Let S^* be the subset of honest parties queried by the adversary to sign m^*.
10. $\mathcal{A}$ wins the forgery game if there exists a threshold $|(S^* \cup A) \cap \mathsf{Com}_e| < t \leq |\mathsf{Com}_e|$ such that $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma, t, e) = 1$, in which case, the output of the game is 1. Otherwise, the output is 0.

B SECURITY ANALYSIS OF C-STS

B.1 Models and Assumptions

We prove unforgeability in the Algebraic Group Model (AGM) with programmable random oracles. For $\mathbb{G}_1$ and $\mathbb{G}_2$, algebraic representations are rooted in the public parameters and the elements issued by the challenger or the prescribed oracles. Every $\mathbb{G}_1$ or $\mathbb{G}_2$ element supplied by the adversary to the challenger or an oracle must be accompanied by a $\mathbb{Z}_p$-linear representation in previously available elements of the same group.

Definition B.1 (q-DLOG). For polynomially bounded q, the q-DLOG assumption states that no PPT algebraic adversary can recover $z \leftarrow_\$ \mathbb{Z}_p$ from

$$([1]_1, [z]_1, \ldots, [z^q]_1, [1]_2, [z]_2, \ldots, [z^q]_2)$$

with non-negligible probability.

We use the q-DLOG assumption with $q = 2N + 4$. The degree analysis below shows that this is a sufficient conservative bound for all exponent relations used in the proof. This bound is established from the available algebraic inputs, without assuming that corrupted registrations have constant public-key exponents.

Throughout the security analyses, q_0 bounds the total number of queries to $\mathcal{H}_0$ by the adversary, verifier, and challenger, excluding simulator programming. In particular, queries made internally while verifying registrations or partial proofs are included. We retain $q_0 + |A|$ as a conservative upper bound in the registration analysis.

B.2 Simulation of Honest Eq-DL Proofs

Lemma B.2. *For a true Eq-DL statement $X = kA, Y = kB$ with $A = [1]_1$, an honest proof can be simulated from its public statement in the programmable random-oracle model. If at most L proofs are simulated and at most q_0 other queries to $\mathcal{H}_0$ occur, then the statistical simulation error*

$$\epsilon_{\text{sim}} \leq O\left(\frac{L(q_0 + L)}{p}\right).$$

Proof. Sample $c, s \leftarrow_\$ \mathbb{Z}_p$, compute $\alpha = sA - cX, \beta = sB - cY$ and program $\mathcal{H}_0(A, X, B, Y, \alpha, \beta) = c$; abort if the input already occurs in the oracle table.

For a true statement, the mapping

$$(r, c) \longmapsto (r + ck, c)$$

is a bijection. Therefore, the real and simulated distributions agree until there exists a programming collision. Based on the preceding history and the public statement, α is uniform in $\mathbb{G}_1$, which means that the probability of a collision with any previous oracle input is at most $(q_0+L)/p$ per simulation. The lemma is proven by applying a union bound over L simulations.

The reconstructed commitments are linear combinations of the public statement elements. Consequently, proof simulation does not enlarge the $\mathbb{G}_1$ or $\mathbb{G}_2$ linear span beyond the elements in the statement. □

B.3 From Algebraic Relations to Polynomial Identities

Associate a formal variable X with τ, a variable K_i with every honest secret key sk_i, and a fresh variable with the exponent of every distinct $\mathcal{H}_2$ output.

Assume $N \geq 1$, and measure total degree in all hidden-exponent indeterminates. The CRS elements and public committee representations have exponent polynomials of degree at most $N + 1$. Every honest public key or algebraic hint component has the form $K_i P(X)$ with $\deg P \leq N$, and therefore has total degree at most $N + 1$. Each hash-to-group output has degree 1, and each honest partial signature has exponent $K_i T_j$ of degree 2, where T_j denotes the corresponding hash-output indeterminate.

Since $\mathbb{G}_1, \mathbb{G}_2$ linear combinations do not increase these bounds, and the reconstructed commitments in simulated Eq-DL proofs are also linear combinations of their statement elements, $\mathbb{G}_1, \mathbb{G}_2$ equality and representation-consistency relations have degree at most $N + 1$. Each pairing multiplies two $\mathbb{G}_1, \mathbb{G}_2$ exponent polynomials, so a pairing-product equality has degree at most $2N + 2$. This includes the hint-consistency checks and the final signature-verification equations. Linear operations in $\mathbb{G}_T$ do not increase this bound, and $\mathbb{G}_T$ elements cannot be paired again. Thus $q = 2N + 4$ is sufficient for all relations considered here.

Lemma B.3. *Under the q-DLOG assumption, except with negligible probability, the polynomially many accepted $\mathbb{G}_1$ or $\mathbb{G}_2$ equalities, pairing-product equalities, and representation-consistency equalities used in the proof hold as formal polynomial identities. This applies to adaptively chosen algebraic coefficients and to the execution containing the simulated Eq-DL proofs.*

Proof. Consider a nonzero polynomial relation $F(Z_1, \ldots, Z_\ell)$ of total degree at most $d \leq q$ that nevertheless evaluates to zero at the hidden exponents, and its coefficients may be chosen adaptively by the adversary. Assume that a PPT algebraic adversary $\mathcal{A}$ gives such a relation with non-negligible probability; we construct a PPT algebraic reduction $\mathcal{B}$ that solves q-DLOG with non-negligible probability as follows.

Given a q-DLOG challenge with unknown exponent z, the reduction samples independent vectors $\boldsymbol{a}, \boldsymbol{b} \leftarrow_\$ \mathbb{Z}_p^\ell$ to represent each

hidden exponent by $z_j = a_j z + b_j$. Here ℓ may be fixed to a polynomial upper bound on the number of hidden exponents used during the execution. Positions for honest secret keys are indexed by party, and positions for hash-output exponents are assigned in the order in which distinct $\mathcal{H}_2$ inputs are first queried; unused positions are ignored. Equivalently, the corresponding pairs (a_j, b_j) may be sampled lazily when their positions are first used. Every prescribed $\mathbb{G}_1$ or $\mathbb{G}_2$ input can be computed from the challenge powers by expanding its exponent polynomial after this substitution. Honest Eq-DL proofs are simulated as in Lemma B.2. Since $a_j z + b_j$ are independent and uniform for fixed z, the reduction generates the same distribution as the hybrid execution with simulated honest Eq-DL proofs and uniformly sampled hidden exponents in $\mathbb{Z}_p$.

The reduction keeps $\boldsymbol{a}, \boldsymbol{b}$ and all representations relative to the q-DLOG challenge private. The adversary receives only the protocol elements and oracle responses of the simulated execution. The adversary's algebraic representations are relative to the elements available in its simulated protocol execution.

Fix the challenge exponent z, the actual hidden-exponent tuple $\boldsymbol{v} = (z_1, \ldots, z_\ell)$, and the randomness determining the group representation, the adversary, the random oracles, and honest-proof simulation. The adversary-visible execution and its supplied algebraic coefficients then depend on $\boldsymbol{a}, \boldsymbol{b}$ only through $\boldsymbol{v}$. Based on these fixed quantities, $\boldsymbol{a}$ remains uniform over $\mathbb{Z}_p^\ell$, with $\boldsymbol{b} = \boldsymbol{v} - z\boldsymbol{a}$. In particular, any relation F selected from this execution using only its transcript and algebraic representations is fixed while $\boldsymbol{a}$ remains uniform.

For the univariate polynomial

$$\widetilde{F}(T) = F(\boldsymbol{a}T + \boldsymbol{b})$$

we have $\widetilde{F}(z) = F(\boldsymbol{v}) = 0$. Since $\boldsymbol{a}T + \boldsymbol{b} = \boldsymbol{v} + (T - z)\boldsymbol{a}$, we can write

$$F(\boldsymbol{v} + (T - z)\boldsymbol{a}) = \sum_{r=0}^{d} H_r((T - z)\boldsymbol{a}),$$

where H_r is homogeneous of degree r. Since $F \neq 0$ is nonzero but $F(\boldsymbol{v}) = 0$ (which means that $H_0 = 0$), there must exist $H_r \neq 0$ for some $r > 0$. Therefore,

$$\widetilde{F}(T) = \sum_{r=0}^{d} (T - z)^r H_r(\boldsymbol{a}).$$

If $\widetilde{F} = 0$, we must have $H_r(\boldsymbol{a}) = 0$, whose probability is at most d/p by the Schwartz-Zippel lemma. If $\widetilde{F} \neq 0$, the reduction can factor the nonzero polynomial $\widetilde{F}$, and check $[r]_1 = [z]_1$ for each root r to identify z since $\widetilde{F}(z) = 0$. A non-negligible probability of such an accepted nonidentity would therefore contradict q-DLOG.

The same argument applies to $\mathbb{G}_1$ and $\mathbb{G}_2$ representation consistency and to pairing-product relations, and there are only polynomially many relevant relations. □

B.4 Malicious Registrations

Lemma B.4. *After honest-proof simulation, except with probability*

$$\epsilon_{\text{reg}} \leq \frac{q_0 + |A|}{p} + \mathsf{negl}(\lambda),$$

every accepted corrupted registration has a constant formal public-key exponent $\kappa_i \in \mathbb{Z}_p$; its accepted hint components have the prescribed form obtained by multiplying the corresponding public hint polynomials by κ_i.

Proof. Consider a corrupted registration proof accepted by the verifier. At the first query of its Fiat-Shamir input, let P be the formal polynomial representing pk_i and R the polynomial representing its first commitment α. If the same $\mathbb{G}_1$ or $\mathbb{G}_2$ element is later supplied with a different algebraic representation, Lemma B.3 implies that the two representations agree as formal polynomials except with negligible probability. Thus we may use the representations fixed at the first query.

By Lemma B.3, $R + cP = s$ holds as a formal identity. Suppose P is nonconstant, then we can select a nonconstant monomial whose coefficient in P is $p_m \neq 0$, and denote its coefficient in R by r_m, then the identity requires $r_m + cp_m = 0$.

Before the first query is answered, P and R (including their algebraic coefficients) are fixed, and at most one challenge c satisfies this condition. The registration statement at position i contains the second base $B = [L_i(\tau)]_1$, which is included in the Fiat-Shamir input $\mathcal{H}_0(A, X, B, Y, \alpha, \beta)$. For any honest position $j \neq i$, $L_i(X) \neq L_j(X)$ since $L_i(\omega^i) = 1$ and $L_j(\omega^i) = 0$. If their evaluated bases were equal, this would be a nonzero polynomial relation $L_i(\tau) - L_j(\tau) = 0$, covered by the negligible exceptional event of Lemma B.3. Outside this event, the query cannot coincide with an honestly programmed registration input. All corrupted registrations are verified during preprocessing before the signing oracle is enabled, so no honest partial-signature proof has yet been programmed when a corrupted registration is verified.

Therefore, each fresh relevant random-oracle answer permits acceptance of a nonconstant formal public key with probability at most $1/p$. The lemma holds by taking the union over all adversarial and verifier queries, and adding the negligible probability from Lemma B.3.

Let $H_{i,r}$ denote the formal exponent of the r-th hint component for $1 \leq r \leq 4$, and let $H_{i,5,j}$ denote the formal exponent of its cross component indexed by j. When $P = \kappa_i$, the accepted hint-consistency relations and Lemma B.3 imply

$$\begin{aligned}
H_{i,1} &= \kappa_i L_i(X), \\
H_{i,2} &= \kappa_i (L_i(X) - L_i(0)), \\
H_{i,3} &= \kappa_i \frac{L_i(X)^2 - L_i(X)}{Z(X)}, \\
H_{i,4} &= \kappa_i \frac{L_i(X) - L_i(0)}{X}, \\
H_{i,5,j} &= \kappa_i \frac{L_i(X) L_j(X)}{Z(X)}.
\end{aligned}$$

□

B.5 Correctness

In this section, we prove that Construction 2 satisfies the correctness (Definition A.1).

Proof. For each registered public key pk_i, let x_i be the unique scalar satisfying $\mathsf{pk}_i = [x_i]_1$. Every accepted hint-consistency equation uniquely determines the corresponding hint component, therefore accepted hints satisfy the required algebraic relations exactly.

Except with negligible probability, every accepted partial-signature proof satisfies $s_i = x_i\mathsf{tag}$: This follows from the soundness of the Eq-DL protocol. For a false statement and fixed commitments, two distinct accepting challenges would yield a witness by special soundness; therefore at most one challenge admits an accepting response. Honest programmed inputs correspond to true statements and cannot be reused as proofs of any false statement. Taking the union over the probability of polynomially many queries gives a negligible error.

The analysis below assumes that the exceptional event does not occur. Let

$$SK(Y) = L_0(Y) + \sum_{i\in V} x_i L_i(Y),$$

where the $x_i \in \mathbb{Z}_p$ are fixed in this correctness calculation. The aggregation equation and generalized sumcheck imply

$$\begin{aligned} SK(Y)B(Y) &= \mathsf{aSK} + Q_Z(Y)Z(Y) + YQ_x(Y),\\ YQ_x(Y) &= \hat{Q}_x(Y),\\ \sigma^* &= \mathsf{aSK}\cdot\mathsf{tag}. \end{aligned}$$

The remaining polynomial identities are

$$\begin{aligned} Y^{t+1}B(Y) &= \hat{B}(Y),\\ C_{\mathsf{Com}_e}(Y)B(Y) &= Z(Y)\bar{B}(Y),\\ B(Y) - L_0(Y) &= (Y-1)Q_0(Y). \end{aligned}$$

Evaluating them at $Y = \tau$ gives all six final verification equations, therefore an aggregate formed from at least t accepted shares passes the verification except with negligible probability. □

B.6 A Formal Polynomial Threshold Bound

The following lemma concerns formal polynomial identities and membership in polynomial spaces, and uses no computational assumption or group model.

Lemma B.5 (Threshold bound for formal verification). *Let $\omega \in \mathbb{Z}_p^*$ have order $N+1$, let $L_i(X)$ be the Lagrange basis polynomials on $\{\omega^i : i \in [N]'\}$, and let $Z(X) = X^{N+1} - 1$. Fix $A \subseteq [N]$, $H = [N] \setminus A$, $D \subseteq H$, a committee $\mathsf{Com}_e \subseteq [N]$, and $1 \le t \le |\mathsf{Com}_e|$. For constants $\kappa_i \in \mathbb{Z}_p$, define*

$$\begin{aligned} F_1 &= L_0(X) + \sum_{i\in H} K_i L_i(X) + \sum_{i\in A} \kappa_i L_i(X),\\ C_{\mathsf{Com}_e}(X) &= \prod_{i\in \mathsf{Com}'_e} (X - \omega^i). \end{aligned}$$

In the polynomial ring over $\mathbb{Z}_p$ with independent indeterminates X, $\{K_i\}_{i\in H}$, Γ, and $\{\Theta_j\}_j$, define the following $\mathbb{Z}_p$-linear spaces:

$$\begin{aligned} \mathcal{V}_1 &= span\left(\{X^j\}_{j=0}^{N+1} \cup \{K_i X^j : i \in H,\ 0 \le j \le N\}\right),\\ \mathcal{V}_2(D) &= span\big(\{X^j\}_{j=0}^{N+1} \cup \{\Gamma\} \cup \{\Theta_j\}_j\\ &\qquad \cup \{K_i\Theta_j : i \in H\}_j \cup \{K_i\Gamma : i \in D\}\big). \end{aligned}$$

Suppose $B, \Sigma \in \mathcal{V}_2(D)$ and $a, Q_Z, Q_x, \hat{Q}_x, \hat{B}, \bar{B}, Q_0 \in \mathcal{V}_1$ satisfy the formal identities

$$\begin{aligned} F_1 B &= a + ZQ_Z + XQ_x,\\ XQ_x &= \hat{Q}_x,\\ \Gamma a &= \Sigma,\\ X^{t+1}B &= \hat{B},\\ C_{\mathsf{Com}_e}B &= Z\bar{B},\\ B - L_0 &= (X-1)Q_0. \end{aligned} \tag{8}$$

Then $B \in \mathbb{Z}_p[X]$, $B(1) = 1$, $\deg B \le N - t$, and $B(\omega^i) = 0$ for every $i \in [N] \setminus ((A \cup D) \cap \mathsf{Com}_e)$. Consequently,

$$t \le |(A \cup D) \cap \mathsf{Com}_e|.$$

Proof. The last identity shows that B contains neither Γ nor any Θ_j since $Q_0 \in \mathcal{V}_1$. The part of $\mathcal{V}_2(D)$ containing none of these indeterminates consists only of polynomials in X, so $B \in \mathbb{Z}_p[X]$. Evaluating the last identity at $X = 1$ gives $B(1) = L_0(1) = 1$. Since $\hat{B} \in \mathcal{V}_1$ and $X^{t+1}B = \hat{B}$ contains only X, its degree is at most $N+1$, giving $\deg B \le N - t$. Evaluating the fifth identity at ω^i gives $B(\omega^i) = 0$ for $i \notin \mathsf{Com}'_e$, since $C_{\mathsf{Com}_e}(\omega^i) \neq 0$ there.

From $\Gamma a = \Sigma$ and the structure of $\mathcal{V}_2(D)$, we have $a = a_0 + \sum_{i\in D} a_i K_i$ for constants $a_0, a_i \in \mathbb{Z}_p$. In particular, a has zero coefficient on K_i for $i \in H \setminus D$.

Fix $i \in H \setminus D$, and let $U_i, V_i, W_i \in \mathbb{Z}_p[X]$ be the coefficients of K_i in $Q_Z, Q_x, \hat{Q}_x$, respectively. Membership in $\mathcal{V}_1$ gives $\deg W_i \le N$. Taking coefficients in the first two identities gives $BL_i = ZU_i + XV_i$ and $XV_i = W_i$. Since $BL_i \equiv B(\omega^i)L_i \pmod Z$, we have $Z | B(\omega^i)L_i - XV_i$. However, $\deg(B(\omega^i)L_i - XV_i) \le N$ and $\deg Z = N+1$, which means that $B(\omega^i)L_i - XV_i = 0$. Evaluation at $X = 0$ gives $B(\omega^i)L_i(0) = 0$. Since $L_i(0) = 1/(N+1) \neq 0$, we obtain $B(\omega^i) = 0$.

Thus B vanishes at all real-party positions outside $(A \cup D) \cap \mathsf{Com}_e$. Since $B(1) = 1$, it is nonzero, and counting these distinct roots gives

$$N - |(A \cup D) \cap \mathsf{Com}_e| \le \deg B \le N - t.$$

This proves the claimed threshold bound. □

B.7 Unforgeability

In this section, we prove that under the q-DLOG assumption, Construction 2 satisfies the unforgeability (Definition A.2) against algebraic adversaries in the programmable random-oracle model.

Proof. Let $H = [N] \setminus A$. Simulate all honest registration proofs and all honest partial-signature proofs by Lemma B.2. If the number of signing queries is at most q_{sig}, then $L = N + q_{\mathrm{sig}}$ is an upper bound on the number of simulated proofs.

By Lemma B.3 and Lemma B.4, we may exclude negligible-probability executions in which a relevant accepted algebraic relation is not a formal identity or an accepted corrupted public key has a nonconstant formal exponent.

Set $\kappa_i = 0$ for all rejected corrupted registrations, and the aggregate public-key commitment will have formal exponent

$$F_1 = L_0(X) + \sum_{i\in H} K_i L_i(X) + \sum_{i\in A} \kappa_i L_i(X).$$

Consider a forgery on (m^*, e). Let Γ denote the formal exponent of $\mathcal{H}_2(m^*, e)$, if it has not previously been queried, it is queried during verification. Let $D \subseteq H$ be the set of honest parties queried on m^*, and let Θ_j denote all other exponents output by $\mathcal{H}_2$.

Use the spaces $\mathcal{V}_1$ and $\mathcal{V}_2(D)$ defined in Lemma B.5. All $\mathbb{G}_1$ and $\mathbb{G}_2$ outputs have formal exponents in the corresponding spaces. $\mathcal{V}_2(D)$ deliberately includes all honest non-target shares, whether or not queried. This can only enlarge the adversary's capabilities, and simulated proof commitments do not enlarge these spaces.

The formal exponents of the aggregated signature can be written as

$$(B, -a, -Q_Z, -Q_x, \hat{Q}_x, \Sigma, -\hat{B}, -\bar{B}, -Q_0).$$

Therefore, $B, \Sigma \in \mathcal{V}_2(D)$ and the other components belong to $\mathcal{V}_1$. Acceptance of the forged signature implies the six formal identities in Equation 8. Lemma B.5 then gives

$$t \le |(A \cup D) \cap \mathsf{Com}_e|,$$

contradicting the winning condition.

Finally, combining the related bounds yields

$$\Pr[\mathsf{Forge} = 1] \le O\left(\frac{L(q_0 + L) + q_0 + |A|}{p}\right) + \mathsf{negl}_{q\text{-DLOG}}(\lambda),$$

which is negligible. □

C DEFINITIONS OF CORRECTNESS AND SEMANTIC SECURITY FOR C-STE

In both games below, all corrupted registrations are verified during Preprocess. The partial-decryption oracle in the correctness game is enabled only after this verification; the semantic-security game provides no partial-decryption oracle. Random-oracle queries remain available during registration.

Definition C.1 (Correctness). The c-STE scheme satisfies correctness if, for any PPT adversary $\mathcal{A}$ and any $N = \mathrm{poly}(\lambda)$, the output of the correctness game defined below is 1 with probability $\le \mathsf{negl}(\lambda)$.

1. The challenger runs $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives CRS to $\mathcal{A}$.
2. $\mathcal{A}$ picks a subset A of corrupt parties such that $A \subseteq [N]$, and A remains fixed throughout the game.
3. For all $i \in [N] \setminus A$, the challenger samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, computes $\mathsf{hint}_i \leftarrow \mathsf{HintGen}(\mathsf{CRS}, \mathsf{sk}_i, N, i)$, and gives only $(\mathsf{pk}_i, \mathsf{hint}_i)$ to $\mathcal{A}$.
4. For all $i \in A$, the adversary returns a public key pk_i and a hint hint_i to the challenger.
5. The challenger invokes the preprocessing as $(\mathsf{ak}, \mathsf{ek}) \leftarrow \mathsf{Preprocess}(\mathsf{CRS}, \{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]})$ and the outputs are given to $\mathcal{A}$. Let $V \subseteq [N]$ be the set of parties whose hints are valid.
6. $\mathcal{A}$ picks an epoch e.
7. The challenger obtains the committee Com_e with its commitment by $B_e \leftarrow \mathsf{EpochRepr}(e)$ and gives it to $\mathcal{A}$.
8. $\mathcal{A}$ is given access to the partial decryption oracle of the honest parties for the fixed epoch e. This oracle returns $\perp$ if $i \notin ([N] \setminus A) \cap V \cap \mathsf{Com}_e$ or if $\widetilde{\mathsf{ct}}$ is malformed. Otherwise, it returns $\mathsf{PartDec}(\mathsf{sk}_i, \widetilde{\mathsf{ct}}, e)$, including the Eq-DL proof. This can be run polynomially many times by $\mathcal{A}$ until it submits the malicious target shares in Step 10, including after receiving the target ciphertext in Step 9. It picks a message msg and a threshold $1 \le t \le |\mathsf{Com}_e|$.
9. The challenger generates the honest ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{msg}, t, e)$ and sends it to $\mathcal{A}$.
10. $\mathcal{A}$ prepares the partial decryptions $\{\sigma_i\}_{i\in S_1}$ for some subset of malicious parties $S_1 \subseteq A$. Let $S_1' \subseteq S_1$ be the subset of indices whose maliciously generated partial decryptions verify under PartVerify.
11. $\mathcal{A}$ may also request a subset of honest parties $S_2 \subseteq \mathsf{Com}_e \setminus A$ for partial decryptions (including their Eq-DL proofs), which are returned by computing $\sigma_i \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, e)$.
12. Let $S = (S_1' \cup S_2) \cap V \cap \mathsf{Com}_e$, and the game outputs 0 and terminates if $|S| < t$. Otherwise, the challenger computes the aggregated decryption as $\mathsf{msg}' \leftarrow \mathsf{DecAggr}(\mathsf{CRS}, \mathsf{ak}, \mathsf{ct}, \{\sigma_i\}_{i\in S}, t, e)$.
13. The output of the game is 1 if $\mathsf{msg}' \neq \mathsf{msg}$ or $\mathsf{msg}' = \perp$. Otherwise, the output is 0.

Definition C.2 (Security). The c-STE scheme satisfies semantic security if, for any PPT adversary $\mathcal{A}$ and any $N = \mathrm{poly}(\lambda)$, the output of the security game defined below is 1 with probability $\le \frac{1}{2} + \mathsf{negl}(\lambda)$.

1. The challenger runs $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives CRS to $\mathcal{A}$.
2. $\mathcal{A}$ picks a subset A of corrupt parties such that $A \subseteq [N]$, and A remains fixed throughout the game.
3. For all $i \in [N] \setminus A$, the challenger samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, computes $\mathsf{hint}_i \leftarrow \mathsf{HintGen}(\mathsf{CRS}, \mathsf{sk}_i, N, i)$, and gives only $(\mathsf{pk}_i, \mathsf{hint}_i)$ to $\mathcal{A}$.
4. For all $i \in A$, the adversary returns a public key pk_i and a hint hint_i to the challenger.
5. The challenger invokes the preprocessing as $(\mathsf{ak}, \mathsf{ek}) \leftarrow \mathsf{Preprocess}(\mathsf{CRS}, \{\mathsf{hint}_i, \mathsf{pk}_i\}_{i\in[N]})$ and the outputs are given to $\mathcal{A}$. Let $V \subseteq [N]$ be the set of parties whose hints are valid.
6. $\mathcal{A}$ picks an epoch e.
7. The challenger obtains the committee Com_e with its commitment by $B_e \leftarrow \mathsf{EpochRepr}(e)$ and gives it to $\mathcal{A}$.
8. $\mathcal{A}$ picks messages $\mathsf{msg}_0, \mathsf{msg}_1 \in \mathbb{G}_T$ and a threshold $1 \le t \le |\mathsf{Com}_e|$.
9. The challenger picks a bit $b \leftarrow \{0, 1\}$, generates a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{msg}_b, t, e)$ and sends it to $\mathcal{A}$.
10. $\mathcal{A}$ outputs a bit b' and wins the semantic security game if $t > |A \cap \mathsf{Com}_e|$ and $b' = b$, in which case the output of the game is 1. Otherwise, the output is 0.

D SECURITY ANALYSIS OF C-STE

D.1 Correctness

In this section, we prove that Construction 3 satisfies the correctness (Definition C.1).

Proof. For each registered public key, let x_i be its discrete logarithm with respect to $[1]_1$. The hint-consistency checks imply the required hint relations exactly.

By the soundness of the Eq-DL protocol, every accepted partial decryption of the target ciphertext satisfies $s_i = x_i \mathcal{H}_2(u, e)$ except with negligible probability. This remains true after polynomially many prior partial-decryption observations in the correctness game: honest proofs concern true statements, while any newly accepted false statement contradicts the soundness of Eq-DL protocol except with negligible probability.

When all accepted target shares satisfy their required relations, we have $\boldsymbol{A} \circ \boldsymbol{w} = \boldsymbol{b}$ from the aggregation. Therefore, for an honestly generated ciphertext we have

$$\begin{aligned} \mathsf{msg}^* &= ct_3 - \boldsymbol{ct}_2 \circ \boldsymbol{w} \\ &= \boldsymbol{s}^\top \boldsymbol{b} + \mathsf{msg} - (\boldsymbol{s}^\top \boldsymbol{A}) \circ \boldsymbol{w} \\ &= \mathsf{msg} \end{aligned}$$

and Construction 3 satisfies correctness with probability $1 - \mathsf{negl}(\lambda)$. □

D.2 Generic Group Model

Our proof for the security of our scheme relies on the Generic Group Model [12, 15, 34] (GGM). Elements of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are represented by independently typed random opaque handles. The adversary may perform linear group operations and invoke the pairing. No interface converts $\mathbb{G}_T$ elements into $\mathbb{G}_1, \mathbb{G}_2$ elements or converts between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Interfaces accepting group elements require valid, previously issued handles of the correct type. No additional random-group-element sampling interface is provided. The adversary may query $\mathcal{H}_2$ and submit arbitrary bit strings to the random oracles, but arbitrary bit strings are not thereby valid group handles.

The ideal execution represents group exponents by formal polynomials, and group equality is determined by polynomial identity. The simulator may program $\mathcal{H}_0$.

Let Q bound the number of formal expressions processed during the execution. If their total degrees are at most d, the probability that distinct same-group expressions evaluate to equal elements is

$$\epsilon_{\text{coll}} = O(Q^2 d/p).$$

This follows by fixing the ideal execution's auxiliary randomness, applying the Schwartz-Zippel bound to each pair of distinct polynomials, and taking a union bound.

D.3 Accepted Registrations in the Ideal Execution

Honest registration proofs are simulated using Lemma B.2. No partial-decryption proofs are simulated in the semantic-security game, because Definition C.2 provides no partial-decryption oracle.

In the ideal execution, an accepted corrupted registration proof for a public-key polynomial P implies $R + cP = s$, where R is the first commitment polynomial. If P is nonconstant, a nonconstant monomial coefficient gives $r_m + cp_m = 0, p_m \neq 0$.

For a fixed fresh random-oracle input, at most one challenge satisfies this condition. The registration statement at position i uses $B = [L_i(\tau)]_1$ as its second base. This base is included in the Fiat-Shamir hash. For an honest position $j \neq i$, $L_i(X) \neq L_j(X)$ since $L_i(\omega^i) = 1$ and $L_j(\omega^i) = 0$. Thus, in the ideal execution these bases have distinct handles, and a corrupted registration input cannot equal an honestly programmed registration input. In the real generic execution, an accidental equality of these evaluated bases is already covered by ϵ_{coll}. Preprocessing verifies all corrupted registrations before any oracle for honest partial decryptions is enabled.

Therefore, the probability that any corrupted registration proof accepts a nonconstant public-key polynomial is at most

$$\epsilon_{\text{reg}} \leq (q_0 + |A|)/p.$$

Outside this event, accepted corrupted registrations have constant public-key polynomials κ_i, and the hint checks determine all their algebraic hint components.

D.4 Security

In this section, we prove that Construction 3 satisfies the semantic security (Definition C.2) against generic adversaries in the generic bilinear group model with programmable random oracles.

Proof. Use q_0 as defined in the c-STS security analysis, including all adversary, verifier, and challenger queries to $\mathcal{H}_0$ and excluding simulator programming. Let $q_{\mathcal{H}_2}$ bound all queries to $\mathcal{H}_2$, including verifier and challenger queries. Every abort in the following games terminates the game with output 0. Without changing the winning probability, we modify the game to terminate with output 0 immediately after the threshold is selected if $t \leq |A \cap \mathsf{Com}_e|$.

Game$_0$: This is the semantic security game of Definition C.2, with the equivalent early termination rule specified above.

Game$_1$: Simulate all honest registration proofs. By Lemma B.2 we have

$$|\Pr[\mathbf{Game}_0 = 1] - \Pr[\mathbf{Game}_1 = 1]| \leq O\left(\frac{N(q_0 + N)}{p}\right).$$

Game$_2$: Abort if the challenge nonce u^* satisfies that (u^*, e) has previously been queried to $\mathcal{H}_2$. Since e is fixed before the nonce is sampled, we have

$$|\Pr[\mathbf{Game}_1 = 1] - \Pr[\mathbf{Game}_2 = 1]| \leq q_{\mathcal{H}_2}/p.$$

Game$_3$: Replace the real generic execution with the ideal formal-polynomial execution, and abort if a nonconstant public-key polynomial (corresponding to a corrupted registration proof) is accepted. The relevant exponent polynomials have degree $O(N)$, including the registration-verification expressions. Therefore, we have

$$|\Pr[\mathbf{Game}_2 = 1] - \Pr[\mathbf{Game}_3 = 1]| \leq O(Q^2 N/p) + (q_0 + |A|)/p.$$

Let $H = [N] \setminus A$. Let X represent τ, K_i the honest secret keys, Γ the fresh challenge-tag exponent, and $\boldsymbol{S} = (S_1, \ldots, S_6)$ the challenge encryption randomness. Let Θ_j represent all other $\mathcal{H}_2$ outputs.

Let

$$\begin{aligned} F_1 &= L_0(X) + \sum_{i \in H} K_i L_i(X) + \sum_{i \in A} \kappa_i L_i(X), \\ F_2 &= Z(X), \\ F_3 &= X - 1, \\ F_4 &= \prod_{i \in \mathsf{Com}'_e} (X - \omega^i), \end{aligned}$$

where $\kappa_i = 0$ for rejected corrupted registrations.

Before adding the challenge ciphertext, the $\mathbb{G}_1$ and $\mathbb{G}_2$ elements are contained in

$$\mathcal{U}_1 = \text{span}\left(\{X^j\}_{j=0}^{N+1}, \{K_i X^j : i \in H,\ 0 \le j \le N\}\right),$$
$$\mathcal{U}_2 = \text{span}\left(\{X^j\}_{j=0}^{N+1}, \Gamma, \{\Theta_j\}_j\right).$$

Including hash variables first introduced after the challenge in these spaces only enlarges the spaces used in the analysis.

After adding the challenge ciphertext, the $\mathbb{G}_1$ and $\mathbb{G}_2$ elements are contained in

$$\mathcal{L}_1 = \text{span}\left(\mathcal{U}_1, S_3, S_1F_1 + S_4X^{t+1} + S_5F_4 + S_6\right),$$
$$\mathcal{L}_2 = \text{span}\big(\mathcal{U}_2, S_1 + \Gamma S_3, S_1F_2, (S_1 + S_2)X, S_2, S_4, S_5F_2, S_6F_3\big).$$

Let $\mathcal{T}_{\text{pre}}$ contain all $\mathbb{G}_T$ expressions obtained before the challenge, including the message exponents m_0, m_1. These expressions contain no $\boldsymbol{S}$. Let

$$\mathcal{W} = \text{span}\left(\mathcal{L}_1\mathcal{L}_2 \cup \mathcal{T}_{\text{pre}}\right).$$

This space contains every $\mathbb{G}_T$ exponent obtainable without using ct_3. The challenge mask is $f = S_6L_0(X)$.

We claim that $f \notin \mathcal{W}$. Otherwise, taking the homogeneous degree-1 component in $\boldsymbol{S}$ in an expression for f gives

$$S_6L_0(X) = \sum_{j=1}^{9} \boldsymbol{S}^\top \boldsymbol{a}_j g_j = \boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{g},$$

where $g_1, g_6 \in \mathcal{U}_2$, and the other components belong to $\mathcal{U}_1$. Thus $\boldsymbol{A}\boldsymbol{g} = \boldsymbol{b}$ holds as a formal polynomial identity. To apply Lemma B.5, set $D = \varnothing$ and identify

$$(B, a, Q_Z, Q_x, \hat{Q}_x) = (g_1, -g_2, -g_3, -g_4, g_5),$$
$$(\Sigma, \hat{B}, \bar{B}, Q_0) = (g_6, -g_7, -g_8, -g_9).$$

We have $\mathcal{U}_1 = \mathcal{V}_1$ and $\mathcal{U}_2 \subseteq \mathcal{V}_2(\varnothing)$. The six rows of $\boldsymbol{A}\boldsymbol{g} = \boldsymbol{b}$ are exactly Equation 8 under this identification. Lemma B.5 therefore gives $t \le |A \cap \mathsf{Com}_e|$, contradicting the winning condition. This application uses only the formal identities in the ideal generic execution.

Game$_4$: Replace $[f + m_b]_T$ in the challenge ciphertext by $[R + m_b]_T$, where R is a new independent formal variable.

Because $\mathbb{G}_T$ elements admit only linear operations and cannot be converted to $\mathbb{G}_1$ or $\mathbb{G}_2$ elements, every $\mathbb{G}_T$ expression has the form $a(f + m_b) + P$ or $a(R + m_b) + P$, with $a \in \mathbb{Z}_p$ and $P \in \mathcal{W}$.

Since $m_b \in \mathcal{W}$ and $f, R \notin \mathcal{W}$, the correspondence

$$a(f + m_b) + P \longleftrightarrow a(R + m_b) + P$$

preserves group operations and equality tests. Therefore, by assigning corresponding expressions to the same random handles, giving identical responses to identical random-oracle inputs, and assigning corresponding fresh variables Θ_j to newly queried non-challenge hash inputs, the complete adaptive executions can be coupled, yielding

$$\Pr[\mathbf{Game}_3 = 1] = \Pr[\mathbf{Game}_4 = 1].$$

Similarly,

$$a(R + m_0) + P \longleftrightarrow a(R + m_1) + P$$

couples $\mathbf{Game}_4$ for the two challenge bits. Therefore

$$\Pr[\mathbf{Game}_4 = 1] \le \frac{1}{2}.$$

In summary, we have

$$\Pr[\mathbf{Game}_0 = 1] \le \frac{1}{2} + O\left(\frac{N(q_0 + N) + q_{\mathcal{H}_2} + Q^2N + q_0 + |A|}{p}\right),$$

where the additive error is negligible. □

E OPEN SCIENCE

To support the reproducibility of our work, we have provided all necessary artifacts through the following website:

https://github.com/Gorachya/c-STS-STE

This repository includes the source code for implementing our schemes, and our evaluation results can be obtained by following the commands in `README.md`.

F ETHICAL CONSIDERATIONS

This work focuses on the theoretical design and practical implementation of advanced cryptographic protocols, specifically silent threshold signature (STS) and encryption (STE) schemes. Our research does not involve any human subjects, the collection or processing of user data, nor the exploitation or analysis of vulnerabilities in deployed real-world systems. Consequently, we consider this work to have no related ethical concerns.