Constant-Online PVSS from CCA2-Secure Threshold Encryption: A Generic Framework

Liang Zhang[1*], Dongliang Cai[2*], Haibin Kan[2], Jiheng Zhang[1], and Moti Yung[3]

[1] The Hong Kong University of Science and Technology
[2] Fudan University
[3] Google LLC and Columbia University

Abstract. Publicly Verifiable Secret Sharing (PVSS) is widely used in distributed systems. Existing schemes usually incur at least $O(n)$ online cost: the dealer encrypts, proves, and publishes n shareholder-dependent objects, which public verification must process. In this work, we present a generic framework that transforms publicly verifiable CCA2-secure threshold encryption (CCATE) into *constant-online* PVSS, with distribution and public- verification costs independent of the number of shareholders. The framework moves the share-generation work into a reusable setup phase: once threshold keys and public verification material are fixed, online sharing amounts to a single publicly verifiable threshold encryption. We instantiate the framework with two CCATE constructions: 1) a pairing-free instantiation using standard Threshold ElGamal encryption under a committee-based setup assumption; and 2) a silent-setup scheme leveraging non-interactive key generation via a Power-of-Tau ceremony, eliminating inter-party coordination during setup. Furthermore, we discuss epoch-based membership updates under the corresponding setup assumptions, clarifying the security boundary of reconfiguration. The resulting schemes incur higher setup costs, but the critical online distribution and public-verification phases are constant-size and constant-time. This trade-off is particularly useful when setup can be amortized over many PVSS instances, as in blockchain and distributed-system deployments.

Keywords: PVSS · CCA2-Secure Threshold Encryption · Public Verifiability · Constant-Online Complexity · Distributed Systems · Blockchains

1 Introduction

Scaling decentralized systems and blockchains raises a recurring efficiency challenge: as the number of participants grows, many cryptographic protocols inherit costs that scale linearly with the committee size. This is natural when responsibilities and data are distributed across many parties, but it is also a bottleneck for primitives whose outputs must be publicly stored and repeatedly verified. This motivates the question we study in this work: can a secret-sharing

* Liang Zhang and Dongliang Cai contributed equally to this work.

primitive used in distributed systems achieve constant *online* distribution and public-verification complexity, while still supporting recovery by a threshold of n shareholders?

At first glance, the goal appears paradoxical. If a dealer explicitly creates and publishes n fresh shares in every instance, then an $O(n)$ online cost is unavoidable. We therefore consider an amortized computational setting: shareholder-dependent work may be moved into a reusable, publicly auditable setup phase, and subsequent sharing instances should be as small as possible. We investigate this question for publicly verifiable secret sharing (PVSS) [1,2,3,4,5,6,7], where public verification may be performed independently by many parties.

Our approach is to shift expensive shareholder-dependent operations to setup and use threshold encryption as the online sharing mechanism. PVSS and threshold encryption share a common recovery structure: a secret is recoverable only with sufficiently many parties. At the same time, threshold encryption offers a constant-size ciphertext. Building on this observation, we define a publicly verifiable CCA2-secure threshold-encryption interface, called CCATE, and show how it generically yields constant-online PVSS.

This construction direction deserves clarification. Threshold encryption [8,9] is often built from secret sharing, for example by sharing the decryption key. In contrast, we use a verifiable threshold encryption scheme as a primitive for constructing PVSS. This is not circular: the long-term recovery capability is distributed once during setup, while the online sharing of a new secret is simply an encryption under the resulting threshold public key. The constant-size ciphertext compresses the online public sharing information, and the assoaciated NIZK proofs of plaintext knowledge provide public verifiability. Thus the $O(1)$ online complexity is obtained by an explicit setup/online trade-off, not by information-theoretically compressing n independent fresh shares.

To validate the framework, we instantiate it with two CCATE constructions. 1) *A Pairing-Free Construction:* We combine Threshold ElGamal encryption [8,10] with a committee-based setup assumption that provides publicly auditable threshold keys, and augment the construction with NIZK proofs to achieve the CCATE properties. 2) *A Silent-Setup Construction:* We build on the silent-setup threshold-encryption paradigm [9], using Power-of-Tau public parameters [11] to reduce setup interaction at the cost of heavier public-key material and pairing-based verification. Both PVSS constructions achieve online distribution and public verification whose size and running time are independent of the number of shareholders.

PVSS is particularly suitable for blockchains [12,13] and decentralized systems [14,15], where setup transcripts and distribution data are publicly stored and verification can be performed by independent validators or smart contracts. The decoupling of share distribution from reconstruction also makes PVSS useful as a threshold-opening commitment primitive [16]. Our constructions are therefore most directly applicable to use cases that consume threshold openings, including distributed randomness beacons (DRBs) [4,5,17,18,19], E-voting [2,20], On-chain witness encryption [21], blockchain randomness and consensus appli-

cations [22,23], and frameworks where PVSS commitments are repeatedly produced and consumed [24]. Epoch-based setup renewal can support changing participation, but our security claims are per epoch rather than fully adaptive across arbitrary membership changes.

From Attempts to Construction. Our work originated from the goal of obtaining sublinear, ideally constant, public verification in PVSS, where the verifier traditionally processes n individually encrypted shares. Verifying n distinct online objects with less than $O(n)$ work is implausible if these objects are freshly generated in each instance. This led us to look for a way to place the threshold structure into reusable setup rather than into every online sharing. This viewpoint is reminiscent of fuzzy identity-based encryption (FIBE) [25] and KP-ABE [26], where threshold access is enforced by embedding secret-sharing structure into key generation rather than into each ciphertext. This perspective suggests that the setup can carry the shareholder-dependent structure, while the online ciphertext remains compact. These attempts eventually led us to CCA2-secure threshold encryption as the right abstraction: constant-size ciphertexts, threshold recovery, and public validity via NIZK proofs. The resulting PVSS schemes decouple reusable setup from online sharing and achieve constant-online dealer work and public verification for each PVSS instance.

1.1 Contributions

We present a generic framework for constructing PVSS schemes with *constant online complexity*: after setup, the dealer's online work and public-verification cost are independent of the number of shareholders n. The main contributions are as follows.

1. **A generic constant-online PVSS framework.** We formalize the threshold-encryption interface needed by the transformation and show that any publicly verifiable CCA2-secure threshold encryption scheme yields a secure PVSS scheme. The transformation makes the setup/online trade-off explicit: shareholder-dependent work is moved to reusable setup, while each online sharing consists of one verifiable ciphertext.
2. **Two concrete instantiations.** We give two CCATE constructions realizing the framework:
   - a **pairing-free construction** based on threshold ElGamal encryption and a committee-based setup assumption for publicly auditable threshold keys; and
   - a **silent-setup construction** based on Power-of-Tau public parameters and silent-setup threshold encryption [11,9].

   The second instantiation trades heavier local computation and pairing checks for reduced setup interaction.
3. **Epoch-based reconfiguration and evaluation.** Both constructions support membership changes by starting a new epoch with a fresh, publicly auditable setup transcript. Security is stated per epoch under the corresponding setup assumptions. We also provide an implementation showing

that the online distribution and verification costs are essentially constant, compared to existing schemes [6].

2 Preliminaries

Let $\mathbb{G}$ be a cyclic group of prime order p with generator g, and let $\mathbb{Z}_p$ denote the finite field of integers modulo p. Throughout the paper, secrets are exponents in $\mathbb{Z}_p$, while reconstruction outputs their group encodings in $\mathbb{G}$. We capture this consistency condition by the relation

$$\mathcal{PK} = \{(s, S) \in \mathbb{Z}_p \times \mathbb{G} : S = g^s\}.$$

2.1 Polynomials over $\mathbb{Z}_p$

Secret Sharing Given a secret $s \in \mathbb{Z}_p$, choose random coefficients $a_1, \ldots, a_{t-1} \in \mathbb{Z}_p$ and define $f(x) = s + \sum_{i=1}^{t-1} a_i x^i \pmod p$. The share of party P_i is the point $(x_i, f(x_i))$ for a distinct nonzero evaluation point x_i. Shamir secret sharing provides two standard properties:

- Robustness: any t or more shares can reconstruct s;
- Perfect secrecy: fewer than t shares reveal no information about s.

Lagrange Interpolation (on $\mathbb{G}$) To reconstruct $s = f(0)$ from any t shares, let $\{(x_j, y_j)\}_{j=1}^t$ be a qualified set with distinct nonzero points. The secret is recovered as $s = \sum_{j=1}^t y_j L_j(0), L_j(x) = \prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{x - x_k}{x_j - x_k}$, where $L_j(x)$ is the Lagrange basis polynomial for the point x_j. We write Intpl for this interpolation algorithm. We use the following standard properties:

- Linearity: for shares $y_j = \sum_i y_{i,j}$ over the same evaluation points, $\mathsf{Intpl}(\{(x_j, y_j)\}_{j=1}^t) = \sum_i \mathsf{Intpl}(\{(x_j, y_{i,j})\}_{j=1}^t)$.
- Uniqueness: any t points determine a unique polynomial of degree at most $t-1$, and hence a unique value at 0.

The same interpolation can be applied in the exponent. Given group-encoded shares $\{(x_j, g^{y_j})\}_{j=1}^t$, one obtains $g^s = \prod_{j=1}^t \left(g^{y_j}\right)^{L_j(0)}$.

2.2 Sigma Protocols and NIZK Proofs

A Sigma protocol is a three-move public-coin proof of knowledge for an NP relation $R \subseteq \mathcal{Y} \times \mathcal{W}$, where $y \in \mathcal{Y}$ is the statement and $w \in \mathcal{W}$ is the witness. The prover first sends a commitment a, the verifier samples a challenge $c \in \mathcal{C}$, and the prover replies with a response z. The verifier then applies a deterministic predicate $\mathsf{V}(y, a, c, z)$.

We use Sigma protocols with the following standard properties.

- **Completeness**: for every $(y, w) \in R$, an honest execution between $P(w, y)$ and $V(y)$ is accepted with probability one.
- **Special soundness**: there is an efficient extractor that, given two accepting transcripts (a, c, z) and (a, c', z') for the same statement y and commitment a, with $c \neq c'$, outputs a witness w such that $(y, w) \in R$.
- **Honest-verifier zero knowledge**: there is an efficient simulator that, on input a statement y and a challenge $c \in \mathcal{C}$, outputs a transcript (a, c, z) whose distribution is indistinguishable from that of an honest execution with challenge c.

In the random-oracle model, we use the labelled Fiat–Shamir transform [27], where the challenge is $c = H(\ell, y, a)$ for a public context label ℓ. Unless stated otherwise, $H : \{0,1\}^* \to \mathbb{Z}_p$ and $|p| = \Omega(\lambda)$, so challenge collisions and guessing events are negligible. For each proof system, the label ℓ contains the protocol identifier, proof purpose, public parameters, and setup or epoch identifier. Thus labels separate DLEQ proofs from ciphertext-validity proofs and bind proofs to the intended setup instance. We use ℓ_{DLEQ} for partial-decryption DLEQ proofs, ℓ_{inst_1} for the ciphertext-validity proof of Instantiation I, and ℓ_{inst_2} for the ciphertext-validity proof of Instantiation II.

Remark 1 (Labelled simulation-sound extractability of Fiat–Shamir proofs). Correctness, public verifiability, and ordinary soundness use the standard Fiat–Shamir view above. The CCA2 reductions additionally require labelled simulation-sound online extractability, because the simulator may issue a simulated challenge proof and later extract witnesses from other accepting ciphertext proofs under the same label while answering partial-decryption queries. We instantiate ciphertext-validity proofs with labelled simulation-sound online-extractable Fiat–Shamir proofs in the ROM, e.g., via Fischlin-style online-extractable proofs [28].

Throughout the security proofs, $\epsilon_{\mathsf{ext}}(\lambda)$ denotes the total failure probability of the labelled simulation-sound online extractor for all accepting non-challenge ciphertext-validity proofs submitted by a PPT adversary. The term $\epsilon_{\mathsf{zk}}(\lambda)$ denotes the distinguishing advantage against the zero-knowledge simulation of the ciphertext-validity NIZK, and $\epsilon_{\mathsf{ro}}(\lambda)$ denotes the probability that a random-oracle point needed for Fiat–Shamir simulation has already been queried before it is programmed. These quantities are negligible in the security parameter under the stated proof-system assumptions; when clear from context, we omit the argument λ.

2.3 Discrete Logarithm Equality (DLEQ)

The Chaum–Pedersen protocol is a Sigma protocol for proving equality of discrete logarithms [29]. Let $\mathbb{G}$ be a prime-order group of order p, and let $g, h, G, T \in \mathbb{G}$. The DLEQ relation is $\mathcal{R}_{\mathsf{DLEQ}} = \{((g, h, G, T), s) : G = g^s \wedge T = h^s\}$. We use its Fiat–Shamir version, shown in Figure 1, as a publicly verifiable proof that the same exponent s links both pairs (g, G) and (h, T). We write $\epsilon_{\mathsf{dleq\text{-}snd}}(\lambda)$ for the total soundness failure probability that any invalid DLEQ proof is accepted

over all polynomially many verification attempts made by a PPT adversary. We write $\epsilon_{\mathsf{dleq\text{-}zk}}(\lambda)$ for the total distinguishing advantage between real DLEQ proofs and simulated DLEQ proofs over all polynomially many DLEQ proofs used in the CCA2 game hops. Both are negligible for the Fiat–Shamir DLEQ proof in the random-oracle model; we often omit (λ) in game-based bounds.

2.4 Publicly Verifiable Secret Sharing (PVSS)

Definition 1 (PVSS). *A PVSS scheme includes seven probabilistic polynomial-time (PPT) algorithms (*PVSS.Setup*,* PVSS.SetupVrf*,* PVSS.Share*,* PVSS.Verify*,* PVSS.PreRecon*,* PVSS.DecShareVrf*,* PVSS.Recon*):*

1. $(\mathsf{pp}, \{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[n]}) \leftarrow \mathsf{PVSS.Setup}(\lambda, t, n)$*: The algorithm outputs public parameters* pp *and a secret/public key pair* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *for each shareholder* P_i*.*
2. $bool \leftarrow \mathsf{PVSS.SetupVrf}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$*: Given* pp *and the public keys, verify that they define a valid setup for the PVSS instance family.*
3. $(C, \pi_s) \leftarrow \mathsf{PVSS.Share}(s, \mathsf{pp})$*: Given a secret exponent* s *from the secret space, the dealer outputs public distribution data* C *and a proof* π_s*. The value reconstructed from a valid sharing is a group-encoded secret* S *such that* $(s, S) \in \mathcal{PK}$*.*
4. $bool \leftarrow \mathsf{PVSS.Verify}(C, \pi_s)$*: Any party can check whether the public distribution data and proof are valid.*
5. $(D_i, \pi_{\mathsf{sk}_i}) \leftarrow \mathsf{PVSS.PreRecon}((C, \pi_s), \mathsf{sk}_i)$*: Shareholder* P_i *first verifies* (C, π_s)*. If it is valid,* P_i *uses* sk_i *to produce a reconstruction share* D_i *and a proof* π_{sk_i}*; otherwise it returns* $\perp$*.*
6. $bool \leftarrow \mathsf{PVSS.DecShareVrf}((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)$*: Anyone can check whether* D_i *is a valid reconstruction share for* C *under public key* pk_i*.*

Functionality Non-interactive DLEQ

$\pi_s \leftarrow DLEQ(g, h, G, T, s)$

$r \xleftarrow{\$} \mathbb{Z}_p$
$A = g^r,\ B = h^r$
$c = H(\ell_{\mathsf{DLEQ}}, g, h, G, T, A, B)$
$z = r + c \cdot s$
$\pi_s = (A, B, c, z)$

$bool \leftarrow DLEQVrf(g, h, G, T, \pi_s)$

parse $\pi_s = (A, B, c, z)$
$c' \leftarrow H(\ell_{\mathsf{DLEQ}}, g, h, G, T, A, B)$
return $c' \stackrel{?}{=} c$ and $g^z \stackrel{?}{=} AG^c$ and $h^z \stackrel{?}{=} BT^c$

Fig. 1. DLEQ proof generation and verification

7. $S \leftarrow \mathsf{PVSS.Recon}(C, \{(i, D_i)\}_{i \in Q})$*: Given* C *and a set of valid indexed reconstruction shares* $\{(i, D_i)\}_{i \in Q}$ *with* $|Q| \geq t$*, output the reconstructed group element* S*, such that* $(s, S) \in \mathcal{PK}$*.*

A PVSS scheme satisfies the properties of correctness, public verifiability, and IND1-Secrecy.

• ***Correctness*** If the dealer and shareholders follow the protocol, then any qualified set of at least t valid reconstruction shares recovers the same group-encoded secret S with $(s, S) \in \mathcal{PK}$.

• ***Public Verifiability*** Anyone can verify the public setup output, the dealer's distribution data, and each reconstruction share using only public information.

• ***IND1-Secrecy*** IND1-Secrecy guarantees that an adversary, given all public parameters and corrupting fewer than t shareholders, cannot distinguish a sharing of one chosen secret from a sharing of another chosen secret except with negligible advantage. We use the following game.

1. The challenger runs $\mathsf{PVSS.Setup}(\lambda, t, n)$, gives pp and all public keys to $\mathcal{A}$, and keeps the secret keys private.
2. $\mathcal{A}$ corrupts a set $W \subseteq [n]$ with $|W| \leq t - 1$ and obtains $\{\mathsf{sk}_i\}_{i \in W}$.
3. $\mathcal{A}$ outputs two secrets x_0, x_1 from the secret space. The challenger samples $b \xleftarrow{\$} \{0, 1\}$, runs $(C^*, \pi^*) \leftarrow \mathsf{PVSS.Share}(x_b, \mathsf{pp})$, and gives (C^*, π^*) to $\mathcal{A}$.
4. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathsf{ind1}}_{\mathsf{PVSS},\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

3 A Generic Constant-Online PVSS Framework from CCATE

Public-key encryption schemes that are IND-CPA secure under standard hardness assumptions can often be strengthened against chosen-ciphertext attacks by adding mechanisms that make invalid or adversarially modified ciphertexts detectable by the decryption algorithm. A classical route is the Naor–Yung paradigm [30], which encrypts the same message twice and attaches a NIZK proof of plaintext consistency; the proof-based consistency check prevents an adversary from obtaining useful decryptions of malformed ciphertexts. Another widely used route is the Fujisaki–Okamoto (FO) transformation [31]. In the random-oracle model, FO derives encryption randomness via hash functions and enforces consistency between the ciphertext and the underlying randomness during decryption. Malformed or inconsistent ciphertexts are rejected in a reconciliation step, effectively neutralizing invalid decryption queries. Structure-based constructions such as Cramer–Shoup encryption [32] achieve CCA2 security by embedding algebraic validity checks directly into the ciphertext. Our use of NIZK proofs is proof-based in spirit, but it is not a Naor–Yung double-encryption transform. Instead, we derive NIZK proofs from Sigma protocols [33] via the Fiat–Shamir transform [27] and use them as public well-formedness checks for ciphertexts and decryption shares; invalid objects are rejected before decryption or reconstruction is attempted.

3.1 CCATE Definition and Security Properties

A CCA2-secure threshold encryption (CCATE) scheme, adapted from [34], is defined by the following algorithms. This paper focuses on CCATE schemes instantiated over prime-order elliptic-curve groups. Public parameters pp are implicit inputs to all algorithms below.

1. $(\mathsf{pp}, \{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[n]}) \leftarrow \mathtt{Setup}(\lambda, t, n)$: given a security parameter λ and integers $1 \le t \le n$, this algorithm outputs public parameters pp and a decryption key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ for each party P_i.
2. $\mathit{bool} \leftarrow \mathtt{VerifySetup}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$: given public parameters pp and public keys $\{\mathsf{pk}_i\}_{i\in[n]}$, check whether the public keys are well formed and define the threshold encryption key encoded in pp.
3. $(C, \pi_s) \leftarrow \mathtt{Encrypt}(s, \mathsf{pp})$: given public parameters pp and a secret exponent s from the message space, output a randomized ciphertext C encrypting the group-encoded plaintext S such that $(s, S) \in \mathcal{PK}$, together with a proof π_s of ciphertext well-formedness.
4. $\mathit{bool} \leftarrow \mathtt{VerifyCT}(C, \pi_s)$: given a ciphertext C and a non-interactive proof π_s, output 1 if the pair passes the public ciphertext-validity check and 0 otherwise.
5. $(D_i, \pi_{\mathsf{sk}_i}) \leftarrow \mathtt{PartialDec}((C, \pi_s), \mathsf{sk}_i)$: given a ciphertext-proof pair (C, π_s) and P_i's secret key sk_i, first run $\mathtt{VerifyCT}(C, \pi_s)$. If verification accepts, output a partial decryption share D_i and a proof π_{sk_i} of correct partial decryption; otherwise return $\perp$.
6. $\mathit{bool} \leftarrow \mathtt{VerifyPD}((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)$: given P_i's purported decryption share D_i, proof π_{sk_i}, public key pk_i, and ciphertext C, output 1 if the share is valid and 0 otherwise.
7. $S \leftarrow \mathtt{Decrypt}(C, \{(i, D_i)\}_{i\in Q})$: given C and a set $Q \subseteq [n]$ with $|Q| \ge t$ valid indexed decryption shares, output the plaintext S. If fewer than t valid shares are provided, output $\perp$.

The security properties of a CCATE scheme are as follows.

Definition 2 (Correctness). *Let $\mathcal{S}$ be the secret space. For every $s \in \mathcal{S}$, every honest setup $(\mathsf{pp}, \{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[n]}) \leftarrow \mathtt{Setup}(\lambda, t, n)$ such that $\mathtt{VerifySetup}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]}) = \mathsf{true}$, every $(C, \pi_s) \leftarrow \mathtt{Encrypt}(s, \mathsf{pp})$, and every $Q \subseteq [n]$ with $|Q| \ge t$, let $(D_i, \pi_i) \leftarrow \mathtt{PartialDec}((C, \pi_s), \mathsf{sk}_i)$ for all $i \in Q$. Then*

$$\Pr\left[\begin{array}{l}\mathtt{VerifyCT}(C, \pi_s) = \mathsf{true},\ \forall i \in Q : \mathtt{VerifyPD}((D_i, \pi_i), \mathsf{pk}_i, C) = \mathsf{true}, \\ \textit{and } (s, \mathtt{Decrypt}(C, \{(i, D_i)\}_{i\in Q})) \in \mathcal{PK}\end{array}\right] = 1.$$

Definition 3 (Robustness). *For any honestly generated and publicly valid ciphertext (C, π_s), any PPT adversary that outputs two sets of accepted decryption shares $\mathcal{D} = \{(i, D_i, \pi_i)\}_{i\in Q}$ and $\mathcal{D}' = \{(i, D'_i, \pi'_i)\}_{i\in Q'}$, where $|Q|, |Q'| \ge t$ and all shares pass $\mathtt{VerifyPD}$ under their corresponding public keys, can make the two reconstructions differ only with negligible probability:*

$$\Pr\left[\mathtt{Decrypt}(C, \{(i, D_i)\}_{i\in Q}) \neq \mathtt{Decrypt}(C, \{(i, D'_i)\}_{i\in Q'})\right] \le \mathsf{negl}(\lambda).$$

Definition 4 (Public Verifiability). *The public outputs of setup, encryption, and partial decryption are publicly verifiable.*

- ***PV1*** *Setup validity: The public setup output* $(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$*, or the transcript that produces it, can be checked using* `VerifySetup` *without access to any secret key.*
- ***PV2*** *Ciphertext validity: A CCATE ciphertext-proof pair* (C, π_s) *can be checked by any party via* $\texttt{VerifyCT}(C, \pi_s)$*. Acceptance means that the pair satisfies the public syntactic and algebraic validity conditions required by the scheme.*
- ***PV3*** *Partial decryption validity: A partial decryption share* D_i *for ciphertext* C *can be checked by any party using* $\texttt{VerifyPD}((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)$*.*

Definition 5 (Indistinguishability under chosen-ciphertext attack (IND-CCA2 or CCA2)). *A non-interactive* (t, n)*-threshold encryption scheme is secure against chosen-ciphertext attacks with static corruptions if every PPT adversary has negligible advantage in the following game:*

1. *The challenger runs* $\texttt{Setup}(\lambda, t, n)$ *to obtain public parameters* pp *and key pairs* $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[n]}$*. It gives* pp *and* $\{\mathsf{pk}_i\}_{i\in[n]}$ *to* $\mathcal{A}$*.*
2. *The adversary chooses a static corruption set* $W \subseteq [n]$ *with* $|W| \leq t-1$ *and obtains* $\{\mathsf{sk}_i\}_{i\in W}$*.*
3. *The adversary* $\mathcal{A}$ *may issue partial-decryption queries on any ciphertext-proof pair* (C, π)*. The challenger returns* $\{(D_i, \pi_i) \leftarrow \texttt{PartialDec}((C, \pi), \mathsf{sk}_i)\}_{i\in[1,n]}$*; by definition, each call returns* $\perp$ *if* $\texttt{VerifyCT}(C, \pi) = \mathsf{false}$*.*
4. $\mathcal{A}$ *outputs two challenge secrets* $s_0, s_1 \in \mathcal{S}$*. The challenger samples* $b \xleftarrow{\$} \{0,1\}$*, computes* $(C^*, \pi^*) \leftarrow \texttt{Encrypt}(s_b, \mathsf{pp})$*, and sends* (C^*, π^*) *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *continues issuing partial-decryption queries as in Step 3, except that it may not query any pair* (C^*, π) *with the challenge ciphertext component* C^**.*
6. $\mathcal{A}$ *outputs a bit* b' *and wins if* $b' = b$*. Its advantage is* $\mathbf{Adv}^{\mathsf{cca2}}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$*.*

3.2 The Framework

Let $\Pi_{\mathsf{TE}} = (\texttt{Setup}, \texttt{VerifySetup}, \texttt{Encrypt}, \texttt{VerifyCT}, \texttt{PartialDec}, \texttt{VerifyPD}, \texttt{Decrypt})$ be a publicly verifiable threshold-encryption scheme satisfying the CCATE correctness, robustness, public-verifiability, and CCA2-security definitions above. We construct a PVSS scheme Π_{PVSS} by using the threshold-encryption ciphertext as the public distribution object and the threshold partial decryptions as reconstruction shares. Figure 2 gives the resulting wrapper construction.

Theorem 1 (Security of the generic transformation). *If* Π_{TE} *satisfies CCATE correctness, robustness, public verifiability, and CCA2 security, then the scheme* Π_{PVSS} *in Figure 2 satisfies PVSS correctness, public verifiability, and IND1-Secrecy. Moreover, if* `Encrypt` *and* `VerifyCT` *have size and running time independent of* n*, then* PVSS.Share *and* PVSS.Verify *also have size and running time independent of* n*.*

Functionality The Generic Constant-Online PVSS Framework

$\underline{(\mathsf{pp}, \{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[n]}) \leftarrow \mathsf{PVSS.Setup}(\lambda, t, n)}$

return $\mathtt{Setup}(\lambda, t, n)$

$\underline{bool \leftarrow \mathsf{PVSS.SetupVrf}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})}$

return $\mathtt{VerifySetup}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$

$\underline{(C, \pi_s) \leftarrow \mathsf{PVSS.Share}(s, \mathsf{pp})}$

return $\mathtt{Encrypt}(s, \mathsf{pp})$

$\underline{bool \leftarrow \mathsf{PVSS.Verify}(C, \pi_s)}$

return $\mathtt{VerifyCT}(C, \pi_s)$

$\underline{(D_i, \pi_{\mathsf{sk}_i}) \leftarrow \mathsf{PVSS.PreRecon}((C, \pi_s), \mathsf{sk}_i)}$

return $\mathtt{PartialDec}((C, \pi_s), \mathsf{sk}_i)$

$\underline{bool \leftarrow \mathsf{PVSS.DecShareVrf}((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)}$

return $\mathtt{VerifyPD}((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)$

$\underline{S \leftarrow \mathsf{PVSS.Recon}(C, \{(i, D_i)\}_{i\in Q})}$

return $\mathtt{Decrypt}(C, \{(i, D_i)\}_{i\in Q})$

Fig. 2. Generic constant-online PVSS framework based on CCATE

Proof. The PVSS algorithms are direct wrappers around the corresponding CCATE algorithms, so each security property follows by substitution.

- ***Correctness*** Let $(C, \pi_s) \leftarrow \mathsf{PVSS.Share}(s, \mathsf{pp})$ and let $\{(i, D_i)\}_{i\in Q}$ be honestly generated reconstruction shares with $|Q| \geq t$. By CCATE correctness, $\mathsf{PVSS.Verify}(C, \pi_s) = \mathsf{true}$, all honest partial-decryption proofs are accepted by $\mathsf{PVSS.DecShareVrf}$, and $\mathsf{PVSS.Recon}(C, \{(i, D_i)\}_{i\in Q}) = \mathtt{Decrypt}(C, \{(i, D_i)\}_{i\in Q})$ outputs S such that $(s, S) \in \mathcal{PK}$. CCATE robustness ensures that any accepted qualified set yields the same reconstruction value except with negligible probability.
- ***Public Verifiability***
  - ***Verifiability of Setup*** This is exactly `VerifySetup`, and follows from Property **PV1** (i.e., CCATE setup validity).
  - ***Verifiability of Distribution*** This is exactly `VerifyCT`, and follows from Property **PV2** (i.e., CCATE ciphertext validity).
  - ***Verifiability of Decryption*** This is exactly `VerifyPD`, and follows from Property **PV3** (i.e., CCATE partial-decryption validity).
- ***IND1-Secrecy*** Given an adversary $\mathcal{A}_{\mathsf{PVSS}}$ that distinguishes $(C^*, \pi^*) \leftarrow \mathsf{PVSS.Share}(x_b, \mathsf{pp})$ for chosen x_0, x_1, construct a CCATE adversary $\mathcal{A}$ that forwards the same public parameters, corruption answers, and challenge messages to its CCATE challenger. Corrupted shareholders' reconstruction shares are computable from the corrupted secret keys exactly as in the real PVSS game. If the PVSS game gives access to reconstruction-share

queries for non-challenge sharings, $\mathcal{A}$ answers them using its CCATE partial-decryption oracle; challenge-ciphertext queries remain disallowed, as in the CCATE CCA2 game. The CCATE challenge $(C^*, \pi^*) \leftarrow \texttt{Encrypt}(x_b, \mathsf{pp})$ is distributed exactly as the PVSS sharing output. Therefore $\mathcal{A}$ outputs $\mathcal{A}_{\mathsf{PVSS}}$'s guess, and $\mathbf{Adv}^{\mathsf{ind1}}_{\mathsf{PVSS},\mathcal{A}_{\mathsf{PVSS}}} = \mathbf{Adv}^{\mathsf{cca2}}_{\Pi,\mathcal{A}}$. Thus PVSS IND1-Secrecy is reduced to the corresponding CCATE CCA2 game with the same corruption set and no stronger challenge access.

The complexity claim follows because PVSS.Share invokes only `Encrypt`, and PVSS.Verify invokes only `VerifyCT`, with no additional loop over shareholders.

4 Instantiation I: CCATE from Threshold ElGamal

Overview. This section presents a baseline instantiation of our generic framework using standard Threshold ElGamal encryption. The construction serves two purposes: (1) demonstrating that the framework applies to minimal, well-established assumptions (DDH, without pairings), and (2) providing a practical reference point for comparison. For setup, we assume publicly auditable threshold keys; Appendix C gives one committee-based realization for completeness.

4.1 Committee-Based Setup Assumption

DKG [35,36,37] is a standard way to obtain threshold keys, but this work does not claim a new DKG protocol. The pairing-free instantiation only needs the following setup interface: before any PVSS instance is shared, the parties obtain threshold ElGamal key shares and public verification material that anyone can audit. This setup can be implemented by existing DKG/VSS techniques, by a committee-assisted key-generation ceremony, or by any other mechanism satisfying the assumption below.

Definition 6 (Committee-based setup assumption). *An N-member setup committee assists in generating threshold keys for n parties, where the N committee members and the n parties are not necessarily overlapping. The setup is run before the online PVSS phase and is treated as an assumption of Instantiation I.*

1. *At the end of setup, each party obtains a secret key share sk_i and publishes a corresponding public key share $\mathsf{pk}_i = g^{\mathsf{sk}_i}$. A master public key $\mathsf{pk} = g^{\mathsf{sk}}$ is publicly defined.*
2. *The key shares are consistent: for every qualified set $Q \subseteq [1, n]$ with $|Q| \geq t$, the interpolation relation $\sum_{i \in Q} L_i(0)\mathsf{sk}_i = \mathsf{sk}$ holds.*
3. *The master secret sk remains hidden from any adversary corrupting fewer than t parties, assuming the setup ceremony contains sufficient honest entropy, e.g., at least one honest committee contribution in a concrete realization.*
4. *The public transcript contains enough commitments, signatures, and complaint information for any verifier to audit that the accepted public key shares and pk satisfy the advertised setup rules.*

The online Threshold ElGamal construction below uses only Definition 6. One possible Feldman-VSS-based realization, the corresponding algebraic checks, and an epoch-based reconfiguration discussion are deferred to Appendix C; they are included only to make the setup assumption concrete.

4.2 CCATE Construction

Given setup outputs $(\mathsf{p}, \{\mathsf{sk}_i, \mathsf{pk}_i\})$ satisfying Definition 6, Figure 3 presents the core CCATE algorithms. The construction is pairing-free and builds on standard Threshold ElGamal. Our technical contribution here is the design of efficient NIZK proofs (π_s for encryption validity, π_{sk_i} for partial decryption) that enable public verifiability while maintaining $O(1)$ ciphertext size. The ciphertext-validity proof π_s proves membership in the following relation, where the statement is (C_0, C_1, pk) and the witness is the plaintext/randomness pair (s, w):

$$R_{\mathsf{EG}} = \left\{ \big((C_0, C_1, \mathsf{pk}), (s, w)\big) : C_0 = g^s \mathsf{pk}^w \wedge C_1 = g^w \right\}.$$

4.3 Security Analysis

We prove that the construction depicted in Figure 3 achieves **correctness**, **robustness**, **public verifiability**, and **CCA2 security**. Further, by the framework established in Section 3, the construction implies a secure PVSS construction.

Lemma 1 (NIZK Proof Completeness I). *If an encryptor is honest, then* $\Pr[(C, \pi_s) \leftarrow \texttt{Encrypt}(s, \mathsf{pp}) : \texttt{VerifyCT}\ (C, \pi_s) = 1] = 1$.

Proof : The Fiat–Shamir challenge check holds by construction because $c = H(\ell_{\mathsf{inst}_1}, C, C')$. We check the two remaining equations in the `VerifyCT` algorithm as follows:

$$C[0]^c \cdot g^{\hat{s}} \cdot \mathsf{pk}^{\hat{w}} = (g^s \mathsf{pk}^w)^c \cdot g^{s'-cs} \cdot \mathsf{pk}^{w'-wc} = g^{s'} \mathsf{pk}^{w'} = C'[0],$$

$$C[1]^c \cdot g^{\hat{w}} = (g^w)^c \cdot g^{w'-wc} = g^{w'} = C'[1].$$

Further, completeness indicates that the ciphertext is well formed. □

Lemma 2 (NIZK Proof Soundness I). *Given two accepting transcripts* $(C, \pi_{s1} = (C', c_1, (\hat{w}_1, \hat{s}_1)))$ *and* $(C, \pi_{s2} = (C', c_2, (\hat{w}_2, \hat{s}_2)))$ *with* $c_1 \neq c_2$*, an efficient extractor can output the witness* s *and* w.

Proof : Since the two transcripts are accepting, we have

$$\hat{w}_i = w' - c_i w, \quad \hat{s}_i = s' - c_i s,$$

where $i = 1, 2$. Then, by subtracting the equations from each other, we obtain

$$\hat{w}_1 - \hat{w}_2 = (c_1 - c_2)w, \quad \hat{s}_1 - \hat{s}_2 = (c_1 - c_2)s.$$

Functionality CCA2-Secure Threshold ElGamal

$(\mathsf{pp}, \{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[n]}) \leftarrow \mathtt{Setup}(\lambda, t, n)$

return $\mathsf{CommitteeSetup}(\lambda, t, n)$

$bool \leftarrow \mathtt{VerifySetup}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in[n]})$

verify the public setup transcript, accepted commitments, complaints, pk, and $\{\mathsf{pk}_i\}_{i\in[n]}$ as required by Definition 6; return true iff all checks pass

$(C, \pi_s) \leftarrow \mathtt{Encrypt}(s, \mathsf{pp})$

$w, w' \xleftarrow{\$} \mathbb{Z}_p,\ s' \xleftarrow{\$} \mathbb{Z}_p$

$C = (g^s \cdot \mathsf{pk}^w, g^w), C' = (g^{s'} \cdot \mathsf{pk}^{w'}, g^{w'})$

$c = H(\ell_{\mathsf{inst}_1}, C, C'), \hat{w} = w' - wc, \hat{s} = s' - cs$

$\pi_s = (C', c, (\hat{w}, \hat{s}))$

$bool \leftarrow \mathtt{VerifyCT}(C, \pi_s)$

parse π_s as $(C', c, (\hat{w}, \hat{s}))$

parse C as $(C[0], C[1])$ and C' as $(C'[0], C'[1])$

$c \stackrel{?}{=} H(\ell_{\mathsf{inst}_1}, C, C')$

$C'[0] \stackrel{?}{=} C[0]^c \cdot g^{\hat{s}} \cdot \mathsf{pk}^{\hat{w}}$

$C'[1] \stackrel{?}{=} C[1]^c \cdot g^{\hat{w}}$

$(D_i, \pi_{\mathsf{sk}_i}) \leftarrow \mathtt{PartialDec}((C, \pi_s), \mathsf{sk}_i)$

if $\mathtt{VerifyCT}(C, \pi_s) = \mathsf{false}$, return $\perp$

$D_i = C[1]^{\mathsf{sk}_i}$

$\pi_{\mathsf{sk}_i} = DLEQ(g, C[1], \mathsf{pk}_i, D_i, \mathsf{sk}_i)$

$bool \leftarrow \mathtt{VerifyPD}((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)$

$\mathsf{true} \stackrel{?}{=} DLEQVrf(g, C[1], \mathsf{pk}_i, D_i, \pi_{\mathsf{sk}_i})$

$S \leftarrow \mathtt{Decrypt}(C, \{(i, D_i)\}_{i\in Q})$

$S \leftarrow C[0] \cdot \left(\prod_{i\in Q} D_i^{L_i(0)}\right)^{-1}$, where $L_i(x)$ is the Lagrange basis polynomial.

Fig. 3. CCATE based on committee-based setup and Threshold ElGamal

Since $c_1 \neq c_2$, we can invert $(c_1 - c_2)$ in the underlying field, yielding:

$$w = (\hat{w}_1 - \hat{w}_2)(c_1 - c_2)^{-1}, \quad s = (\hat{s}_1 - \hat{s}_2)(c_1 - c_2)^{-1}.$$

This extraction is computable and efficient. The resulting (w, s) satisfies the witness relation by construction, completing the proof. □

Theorem 2 (Correctness). *The CCATE described in Figure 3 satisfies correctness, defined by Definition 2.*

Proof : For honest correctness, Lemma 1 gives $\mathtt{VerifyCT}(C, \pi_s) = \mathsf{true}$ for $(C, \pi_s) \leftarrow \mathtt{Encrypt}(s, \mathsf{pp})$. For every $i \in Q$, the honest partial-decryption algorithm outputs $D_i = C[1]^{\mathsf{sk}_i}$ together with a DLEQ proof accepted by $\mathtt{VerifyPD}$

by completeness of the DLEQ protocol. It remains to check that reconstruction outputs S such that $(s, S) \in \mathcal{PK}$:

$$\begin{aligned} C[0] \cdot (\prod_{i \in Q} D_i^{L_i(0)})^{-1} &= g^s \mathsf{pk}^w \cdot (\prod_{i \in Q} (g^w)^{\mathsf{sk}_i L_i(0)})^{-1} \\ &= g^s \mathsf{pk}^w \cdot ((g^w)^{\sum_{i \in Q} \mathsf{sk}_i L_i(0)})^{-1} \\ &= g^s \mathsf{pk}^w \cdot ((g^w)^{\mathsf{sk}})^{-1} = g^s = S. \end{aligned}$$

Thus $\Pr[(s, S) \in \mathcal{PK}] = 1$. □

Theorem 3 (Robustness). *The CCATE described in Figure 3 satisfies robustness, defined by Definition 3.*

Proof : Let $C = (C[0], C[1])$ be any ciphertext accepted by `VerifyCT`, and let $Q, Q' \subseteq [n]$ be two qualified sets of decryption shares accepted by `VerifyPD`. Let $S_Q = \texttt{Decrypt}(C, \{(i, D_i)\}_{i \in Q})$ and $S_{Q'} = \texttt{Decrypt}(C, \{(i, D'_i)\}_{i \in Q'})$. For each accepted tuple $(D_i, \pi_{\mathsf{sk}_i})$, the DLEQ verification statement is $\log_g(\mathsf{pk}_i) = \log_{C[1]}(D_i)$. Let $\epsilon_{\mathsf{dleq\text{-}snd}}(\lambda)$ denote the soundness failure probability that any invalid DLEQ proof is accepted over all polynomially many partial-decryption verification attempts made by a PPT adversary. Except with this probability, every accepted share satisfies $D_i = C[1]^{\mathsf{sk}_i}$ and $D'_i = C[1]^{\mathsf{sk}_i}$ for its claimed index i. Conditioned on no DLEQ-soundness failure, reconstruction from any qualified set Q yields $S_Q = C[0] \cdot \left(\prod_{i \in Q} D_i^{L_i(0)}\right)^{-1} = C[0] \cdot \left(C[1]^{\sum_{i \in Q} \mathsf{sk}_i L_i(0)}\right)^{-1} = C[0] \cdot \left(C[1]^{\mathsf{sk}}\right)^{-1}$. The same expression is obtained for $S_{Q'}$ with qualified set Q', because the setup assumption gives the same interpolated master secret sk for every qualified set and Lagrange interpolation is unique. Hence two accepted qualified reconstructions can differ only if the DLEQ soundness event fails. Therefore

$$\Pr[S_Q \neq S_{Q'}] \leq \epsilon_{\mathsf{dleq\text{-}snd}}(\lambda),$$

which is negligible. □

Theorem 4 (Public Verifiability). *The CCATE described in Figure 3 satisfies public verifiability, defined by Definition 4.*

Proof : We verify the three public-verifiability requirements in Definition 4.

- Property **PV1** (**Setup validity**) is implemented by `VerifySetup`. The verifier uses only the public setup transcript, the public parameters pp, the threshold public key pk, and the shareholder public keys $\{\mathsf{pk}_i\}_{i \in [n]}$. It accepts only if these values pass the public consistency checks required by the committee-based setup assumption in Definition 6. No secret key or dealer-side randomness is used in this verification.
- Property **PV2** (**Ciphertext validity**) is implemented by $\texttt{VerifyCT}(C, \pi_s)$. On input $C = (C[0], C[1])$ and $\pi_s = (C', c, (\hat{w}, \hat{s}))$, the verifier first checks $c = H(\ell_{\mathsf{inst}_1}, C, C')$, and then checks the two public equations $C'[0] =$

$C[0]^c \cdot g^{\hat{s}} \cdot \mathsf{pk}^{\hat{w}}, C'[1] = C[1]^c \cdot g^{\hat{w}}$. By Lemma 1, honestly generated ciphertext-proof pairs are accepted. By Lemma 2, except with negligible probability, an accepted pair has a valid witness (s, w) such that $C = (g^s \mathsf{pk}^w, g^w)$. Thus any party can publicly check the algebraic well-formedness required for decryption.

– Property **PV3** (**Partial-decryption validity**) is implemented by `VerifyPD` $((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)$. The verifier uses only C, the claimed share D_i, the public key pk_i, and the DLEQ proof π_{sk_i}. It accepts only if $DLEQVrf(g, C[1], \mathsf{pk}_i, D_i, \pi_{\mathsf{sk}_i}) = \mathsf{true}$, which proves equality of discrete logarithms $\log_g(\mathsf{pk}_i) = \log_{C[1]}(D_i)$. Hence, except with the DLEQ soundness failure probability, an accepted partial decryption share is bound to the public key pk_i and must equal $D_i = C[1]^{\mathsf{sk}_i}$. This check is public and does not reveal or require sk_i.

Therefore all public outputs produced by setup, encryption, and partial decryption are publicly verifiable. □

Lemma 3 (NIZK Proof Zero-Knowledge I). *Given input (C, c), a simulator can output a transcript $(\tilde{C}', c, (\tilde{w}, \tilde{s}))$ identical to real-world output between an honest dealer and an honest shareholder.*

Proof : We construct a simulator that, given $C = (C[0], C[1]) = (g^s \cdot \mathsf{pk}^w, g^w)$ and a challenge $c \in \mathbb{Z}_p$, outputs a valid transcript $(\tilde{C}', c, (\tilde{w}, \tilde{s}))$ without knowing s or w.

1. Sample random values: $\tilde{w} \xleftarrow{\$} \mathbb{Z}_p, \quad \tilde{s} \xleftarrow{\$} \mathbb{Z}_p$.
2. Compute the commitment $\tilde{C}'$ as: $\tilde{C}' = (C'[0], C'[1]) = \left(g^{\tilde{s}} \cdot \mathsf{pk}^{\tilde{w}} \cdot C[0]^c,\ g^{\tilde{w}} \cdot C[1]^c\right)$. This ensures that the verification equation will pass with the response $(\tilde{w}, \tilde{s})$ and challenge c.
3. Program the random oracle so that $H(\ell_{\mathsf{inst}_1}, C, \tilde{C}') = c$. In the random oracle model this programming is valid unless the labelled point $(\ell_{\mathsf{inst}_1}, C, \tilde{C}')$ was queried before, which occurs with negligible probability for fresh $\tilde{C}'$.
4. Output the simulated transcript: $(\tilde{C}', c, (\tilde{w}, \tilde{s}))$.

In the real protocol, w, w', s' are sampled uniformly at random in $\mathbb{Z}_p$. $c = H(\ell_{\mathsf{inst}_1}, C, C')$ is determined by the random oracle. Consequently, $\hat{w} = w' - wc$, $\hat{s} = s' - sc$ are uniform (due to one-time pad-like masking), and C' is a uniformly random group element in the image of the commitment. In the simulation, the simulator samples $\tilde{w}$ and $\tilde{s}$ uniformly from $\mathbb{Z}_p$, resulting in distributions identical to $(\hat{w}, \hat{s})$. $\tilde{C}'$ is computed as a uniformly random-looking group element (since $g^{\tilde{s}}, g^{\tilde{w}}$ are random). Thus, the distribution of $(\tilde{C}', c, (\tilde{w}, \tilde{s}))$ is identical to that of a real transcript, because all components are sampled from the same distributions and satisfy the same algebraic relations. □

Theorem 5 (CCA2 Security). *The CCATE described in Figure 3 is **CCA2** secure against the game in Definition 5 under the Decisional Diffie-Hellman (DDH) assumption in the random oracle model, assuming the Fiat-Shamir NIZK used in* `VerifyCT` *has labelled simulation-sound online extractability and the DLEQ proof used in* `VerifyPD` *is sound and zero-knowledge.*

Proof : Let $\mathcal{A}$ be any PPT adversary in the CCA2 experiment of Definition 5. We prove the theorem through a sequence of games. Let W_j denote the event that $\mathcal{A}$ outputs the challenge bit in Game G_j.

Game G_0**.** This is the real CCA2 experiment. The challenger runs `Setup`, gives `pp` and all public keys to $\mathcal{A}$, answers the static corruption query set W with $|W| \leq t-1$, answers partial-decryption queries according to `PartialDec`, generates the challenge $(C^*, \pi^*) \leftarrow \texttt{Encrypt}(s_b, \mathsf{pp})$, and rejects any later query whose ciphertext component is C^*. Thus, $\mathbf{Adv}^{\mathsf{cca2}}_{\Pi,\mathcal{A}} = \left|\Pr[W_0] - \frac{1}{2}\right|$.

Game G_1**.** This game is identical to G_0, except that the challenger first evaluates $\texttt{VerifyCT}(C, \pi)$ on every partial-decryption query (C, π) and returns $\perp$ if verification fails. This is only an explicit form of the rejection rule already used by the honest `PartialDec` algorithm. Hence $\Pr[W_1] = \Pr[W_0]$.

Game G_2**.** This game changes how valid non-challenge partial-decryption queries are answered. On a query (C, π) with $C \neq C^*$ and $\texttt{VerifyCT}(C, \pi) = \mathsf{true}$, the challenger invokes the labelled simulation-sound online extractor for the ciphertext NIZK under label ℓ_{inst_1}. Except with probability ϵ_{ext}, extraction returns a witness (s, w) satisfying $C = (C[0], C[1]) = (g^s \mathsf{pk}^w, g^w)$. The challenger then computes every decryption share as $D_i = (\mathsf{pk}_i)^w = (g^{\mathsf{sk}_i})^w = C[1]^{\mathsf{sk}_i}$ using only public keys, and replaces each real DLEQ proof by a simulated DLEQ proof. The values D_i are exactly distributed as in G_1; the only changed objects are the DLEQ proofs. By zero-knowledge of the DLEQ proof system, $|\Pr[W_2] - \Pr[W_1]| \leq \epsilon_{\mathsf{ext}} + \epsilon_{\mathsf{dleq\text{-}zk}}$.

Game G_3**.** This game is identical to G_2, except that the challenge proof π^* is simulated using Lemma 3. More precisely, for the challenge ciphertext C^* the challenger samples the responses first, programs the random oracle at the fresh labelled point $(\ell_{\mathsf{inst}_1}, C^*, C'^*)$, and obtains an accepting Fiat-Shamir transcript. The simulation fails only if this programmed random-oracle point has already been queried. Let this probability be ϵ_{ro}. By zero-knowledge of the ciphertext proof, $|\Pr[W_3] - \Pr[W_2]| \leq \epsilon_{\mathsf{zk}} + \epsilon_{\mathsf{ro}}$. Because the CCA2 experiment still allows post-challenge partial-decryption queries for ciphertexts $C \neq C^*$, these non-challenge queries are answered exactly as in G_2, by extracting their witnesses under the same label ℓ_{inst_1}. Labelled simulation-sound online extractability is needed because the adversary has seen the simulated challenge proof and may later submit other accepting non-challenge proofs.

Game G_4**.** This game replaces the ElGamal mask (i.e., pk^w, $w \xleftarrow{\$} \mathbb{Z}_p$) in the challenge ciphertext by a random group element. Namely, the challenger samples $T \xleftarrow{\$} \mathbb{G}$, sets $C^* = (g^{s_b} T, g^w)$, and simulates π^* exactly as in G_3. Since T is uniform in $\mathbb{G}$, the component $g^{s_b} T$ is uniform and independent of b. Therefore the whole challenge view is independent of b, while all partial-decryption oracle answers are produced without using b. Hence $\Pr[W_4] = \frac{1}{2}$. It remains to relate G_3 and G_4. We construct a DDH distinguisher $\mathcal{B}$ receiving a tuple (g, pk, U, T), where either $U = g^w$ and $T = \mathsf{pk}^w$ or $U = g^w$ and T is uniform in $\mathbb{G}$. $\mathcal{B}$ runs $\mathcal{A}$, uses the given (g, pk) as the master public key, answers corruption queries using the honestly generated Shamir shares, answers all valid non-challenge decryption queries by extraction as in G_2, and sets the challenge

ciphertext to $C^* = (g^{s_b}T, U)$. It then simulates the challenge proof as in G_3. If $T = \mathsf{pk}^w$, the view is exactly G_3; if T is uniform, the view is exactly G_4. Thus $|\Pr[W_3] - \Pr[W_4]| \leq \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G},\mathcal{B}}$.

Combining the games gives

$$\mathbf{Adv}^{\mathsf{cca2}}_{\Pi,\mathcal{A}} \leq \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G},\mathcal{B}} + \epsilon_{\mathsf{ext}} + \epsilon_{\mathsf{dleq\text{-}zk}} + \epsilon_{\mathsf{zk}} + \epsilon_{\mathsf{ro}},$$

which is negligible under the stated assumptions. □

Discussion on Adaptive security. The security proof presented above is established in the static corruption model. We note that our scheme can be naturally extended to achieve adaptive security by applying the recent techniques of Das, Ren, and Yang [38], who constructed adaptively secure threshold ElGamal encryption schemes under DDH in the random oracle model. Since our construction shares the same algebraic structure, their transformation applies in a straightforward manner with comparable efficiency.

5 Instantiation II: CCATE with Silent Setup

We present a CCATE instantiation based on the *silent setup* paradigm recently introduced by Garg et al. [9]. This construction achieves the same constant-online behavior as Section 4, but with a different setup trade-off: each party generates keys locally without pairwise communication, while setup verification and aggregation require heavier public-key material and computation.

In traditional threshold encryption, parties must engage in a costly DKG protocol to jointly establish a shared public key and distribute secret shares. STE overturns this paradigm by enabling each party to locally generate its own key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$, after which a global encryption key is deterministically aggregated from public components alone. This "silent setup" not only removes all online coordination during key generation but also inherently supports dynamic committee membership. The CPA-security of STE relies on a novel application of signature-based witness encryption: the encryptor samples a random tag to construct a ciphertext such that decryption is possible if and only if a valid threshold signature for the tag can be aggregated from at least t parties. In our work, we elevate STE to CCA2 securitya necessary condition for integration into our CCATE frameworkby augmenting ciphertexts with NIZK proofs. This fortified variant preserves STE's silent-setup advantages while meeting the stringent verifiability and security requirements of our generic PVSS transformation.

5.1 Silent Setup

Power-Of-Tau with Public Verifiability The silent-setup instantiation assumes a publicly verifiable Power-of-Tau CRS $\mathsf{CRS} = \{g_1^{\tau^i}, g_2^{\tau^i}\}_{i\in[0,n+1]}$, where the trapdoor τ is unknown as long as at least one ceremony participant honestly erases its contribution. We use this ceremony as standard setup machinery rather than as a contribution of this work. For completeness, Appendix D recalls the CRS update and verification procedures.

Functionality Silent Setup for CCATE

$(\mathsf{pp}, \{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[1,n-1]}) \leftarrow \mathtt{Setup}(\lambda, t, n)$

- Sample $g_1 \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2$, compute $g_t = e(g_1, g_2)$ and sample $g_t' \xleftarrow{\$} \mathbb{G}_T$.
- Compute an n-th root of unity ω such that $\omega^n = 1$. Obtain valid $\mathsf{CRS} = \left(\{g_1^{\tau^i}, g_2^{\tau^i}\}_{i\in[0,n+1]}\right)$ via *PowerOfTau* and *VerifyCRS* (cf. Appendix D).
- Derive $Z_0 = g_2^{\tau^{-1}}$ from the verified CRS.
- Each real party P_i for $i \in [1, n-1]$ randomly samples $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$, computes and publishes $\mathsf{pk}_{i,1} = g_1^{\mathsf{sk}_i}$, $\mathsf{pk}_{i,1}^{(2)} = g_2^{\mathsf{sk}_i}$ and $\mathsf{pk}_{i,2} = (h_{i,1}, h_{i,2}, h_{i,3}, h_{i,4}, \{h_{i,5,j}\}_{j\in[0,n-1],j\neq i})$, where $h_{i,1} = g_1^{\mathsf{sk}_i L_i(\tau)}, h_{i,2} = g_1^{\mathsf{sk}_i (L_i(\tau)-L_i(0))}, h_{i,3} = g_1^{\mathsf{sk}_i \frac{L_i^2(\tau)-L_i(\tau)}{Z(\tau)}}, h_{i,4} = g_1^{\mathsf{sk}_i \frac{(L_i(\tau)-L_i(0))}{\tau}}, \{h_{i,5,j} = g_1^{\mathsf{sk}_i \frac{L_i(\tau)L_j(\tau)}{Z(\tau)}}\}$.
- Index 0 is a deterministic public dummy slot with $\mathsf{sk}_0 = 1_{\mathbb{Z}_p}$;
- Outputs $\mathsf{pp} = (\mathsf{CRS}, \{\omega^i\}, g_1, g_2, g_t, g_t', Z_0)$ and $\{\mathsf{sk}_i, \mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,1}^{(2)}, \mathsf{pk}_{i,2})\}$.

bool $\leftarrow \mathtt{VerifySetup}(\mathsf{pp}, \{\mathsf{pk}_i\})$

- For each real party $i \in [1, n-1]$, an honest-but-curious aggregator verifies each $\mathsf{pk}_{i,2}$ and the cross-group BLS key: $e(\mathsf{pk}_{i,1}, g_2) \stackrel{?}{=} e(g_1, \mathsf{pk}_{i,1}^{(2)})$, $e(h_{i,1}, g_2) \stackrel{?}{=} e(\mathsf{pk}_{i,1}, g_2^{L_i(\tau)}), e(h_{i,2}, g_2) \stackrel{?}{=} e(\mathsf{pk}_{i,1}, g_2^{(L_i(\tau)-L_i(0))}), e(h_{i,3}, g_2) \stackrel{?}{=} e(\mathsf{pk}_{i,1}, g_2^{\frac{L_i^2(\tau)-L_i(\tau)}{Z(\tau)}}), e(h_{i,4}, g_2) \stackrel{?}{=} e(\mathsf{pk}_{i,1}, g_2^{\frac{(L_i(\tau)-L_i(0))}{\tau}}), e(h_{i,5,j}, g_2) \stackrel{?}{=} e(\mathsf{pk}_{i,1}, g_2^{\frac{L_i(\tau)L_j(\tau)}{Z(\tau)}})$.
- Let $I \subseteq [1, n-1]$ be the index set of real parties with valid public keys; the dummy slot 0 is included deterministically.
- The aggregator computes $X = \prod_{i\in I} h_{i,1}, Z = g_2^{Z(\tau)}, A_i = \prod_{j=0,j\neq i}^{n-1} h_{j,5,i} (i \in I \cup \{0\})$ and sets $\mathsf{ek} = (X, Z), \mathsf{ak} = (I, \{\mathsf{pk}_i\}_{i\in I\cup\{0\}}, \{h_{i,2}\}_{i\in I\cup\{0\}}), \{h_{i,3}\}_{i\in I\cup\{0\}}), \{h_{i,4}\}_{i\in I\cup\{0\}}), \{A_i\}_{i\in I\cup\{0\}})$.
- Finally, the algorithm updates $\mathsf{pp} \leftarrow \mathsf{pp} \bigcup \{(\mathsf{ek}, \mathsf{ak})\}$.

Fig. 4. Silent setup with $n-1$ real parties and one dummy slot

Silent Setup for DKG pairs To make the writing consistency, we describe the setup in DKG-style which primarily outputs threshold-enabled key pairs. Figure 4 shows the publicly verifiable DKG construction, which plays the role of silent setup. Note that this DKG does not require Shamir SS or VSS schemes [39,40,41]. The setup phase `Setup` employs the Power of Tau protocol to produce a common reference string CRS, relying on the assurance that at least one participant in the ceremony is honest. Additionally, each party P_i generates a key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ locally, where the public key pk_i enables `VerifySetup` to verify the integrity and correctness of P_i's actions. $L(x)$ is the Lagrange basis polynomial and $Z(x)$ is the vanishing polynomial defined in Definition 8. Note that the computation of $\mathsf{pk}_{i,2}$ relies solely on the CRS and does not require knowledge of the trapdoor τ. For example, the element $h_{i,1} = g_1^{\mathsf{sk}_i L_i(\tau)}$ is computed by first generating a

KZG commitment to the polynomial $L_i(x)$ using the CRS, followed by a scalar exponentiation with the party's secret key sk_i. The setup also asks each party to publish $\mathsf{pk}_{i,1}^{(2)} = g_2^{\mathsf{sk}_i}$. This element is needed only for the partial-decryption proof in Figure 5, because the ciphertext tag $C[0] = g_2^{\gamma}$ and the decryption share $D_i = C[0]^{\mathsf{sk}_i}$ are both elements of $\mathbb{G}_2$. Publishing $\mathsf{pk}_{i,1}^{(2)}$ does not introduce an independent key: `VerifySetup` binds it to the ordinary public key $\mathsf{pk}_{i,1} = g_1^{\mathsf{sk}_i}$ by checking $e(\mathsf{pk}_{i,1}, g_2) = e(g_1, \mathsf{pk}_{i,1}^{(2)})$. Thus, the only additional requirement is the standard discrete-log hardness of $\mathbb{G}_2$ and rejection of keys that fail this cross-group consistency check.

Aggregator trust boundary. The aggregator in Figure 4 is an honest-but-curious public computation service: it receives only public keys and CRS elements, performs the deterministic pairing checks and products specified by `VerifySetup`, and publishes the accepted index set I, ek, ak, and the verification transcript. It is not trusted with any secret key and is not assumed to preserve privacy. Security relies on correctness of the published aggregation, not on hidden aggregator state: anyone can recompute the same checks and products from the public transcript and reject malformed outputs. A malicious aggregator can at most cause denial of service by omitting valid parties or refusing to publish; such liveness failures are outside the cryptographic security model.

Epoch-Based Reconfiguration Silent setup supports epoch-based membership changes by recomputing the aggregated keys for a new party set. In epoch e, the active set P_e is the set of parties whose public keys pass `VerifySetup` and are incorporated into ek_e and ak_e. To move to epoch $e+1$, new parties independently generate public keys, invalid or departing parties are excluded, and anyone can recompute and audit the new $(\mathsf{ek}_{e+1}, \mathsf{ak}_{e+1})$. As in Section C.4, security is per epoch: ciphertexts created under $(\mathsf{ek}_e, \mathsf{ak}_e)$ remain tied to epoch e, while ciphertexts in the new epoch use $(\mathsf{ek}_{e+1}, \mathsf{ak}_{e+1})$. We do not claim security for arbitrary adaptive membership changes inside a single epoch.

5.2 CCATE Instantiation II

Leveraging the silent setup outputs (described in Section 5.1), Figure 5 illustrates the following algorithms of CCATE scheme. Our primary contribution lies in the construction of NIZK proofs integrated into the STE scheme [9] (see Appendix E for details). The ciphertext-validity proof π_s proves membership in the following relation, where $\mathbf{A}$ and $\mathbf{b}$ are derived from $(\mathsf{pp}, \mathsf{ek}, \mathsf{ak})$ as in Figure 5; the statement is $(C, \mathsf{pp}, \mathsf{ek}, \mathsf{ak})$ and the witness is $(\gamma, \mathbf{v}, s)$:

$$R_{\mathsf{STE}} = \left\{ \left((C, \mathsf{pp}, \mathsf{ek}, \mathsf{ak}), (\gamma, \mathbf{v}, s) \right) : C = (g_2^{\gamma}, \mathbf{A}^{\mathbf{v}}, \mathbf{b}^{\mathbf{v}} {g'_t}^{s}) \right\}.$$

These NIZK proofs π_s are derived using the Sigma protocol combined with the labelled Fiat–Shamir heuristic under ℓ_{inst_2}. Specifically, $\pi_s = (C', c, (\hat{\gamma}, \hat{\mathbf{v}}, \hat{s}))$, where C' is the commitment, c is the challenge, and $(\hat{\gamma}, \hat{\mathbf{v}}, \hat{s})$ are the response values. The DLEQ proof in `PartialDec` is instantiated over $\mathbb{G}_2$, since both the

Functionality CCATE from Silent Setup

$\underline{(C, \pi_s) \leftarrow \texttt{Encrypt}(s, \mathsf{pp})}$

$\gamma, \gamma' \xleftarrow{\$} \mathbb{Z}_p$, parse Z_0 from pp, parse $\mathsf{ek} = (X, Z)$ and set

$$\mathbf{A} = \begin{pmatrix} X & g_2 & Z & g_2^{\tau} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_2^{t} & g_2 & 0 & 0 & 0 \\ 0 & g_2^{\gamma} & 0 & 0 & 0 & g_1 & 0 & 0 \\ g_1^{\tau\, t} & 0 & 0 & 0 & 0 & 0 & g_2 & 0 \\ g_1 & 0 & 0 & 0 & 0 & 0 & 0 & Z_0 \end{pmatrix}$$

set $\mathbf{b} = (1_{\mathbb{G}_T}, 1_{\mathbb{G}_T}, 1_{\mathbb{G}_T}, 1_{\mathbb{G}_T}, g_t)^\top$, $\mathbf{v} = (s_1, \cdots, s_5)^\top \xleftarrow{\$} \mathbb{Z}_p^5, \mathbf{v}' = (s_1', \cdots, s_5')^\top \xleftarrow{\$} \mathbb{Z}_p^5$, $s' \xleftarrow{\$} \mathbb{Z}_p$

$C = (g_2^{\gamma},\ \mathbf{A}^{\mathbf{v}},\ \mathbf{b}^{\mathbf{v}} \cdot g_t'^{s})$, where $\mathbf{A}^{\mathbf{v}}$ and $\mathbf{b}^{\mathbf{v}}$ denote scalar exponentiation in matrix.

$C' = (g_2^{\gamma'}, \mathbf{A}^{\mathbf{v}'}, \mathbf{b}^{\mathbf{v}'} \cdot g_t'^{s'})$

$c = H(\ell_{\mathsf{inst}_2}, C, C'), \hat{\gamma} = \gamma' - c\gamma, \hat{\mathbf{v}} = \mathbf{v}' - c\mathbf{v}, \hat{s} = s' - cs$

$\pi_s = (C', c, \hat{\gamma}, \hat{\mathbf{v}}, \hat{s})$

$\underline{bool \leftarrow \texttt{VerifyCT}(C, \pi_s)}$

Parse $\pi_s = (C', c, \hat{\gamma}, \hat{\mathbf{v}}, \hat{s})$

$c \stackrel{?}{=} H(\ell_{\mathsf{inst}_2}, C, C')$

$C'[0] \stackrel{?}{=} C[0]^c g_2^{\hat{\gamma}}$

$C'[1] \stackrel{?}{=} C[1]^c \circ \mathbf{A}^{\hat{\mathbf{v}}}$, where $\circ$ denotes hadamard product.

$C'[2] \stackrel{?}{=} C[2]^c \mathbf{b}^{\hat{\mathbf{v}}} g_t'^{\hat{s}}$

$\underline{(D_i, \pi_{\mathsf{sk}_i}) \leftarrow \texttt{PartialDec}((C, \pi_s), \mathsf{sk}_i) \text{ for } i \in I}$

if $\texttt{VerifyCT}(C, \pi_s) = \mathsf{false}$, return $\perp$

$D_i = C[0]^{\mathsf{sk}_i} = g_2^{\gamma \mathsf{sk}_i}$

$\pi_{\mathsf{sk}_i} = DLEQ(g_2, C[0], \mathsf{pk}_{i,1}^{(2)}, D_i, \mathsf{sk}_i)$

$\underline{bool \leftarrow \texttt{VerifyPD}((D_i, \pi_{\mathsf{sk}_i}), \mathsf{pk}_i, C)}$

$\mathsf{true} \stackrel{?}{=} DLEQVrf(g_2, C[0], \mathsf{pk}_{i,1}^{(2)}, D_i, \pi_{\mathsf{sk}_i})$

$\underline{S \leftarrow \texttt{Decrypt}(C, \{(i, D_i)\}_{i \in Q})}$

Given ak and enough valid shares from $Q \subseteq I$, this algorithm computes the vector w (Refer to Appendix E) such that $e(\mathbf{A}, \mathsf{w}) = \mathbf{b}$ and outputs $S = C[2] \cdot e(C[1], \mathsf{w})^{-1}$.

Fig. 5. Online CCATE algorithms using the silent-setup parameters from Figure 4

first ciphertext component and a partial decryption share lie in $\mathbb{G}_2$: $C[0] = g_2^{\gamma}$ and $D_i = C[0]^{\mathsf{sk}_i} = g_2^{\gamma \mathsf{sk}_i}$. The setup verification binds $\mathsf{pk}_{i,1} = g_1^{\mathsf{sk}_i}$ and $\mathsf{pk}_{i,1}^{(2)} = g_2^{\mathsf{sk}_i}$ by checking $e(\mathsf{pk}_{i,1}, g_2) = e(g_1, \mathsf{pk}_{i,1}^{(2)})$, so using $\mathsf{pk}_{i,1}^{(2)}$ in $DLEQ(g_2, C[0], \mathsf{pk}_{i,1}^{(2)}, D_i, \mathsf{sk}_i)$ keeps the proof in the standard single-group DLEQ form without introducing an independent key. As in Remark 1, the ciphertext proof π_s is instantiated with labelled simulation-sound online extractability, e.g., by a Fischlin-style ROM transformation, so that the CCA2 proof may extract witnesses from valid non-challenge ciphertext proofs after a simulated challenge proof has been issued.

5.3 Security Analysis

Lemma 4 (NIZK Proof Completeness II). *If an encryptor is honest, then* $\Pr[(C, \pi_s) \leftarrow \texttt{Encrypt}(s, \mathsf{pp}) : \texttt{VerifyCT}\ (C, \pi_s) = 1] = 1$.

Proof : The check of the challenge value c follows directly from the labelled Fiat–Shamir derivation $c = H(\ell_{\mathsf{inst}_2}, C, C')$. We check the remaining equations in the `VerifyCT` algorithm as follows:

$$C[0]^c \cdot g_2^{\hat{\gamma}} = g_2^{\gamma c} g_2^{\gamma' - \gamma c} = g_2^{\gamma'} = C'[0]$$

$$C[1]^c \circ \mathbf{A}^{\hat{\mathbf{v}}} = \mathbf{A}^{\mathbf{v}c} \circ \mathbf{A}^{\mathbf{v}' - \mathbf{v}c} = \mathbf{A}^{\mathbf{v}'} = C'[1]$$

$$C[2]^c \mathbf{b}^{\hat{\mathbf{v}}} g_t'^{\hat{s}} = \mathbf{b}^{\mathbf{v}c} g_t'^{sc} \mathbf{b}^{\mathbf{v}' - \mathbf{v}c} g_t'^{s' - cs} = \mathbf{b}^{\mathbf{v}'} g_t'^{s'} = C'[2]$$

Further, completeness indicates that the ciphertext is well-formed. □

Lemma 5 (NIZK Proof Soundness II). *Given two accepting transcripts* $((C, \pi_{s1} = (C', c_1, \hat{\gamma}_1, \hat{\mathbf{v}}_1, \hat{s}_1)))$ *and* $((C, \pi_{s2} = (C', c_2, \hat{\gamma}_2, \hat{\mathbf{v}}_2, \hat{s}_2)))$*, an efficient extractor can output the witness* γ, $\mathbf{v}$ *and* s.

Proof : Since the two transcripts are accepting, then we have

$$\hat{\gamma}_i = \gamma' - c_i \gamma, \quad \hat{\mathbf{v}}_i = \mathbf{v}' - c_i \mathbf{v}, \quad \hat{s}_i = s' - c_i s.$$

where $i = 1, 2$. Then, by subtracting the respective equations to have:

$$\hat{\gamma}_1 - \hat{\gamma}_2 = (c_2 - c_1)\gamma, \quad \hat{\mathbf{v}}_1 - \hat{\mathbf{v}}_2 = (c_2 - c_1)\mathbf{v}, \quad \hat{s}_1 - \hat{s}_2 = (c_2 - c_1)s.$$

Since $c_1 \neq c_2$, we can invert $(c_2 - c_1)$ in the underlying field, yielding:

$$\gamma = (\hat{\gamma}_1 - \hat{\gamma}_2)(c_2 - c_1)^{-1}, \quad \mathbf{v} = (\hat{\mathbf{v}}_1 - \hat{\mathbf{v}}_2)(c_2 - c_1)^{-1}, \quad s = (\hat{s}_1 - \hat{s}_2)(c_2 - c_1)^{-1}.$$

This extraction is computable and efficient. The resulting $(\gamma, \mathbf{v}, s)$ satisfies the witness relation by construction, completing the proof. □

Theorem 6 (Correctness). *The CCATE described in Figure 5 satisfies correctness, defined by Definition 2.*

Proof : For honestly generated ciphertexts, Lemma 4 gives $\texttt{VerifyCT}(C, \pi_s) = \mathsf{true}$. If the public keys pass `VerifySetup`, then $\mathsf{pk}_{i,2} = (h_{i,1}, h_{i,2}, h_{i,3}, h_{i,4}, \{h_{i,5,j}\})$ is well formed by the pairing checks. For every honest partial decryption, $D_i = C[0]^{\mathsf{sk}_i} = g_2^{\gamma \mathsf{sk}_i}$, and the attached DLEQ proof is accepted by `VerifyPD` by completeness of the DLEQ protocol. The calculation of w is described in Appendix E, and the correctness of w follows from Lemma 8. Then reconstruction is correct:

$$C[2] \cdot e(C[1], \mathsf{w})^{-1} = \mathbf{b}^{\mathbf{v}} \cdot g_t'^{s} \cdot e(\mathbf{A}^{\mathbf{v}}, \mathsf{w})^{-1} = \mathbf{b}^{\mathbf{v}} \cdot g_t'^{s} \cdot e(\mathbf{A}, \mathsf{w})^{-\mathbf{v}} = g_t'^{s} = S.$$

Thus $\Pr[(s, S) \in \mathcal{PK}] = 1$. □

Theorem 7 (Robustness). *The CCATE described in Figure 5 satisfies robustness, defined by Definition 3.*

Proof : Let $C = (C[0], C[1], C[2])$ be any ciphertext accepted by `VerifyCT`, and let Q, Q' be two qualified sets of accepted partial decryption shares. Let

$$S_Q = \texttt{Decrypt}(C, \{(i, D_i)\}_{i \in Q}), \qquad S_{Q'} = \texttt{Decrypt}(C, \{(i, D'_i)\}_{i \in Q'}).$$

Let $\epsilon_{\mathsf{dleq\text{-}snd}}(\lambda)$ denote the total soundness failure probability that any invalid DLEQ proof is accepted over all polynomially many partial-decryption verification attempts made by a PPT adversary. Except with this probability, every accepted share satisfies $D_i = C[0]^{\mathsf{sk}_i} = g_2^{\gamma \mathsf{sk}_i}$. Conditioned on no DLEQ-soundness failure, the accepted shares determine the restricted secret-key polynomial values used by the STE reconstruction algorithm. For any fixed accepted set Q, the support polynomial $B_Q(X)$ is uniquely determined by the interpolation constraints $B_Q(\omega^0) = 1$ and $B_Q(\omega^i) = 0$ for $i \notin Q$. Let $\epsilon_{\mathsf{usc}}(\lambda)$ denote the total algebraic soundness failure probability of all univariate-sumcheck reconstruction attempts. Except with this probability, Lemma 8 gives a unique bounded-degree decomposition $\mathsf{SK}(X)B_Q(X) = \frac{\mathsf{sk}'_Q}{n+1} + Q_{Z,Q}(X)Z(X) + Q_{x,Q}(X)X$, and hence the vector w_Q computed by `Decrypt` satisfies the same public linear relation $e(\mathbf{A}, \mathsf{w}_Q) = \mathbf{b}$. The same argument applies to Q'. Therefore, conditioned on neither failure event occurring, both reconstructions remove the same mask from the well-formed ciphertext: $C[2] \cdot e(C[1], \mathsf{w}_Q)^{-1} = \mathbf{b}^{\mathbf{v}} g_t'^{s} \cdot e(\mathbf{A}^{\mathbf{v}}, \mathsf{w}_Q)^{-1} = g_t'^{s}$, and identically for Q'. Thus two accepted qualified reconstructions can differ only if a DLEQ soundness failure or a univariate-sumcheck soundness failure occurs. Therefore

$$\Pr[S_Q \neq S_{Q'}] \leq \epsilon_{\mathsf{dleq\text{-}snd}}(\lambda) + \epsilon_{\mathsf{usc}}(\lambda),$$

which is negligible. □

Theorem 8 (Public Verifiability). *The CCATE described in Figure 5 satisfies public verifiability, defined by Definition 4.*

Proof : Public verifiability in CCATE is proven for each phase of the protocol.

- Property **PV1** (setup validity) is implied by the public verifiability of the Power-Of-Tau ceremony and the public keys from silent setup, which is established in Section 5.1.
- Property **PV2** (ciphertext validity) is implemented by the public algorithm `VerifyCT`. On input $\pi_s = (C', c, \hat{\gamma}, \hat{\mathbf{v}}, \hat{s})$, the verifier first checks $c = H(\ell_{\mathsf{inst}_2}, C, C')$, and then checks the three public algebraic equations in Figure 5. Completeness and soundness follow from Lemma 4 and Lemma 5.
- Property **PV3** (partial-decryption validity) is implemented by the public algorithm `VerifyPD`, which uses the DLEQ proof to verify a partial decryption share without secret information. □

Lemma 6 (NIZK Proof Zero-Knowledge II). *Given input (C, c), a simulator can output a transcript $((\tilde{C}', c, \tilde{\gamma}, \tilde{\mathbf{v}}, \tilde{s}))$ identical to real-world output between an honest dealer and an honest shareholder.*

Proof : We construct a simulator that, given $C = (C[0], C[1], C[2]) = (g_2{}^{\gamma},\ \mathbf{A}^{\mathbf{v}},\ \mathbf{b}^{\mathbf{v}} \cdot g_t'{}^{s})$ and a challenge $c \in \mathbb{Z}_p$, outputs a valid transcript $((\tilde{C}', c, \tilde{\gamma}, \tilde{\mathbf{v}}, \tilde{s}))$ without knowing γ, $\mathbf{v}$, or s.

1. Sample random values: $\tilde{\gamma} \xleftarrow{\$} \mathbb{Z}_p, \quad \tilde{\mathbf{v}} \xleftarrow{\$} \mathbb{Z}_p^5, \quad \tilde{s} \xleftarrow{\$} \mathbb{Z}_p$.
2. Compute the commitment $\tilde{C}'$ as: $\tilde{C}' = (C'[0], C'[1], C'[2]) = (C[0]^c g_2^{\tilde{\gamma}},\ C[1]^c \circ \mathbf{A}^{\tilde{\mathbf{v}}},\ C[2]^c \mathbf{b}^{\tilde{\mathbf{v}}} g_t'{}^{\tilde{s}})$. This ensures that the verification equation will pass with the response $(\tilde{\gamma}, \tilde{\mathbf{v}}, \tilde{s})$ and challenge c.
3. Program the random oracle so that $H(\ell_{\mathsf{inst}_2}, C, \tilde{C}') = c$. As in the standard Fiat–Shamir proof in the random oracle model, the probability that this labelled point has been queried before is negligible for fresh $\tilde{C}'$.
4. Output the simulated transcript: $(\tilde{C}', c, \tilde{\gamma}, \tilde{\mathbf{v}}, \tilde{s})$.

In the real transcript, (γ, γ', s') are uniformly random in $\mathbb{Z}_p$, and $\mathbf{v}, \mathbf{v}'$ are uniformly random in $\mathbb{Z}_p^5$. Thus, $C' = (g_2^{\gamma'}, \mathbf{A}^{\mathsf{v}'}, \mathbf{b}^{\mathsf{v}'} \cdot g_t'{}^{s'})$ is uniformly random. Since $c = H(\ell_{\mathsf{inst}_2}, C, C')$ is generated by the random oracle, $\hat{\gamma} = \gamma' - c\gamma, \hat{\mathbf{v}} = \mathsf{v}' - c\mathbf{v}, \hat{s} = s' - cs$ are uniform. In the simulation, $(\tilde{\gamma}, \tilde{\mathbf{v}}, \tilde{s})$ is randomly chosen from $\mathbb{Z}_p$ and distributed identically to $(\hat{\gamma}, \hat{\mathbf{v}}, \hat{s})$. $\tilde{C}'$ is also uniformly random since $(\tilde{\gamma}, \tilde{\mathbf{v}}, \tilde{s})$ is randomly chosen from $\mathbb{Z}_p$. Thus, the distribution of simulated transcript $((\tilde{C}', c, \tilde{\gamma}, \tilde{\mathbf{v}}, \tilde{s}))$ is identical to that of a real transcript. □

Lemma 7 (Underlying CPA-STE assumption.). *We follow the semantic security of Construction 1 of Garg et al. [9, Theorem 2], proved in the generic group model for their pairing-based STE construction. We denote by* $\mathbf{Adv}^{\mathsf{cpa}}_{\mathsf{STE},\mathcal{B}}$ *the CPA advantage of an adversary* $\mathcal{B}$ *against the STE ciphertext component.*

Theorem 9 (CCA2 Security). *The CCATE instantiation II described in Figure 5 is* ***CCA2*** *secure against the game in Definition 5 in the random oracle model, assuming the ciphertext NIZK has labelled simulation-sound online extractability, the DLEQ proof is sound and zero-knowledge, and the underlying CPA-STE ciphertext component satisfies the security guaranteed by Lemma 7*

Proof : Let $\mathcal{A}$ be any PPT adversary in the CCA2 experiment of Definition 5. We use a game-based reduction to the security of the underlying CPA-STE scheme. Let W_j denote the event that $\mathcal{A}$ guesses the challenge bit in Game G_j.

Game G_0. This is the real CCA2 experiment for Figure 5. The challenger runs the silent setup, publishes pp and all verified public keys, answers static corruptions with at most $t-1$ secret keys, answers partial-decryption queries using $\texttt{PartialDec}$, samples $b \xleftarrow{\$} \{0,1\}$, gives $(C^*, \pi^*) \leftarrow \texttt{Encrypt}(s_b, \mathsf{pp})$ to $\mathcal{A}$, and rejects any post-challenge partial-decryption query whose ciphertext component is C^*. Therefore, $\mathbf{Adv}^{\mathsf{cca2}}_{\Pi,\mathcal{A}} = \left|\Pr[W_0] - \frac{1}{2}\right|$.

Game G_1. This game is identical to G_0, except that every partial-decryption query (C, π) is first checked by $\texttt{VerifyCT}(C, \pi)$. If verification fails, the challenger returns $\perp$. This is the explicit rejection behavior of the real algorithm, and hence $\Pr[W_1] = \Pr[W_0]$.

Game G_2. This game changes the answers to valid non-challenge partial-decryption queries. For each query (C, π) with $C \neq C^*$ and $\texttt{VerifyCT}(C, \pi) = \mathsf{true}$, the challenger applies the labelled simulation-sound online extractor for the ciphertext NIZK under label ℓ_{inst_2}. Except with probability ϵ_{ext}, it obtains a witness $(\gamma, \mathbf{v}, s)$ satisfying the ciphertext relation $C = (g_2^{\gamma}, \mathbf{A}^{\mathbf{v}}, \mathbf{b}^{\mathbf{v}} \cdot g_t'^{s})$. The challenger then computes every decryption share using only public keys: $D_i = (\mathsf{pk}_{i,1}^{(2)})^{\gamma} = (g_2^{\mathsf{sk}_i})^{\gamma} = C[0]^{\mathsf{sk}_i}$. It replaces the real DLEQ proofs with simulated DLEQ proofs. The decryption shares are identically distributed to the real shares in G_1, and the proof replacement is indistinguishable by DLEQ zero-knowledge. Thus $|\Pr[W_2] - \Pr[W_1]| \leq \epsilon_{\mathsf{ext}} + \epsilon_{\mathsf{dleq\text{-}zk}}$.

Game G_3. This game is identical to G_2, except that the challenge proof π^* is simulated using Lemma 6. The simulator samples the responses first and programs the random oracle at the fresh labelled point $(\ell_{\mathsf{inst}_2}, C^*, C'^*)$. Let ϵ_{ro} be the probability that this labelled point was queried before programming. By zero-knowledge of the ciphertext proof, $|\Pr[W_3] - \Pr[W_2]| \leq \epsilon_{\mathsf{zk}} + \epsilon_{\mathsf{ro}}$. Because the CCA2 experiment still allows post-challenge partial-decryption queries for ciphertexts $C \neq C^*$, these non-challenge queries are answered exactly as in G_2, by extracting their witnesses under the same label ℓ_{inst_2}. Labelled simulation-sound online extractability is needed because the adversary has seen the simulated challenge proof and may later submit other accepting non-challenge proofs.

Game G_4. This game replaces the STE encryption component of the challenge ciphertext with a CPA challenge ciphertext of the underlying CPA-STE scheme. More explicitly, a CPA challenger for STE receives the two messages s_0, s_1 and returns an STE challenge ciphertext encrypting s_b; the game attaches a simulated ciphertext proof π^* as in G_3. All partial-decryption queries for non-challenge ciphertexts are still answered by extraction as in G_2, and therefore the game does not require the honest secret keys of uncorrupted parties. Let $\mathcal{B}$ be an adversary against the underlying CPA-STE scheme. $\mathcal{B}$ internally runs $\mathcal{A}$, forwards $\mathcal{A}$'s challenge messages s_0, s_1 to its CPA challenger, receives the CPA challenge ciphertext C^*, simulates π^* using Lemma 6, and answers every valid non-challenge partial-decryption query by extracting γ and returning $D_i = (\mathsf{pk}_{i,1}^{(2)})^{\gamma}$ with simulated DLEQ proofs. If the CPA challenge is a real encryption, $\mathcal{A}$'s view is G_3; if it hides the challenge bit according to CPA security, $\mathcal{A}$'s view is G_4. Hence $|\Pr[W_3] - \Pr[W_4]| \leq \mathbf{Adv}^{\mathsf{cpa}}_{\mathsf{STE},\mathcal{B}}$. In G_4, the STE challenge hides b and the attached proof is simulated, while all partial-decryption oracle answers are independent of b. Therefore $\Pr[W_4] = 1/2$.

Combining the game transitions, we obtain

$$\mathbf{Adv}^{\mathsf{cca2}}_{\Pi,\mathcal{A}} \leq \mathbf{Adv}^{\mathsf{cpa}}_{\mathsf{STE},\mathcal{B}} + \epsilon_{\mathsf{ext}} + \epsilon_{\mathsf{dleq\text{-}zk}} + \epsilon_{\mathsf{zk}} + \epsilon_{\mathsf{ro}},$$

which is negligible under the CPA security of CPA-STE and the stated simulation-sound extractability and zero-knowledge assumptions. □

Discussion on Adaptive security. We remark that achieving fully adaptive security threshold encryption in the silent setting remains an open problem. The best known result in this direction is due to Hall-Andersen, Simkin, and Wagner [42], who introduced a relaxed corruption model called *one-shot adaptive*

corruptions, which lies conceptually between static and fully adaptive corruptions. Following their blueprint, our construction can be strengthened to achieve one-shot adaptive security.

6 Discussion and Comparison with Related Works

6.1 Trade-offs in Setup

In ECC-based PVSS schemes, exponentiation and bilinear pairing are the dominant computational operations. We denote E and P as the cost of an exponentiation and a bilinear pairing operation, respectively. For brevity, we use E to denote exponentiation operations of ECC groups (i.e., $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$).

Table 1. Comparison of setup between the proposed two PVSS constructions

Setup	Implementation	Assumption	Aggregator	Computation	Communication
Instantiation I (Fig. 11)	Committee-based setup	Publicly auditable setup; $< t$ corrupted parties; honest setup entropy	–	$O(t \cdot N)E$	$O(t \cdot N)\lvert\mathbb{G}\rvert$ $+O(N)\lvert\mathbb{Z}_p\rvert$
Instantiation II (Fig. 4)	Power-of-Tau & silent setup	honest parties	honest-but-curious	$O(n^2)E$	$O(n)E$

Previous PVSS schemes assume a Public Key Infrastructure (PKI) and typically require minimal communication and computational overhead during setup. In contrast, our scheme reduces the complexity of distribution and verification to constant levels, albeit at the cost of increased setup complexity. Table 1 compares the setup procedures for constructing CCATE and PVSS schemes. The setup realization in Figure 11 leverages an external N-size committee to produce threshold key shares for n parties, assuming the resulting transcript satisfies Definition 6. In this realization, each committee member invokes Feldman VSS, regarding each party as a shareholder. Each party requires $O(t)$ exponentiations to check the correctness of its share against commitments to the coefficients, i.e., $\{g^{a_i}\}_{i\in[0,t-1]}$. Hence, the computation complexity is $O(t \cdot N)E$ and communication complexity is $O(t \cdot N)|\mathbb{G}|+O(N)|\mathbb{Z}_p|$ for each party. Membership changes are handled at epoch boundaries by a fresh setup or a standard resharing step that re-establishes the setup assumption for the new group. Figure 4 illustrates the silent setup DKG protocol, which enables each party to independently sample its secret/public key pair without any interaction with other parties. Each honest party must generate and publish $O(n)$ group elements (i.e., $h_{i,5,j}$), where the computation of each element requires $O(n)$ exponentiations due to evaluation of polynomial on $\mathbb{G}$ (refer to Definition 7). This results in a total computational cost of $O(n^2)$ exponentiations per party. However, this approach relies on a precomputed Power-of-Tau CRS and requires an honest-but-curious aggregator to validate and aggregate public keys, producing aggregated verification structures.

Table 1 presents a clear efficiency trade-off between the committee-based setup realization and silent setup. The former uses an interactive setup ceremony, resulting in bandwidth overhead that scales with the number of parties. In contrast, the silent setup instantiation removes inter-party setup communication, but it does not remove setup assumptions. It relies on a publicly verifiable Power-of-Tau ceremony in which at least one contributor honestly destroys its trapdoor contribution. It also requires cross-group pairing checks, which makes setup verification heavier than in the pairing-free instantiation.

6.2 Comparison with State-of-The-Art PVSS

Publicly Verifiable Secret Sharing (PVSS) was introduced by Stadler [1], and Schoenmakers later gave a simple ElGamal-based construction with $O(nt)$ public verification [2]. Subsequent work reduced the online verification bottleneck to linear complexity. SCRAPE [4] uses Reed–Solomon-code checks to certify that encrypted shares are consistent with a common sharing polynomial. ALBATROSS [5] builds on this line with packed Shamir sharing and low-degree exponent interpolation, improving amortized costs when many secrets are shared together. YOLO YOSO [6] introduces HEPVSS and DHPVSS: HEPVSS uses homomorphic encryption to support efficient proof aggregation, while DHPVSS obtains a compact proof object by checking a Fiat–Shamir-derived linear combination of encrypted shares. These schemes still publish or process shareholder-dependent online data, and hence their dealer work and public verification remain $O(n)$.

Our comparison in Table 2 focuses on this online cost. The proposed constructions move the shareholder-dependent structure into a reusable, publicly auditable setup and make each online sharing a single publicly verifiable threshold-encryption ciphertext. As a result, distribution and public verification are independent of n, while partial-decryption collection and final reconstruction remain threshold-dependent, as in prior PVSS schemes. This is an amortized setup/online trade-off rather than a replacement for schemes optimized for minimal setup.

In Figure 3, each ElGamal-style ciphertext, i.e., C and C', incurs a computational cost of $3E$ in the `Encrypt` algorithm. Consequently, the total encryption cost is $6E$. The size of the output (C, π_s) is $4|\mathbb{G}| + 3|\mathbb{Z}_p|$. In the `VerifyCT` algorithm, the total verification cost amounts to $5E$: verifying $C[0]$ requires $3E$, while verifying $C[1]$ requires $2E$. In Figure 5, each non-zero element in matrix scalar exponentiation requires $1E$, leading to a total of $30E$ in the calculation of C and C' in the `Encrypt` algorithm. Since $C[0]$ and $C[2]$ are single element and $C[1]$ is a tuple containing 8 group elements. Thus, the `VerifyCT` algorithm requires $25E$. The `Decrypt` algorithm requires $O(n)E$, because `w` requires calculation of $Q(x) = g_1^{Q_x(\tau)}$ (refer to the details in Figure 12), where $Q_x(X)$ is a n-degree polynomial.

SCRAPE [4] is a representative linear-verification PVSS scheme. The dealer shares a secret with a degree-$(t-1)$ polynomial, encrypts one share per party, and

Table 2. PVSS complexity comparison regarding online operations

Ref.	Dist. comp.	Dist. comm.	Verif. comp.	Recon. comp.	Recon. comm.
SCRAPE$_{DDH}$ [4]	$O(n)E$	$O(n)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$	$O(n)E$	$O(t)E$	$O(t)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$
SCRAPE$_{DBS}$ [4]	$O(n)E$	$O(n)\|\mathbb{G}\|$	$O(n)E+O(n)P$	$O(t)E+O(t)P$	$O(t)\|\mathbb{G}\|$
ALBATROSS [5]	$O(n)E$	$O(n)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$	$O(n)E$	$O(\ell(t+\ell))E$ for ℓ secrets	$O(t)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$
HEPVSS [6]	$O(n)E$	$O(n)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$	$O(n)E$	$O(t)E$	$O(t)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$
DHPVSS [6]	$O(n)E$	$O(n)\|\mathbb{G}\|$ $+O(1)\|\mathbb{Z}_p\|$	$O(n)E$	$O(t)E$	$O(t)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$
Instantiation I (Fig. 3, pairing-free)	$\mathbf{O(1)}E$ $6E$	$\mathbf{O(1)}$ $4\|\mathbb{G}\|+3\|\mathbb{Z}_p\|$	$\mathbf{O(1)}E$ $5E$	$O(t)E$	$O(t)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$
Instantiation II (Fig. 5, silent setup)	$\mathbf{O(1)}E$ $30E$	$\mathbf{O(1)}$ $20\|\mathbb{G}\|+8\|\mathbb{Z}_p\|$	$\mathbf{O(1)}E$ $25E$	$O(n)E$	$O(t)(\|\mathbb{G}\|+\|\mathbb{Z}_p\|)$

publishes commitments that allow public consistency checks. In SCRAPE$_{DDH}$, correctness of encrypted shares is checked with DLEQ-style proofs; in SCRAPE$_{DBS}$, the corresponding check is pairing based. A verifier then uses a dual-code test to check that the encrypted shares are consistent with a low-degree sharing polynomial.

ALBATROSS [5] improves amortized efficiency by packing ℓ secrets into one polynomial of degree at most $t+\ell-1$. Its low-degree exponent interpolation proof certifies that the encrypted shares are evaluations of a polynomial of the advertised degree. This reduces the per-secret cost in batched settings, but the online transcript still contains shareholder-dependent encrypted-share information.

YOLO YOSO [6] further simplifies the verification machinery. In HEPVSS, the dealer encrypts Shamir shares and proves that they are consistent with a valid polynomial. In DHPVSS, group-valued Shamir shares are masked using a dealer distribution key and the parties' public keys; public verification checks a Fiat–Shamir-derived linear combination using a single DLEQ proof over aggregated terms. DHPVSS is pairing-free and has a compact proof, but sharing and verification still scale linearly with the number of participants.

6.3 PVSS with Secret Recovery on $\mathbb{Z}_p$

The schemes compared above, and our constructions in this paper, reconstruct a group-encoded value g^s rather than the field element s itself. This is sufficient for threshold openings, randomness beacons, and applications that consume group encodings. Other VSS/PVSS lines target the stronger functionality of recovering secrets in $\mathbb{Z}_p$, which is important for DKG and YOSO-style MPC.

Groth's non-interactive DKG/PVSS approach [44] supports recovery of field elements by using exponentiated ElGamal together with bit decomposition and proofs of correct chunking. This gives strong functionality but large concrete costs, since the dealer must prove that encrypted chunks encode a valid field element. Kate et al.'s cgVSS [40] replaces this cyclic-group bottleneck with class-group techniques [46], allowing large field elements to be handled more directly

and substantially reducing the concrete overhead, while preserving linear-size shareholder-dependent transcripts.

qCLPVSS [7] also uses class groups and builds on the DHPVSS design philosophy to obtain PVSS with secret recovery over $\mathbb{Z}_p$, making it suitable for DKG and YOSO applications. Das et al. [45] pursue a different point in the design space: their Reveal-upon-No-Ack paradigm achieves efficient verifiable secret sharing by combining private distribution, acknowledgments, and a public transcript that reveals shares only for non-responding parties. This improves practical costs but uses dealer-shareholder interaction, whereas our focus is non-interactive online sharing with constant public distribution and verification after setup.

7 Conclusion

We presented a generic framework that separates the reusable setup work of PVSS from its online sharing phase. The main technical message is that a publicly verifiable CCA2-secure threshold encryption scheme can be transformed into a PVSS scheme whose online distribution and public-verification costs are independent of the number of shareholders. We instantiated the framework in two complementary ways. The Threshold-ElGamal-based construction gives a pairing-free realization under a committee-based setup assumption, while the silent-setup construction removes inter-party communication from setup at the cost of stronger setup machinery and heavier reconstruction. Both constructions should be viewed as explicit setup/online trade-offs: expensive public setup material is amortized across many sharings, and membership changes are handled through an epoch-based model rather than fully dynamic security. This perspective clarifies a design point for PVSS in blockchain and distributed-system applications. When setup transcripts can be publicly audited and reused, the online bottleneck need not scale with the number of shareholders. The resulting constructions reduce the critical dealer and public-verification costs while retaining formal correctness, robustness, public verifiability, and secrecy guarantees under the stated assumptions.

Note on AI Assistance. The authors used ChatGPT to improve the clarity, grammar, and overall readability of the manuscript. All AI-generated suggestions were reviewed, edited, and approved by the authors, who take full responsibility for the final content.

References

1. Stadler, M. Publicly verifiable secret sharing. In *Eurocrypt*, 1996, pp. 190-199.
2. Schoenmakers, B. A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. In *CRYPTO*, 1999, pp. 148-164.
3. Baghery, K., Knapen, N., Nicolas, G. and Rahimi, M. Pre-constructed publicly verifiable secret sharing and applications. In *ACNS*, 2025, pp. 89-119.

4. Cascudo, I. and David, B. SCRAPE: Scalable Randomness Attested by Public Entities. In *ACNS*, 2017, pp. 537-556.
5. Cascudo, I. and David, B. ALBATROSS: Publicly Attestable Batched Randomness Based on Secret Sharing. In *ASIACRYPT*, 2020, pp. 311-341.
6. Cascudo, I., David, B., Garms, L. and Konring, A. YOLO YOSO: Fast and Simple Encryption and Secret Sharing in the YOSO Model. In *ASIACRYPT*, 2022, pp. 651-680.
7. Cascudo, I. and David, B. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In *Eurocrypt*, 2024, pp. 216-248.
8. Groneberg, B. Threshold Cryptography. Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam, p.5.
9. Garg, S., Kolonelos, D., Policharla, G.V. and Wang, M. Threshold encryption with silent setup. In *CRYPTO*, 2024, pp. 352-386.
10. Zhang, L., Wu, X., Ma, Y. and Kan, H. Data exchange for the metaverse with accountable decentralized TTPs and incentive mechanisms. *IEEE TBD*, 2025, 11(5): 2431-2442.
11. Nikolaenko, V., Ragsdale, S., Bonneau, J. and Boneh, D. Powers-of-tau to the people: Decentralizing setup ceremonies. In *ACNS*, 2024, pp. 105-134.
12. Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T. and Reyzin, L. Can a public blockchain keep a secret?. In *TCC*, 2020, pp. 260-290.
13. Goyal, V., Kothapalli, A., Masserova, E., Parno, B. and Song, Y. Storing and retrieving secrets on a blockchain. In *PKC*, 2022, pp. 252-282.
14. Kokoris-Kogias, E., Alp, E.C., Gasser, L., Jovanovic, P., Syta, E. and Ford, B. Calypso: Private data management for decentralized ledgers. *Proc. VLDB Endow.*, 2020, pp. 586599.
15. Zyskind, G., Yanai, A. and Pentland, A. Unstoppable wallets: Chain-assisted threshold ECDSA and its applications. In *Asia CCS*, 2024, pp. 1844-1860.
16. Damgård, I. Commitment schemes and zero-knowledge protocols. In School organized by the European Educational Forum, 1998, pp. 63-86.
17. Das, S., Krishnan, V., Isaac, I. M. and Ren, L. Spurt: Scalable distributed randomness beacon with transparent setup. In *S&P*, 2022, pp. 2502-2517.
18. Bhat, A., Shrestha, N., Kate, A. and Nayak, K. OptRand: Optimistically Responsive Reconfigurable Distributed Randomness. In *NDSS*, 2023.
19. Bacho, R., Lenzen, C., Loss, J., Ochsenreither, S. and Papachristoudis, D. GRandLine: adaptively secure DKG and randomness beacon with (log-)quadratic communication complexity. In *CCS*, 2024, pp. 941-955.
20. Baghery, K., Knapen, N., Nicolas, G. and Rahimi, M. Pre-constructed publicly verifiable secret sharing and applications. In *ACNS*, 2025, pp. 89-119.
21. Saereesitthipitak, S. and Zindros, D. Cassiopeia: Practical On-Chain Witness Encryption. In *FC*, 2023, pp. 385-404.
22. Kiayias, A., Russell, A., David, B. and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO*, 2017, pp. 357-388.
23. Ballweg, J., Cai, Z. and Goharshady, A.K., 2023, December. PureLottery: Fair leader election without decentralized random number generation. In *ICBC*, 2023, pp. 273-280.
24. Zhang, L., Liu, T., Kan, H. and Zhang, J. AsyRand: asynchronous distributed randomness beacon with reconfiguration. *Cryptology ePrint Archive*, 2025.
25. Sahai, A. and Waters, B. Fuzzy identity-based encryption. In *Eurocrypt*, 2005, pp. 457-473.

26. Goyal, V., Pandey, O., Sahai, A. and Waters, B, October. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, 2006, pp. 89-98.
27. Fiat, A. and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Eurocrypt*, 2025, pp. 186-194.
28. Fischlin, M. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO*, 2005, pp. 152-168.
29. Chaum, D. and Pedersen, T.P. Wallet databases with observers. In *CRYPTO*, 1992, pp. 89-105.
30. Naor, M. and Yung, M. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, 1990, pp. 427-437.
31. Fujisaki, E. and Okamoto, T. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, 1999, pp. 537-554.
32. Cramer, R. and Shoup, V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, 1998, pp. 13-25.
33. Damgård, I. On Σ-Protocols, `https://www.cs.au.dk/~ivan/Sigma.pdf`, Accessed: 2025-09-16
34. Libert, B. and Yung, M. Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In *CRYPTO*, 2012, pp. 75-93.
35. Das, S., Yurek, T., Xiang, Z., Miller, A., Kokoris-Kogias, L. and Ren, L. Practical asynchronous distributed key generation. In *S&P*, 2022, pp. 2518-2534.
36. Gurkan, K., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G. and Tomescu, A. Aggregatable distributed key generation. In *Eurocrypt*, 2024, pp. 147-176.
37. Feng, H., Mai, T. and Tang, Q. Scalable and adaptively secure any-trust distributed key generation and all-hands checkpointing. In *CCS*, 2024, pp. 2636-2650.
38. Das, S., Ren, L., and Yang, Z. Adaptively secure threshold ElGamal decryption from DDH. *Cryptology ePrint Archive*, 2025.
39. Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In *SFCS*, 1987, pp. 427-438.
40. Kate, A., Mangipudi, E.V., Mukherjee, P., Saleem, H. and Thyagarajan, S.A.K. Non-interactive VSS using class groups and application to DKG. In *CCS*, 2024, pp. 4286-4300.
41. Cascudo, I., Cozzo, D. and Giunta, E. Verifiable secret sharing from symmetric key cryptography with improved optimistic complexity. In *ASIACRYPT*, 2024, pp. 100-128.
42. Hall-Andersen, M., Simkin, M., and Wagner, B. Silent threshold encryption with one-shot adaptive security. *Cryptology ePrint Archive*, 2025.
43. Franklin, M. and Yung, M. Communication complexity of secure computation. In *STOC*, 1992, pp. 699-710.
44. Groth, J. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, 2021.
45. Das, S., Xiang, Z., Tomescu, A., Spiegelman, A., Pinkas, B. and Ren, L. Verifiable secret sharing simplified. In *SP*, 2025, pp. 633-651.
46. Castagnos, G. and Laguillaumie, F. Linearly homomorphic encryption from. In *CT-RSA*, 2015, pp. 487-505.
47. Kate, A., Zaverucha, G.M. and Goldberg, I., 2010, December. Constant-size commitments to polynomials and their applications. In *Asiacrypt*, 2010, pp. 177-194.
48. Zhang, J., Xie, T., Hoang, T., Shi, E. and Zhang, Y., 2022. Polynomial commitment with a One-to-Many prover and applications. In *USENIX Security*, 2022, pp. 2965-2982.

49. Zhang, Z., Li, W., Guo, Y., Shi, K., Chow, S.S., Liu, X. and Dong, J. Fast RS-IOP multivariate polynomial commitments and verifiable secret sharing. In *USENIX Security*, 2024, pp. 3187-3204.
50. Ràfols, C. and Zapico, A. An algebraic framework for universal and updatable SNARKs. In *CRYPTO*, 2021, pp. 774-804.

A Experimental Results

We implement the proposed PVSS schemes in Rust using the arkworks library. All experiments were conducted on a Mac Air equipped with an M4 Max chip, 10-core CPU and 16 GB of RAM, running macOS 26.3 in single-threaded mode. We employ the BLS12-381 elliptic curve, set the recovery threshold to $t = \lfloor n/2 \rfloor + 1$, and vary n from 100 to 1000 in steps of 100. For Instantiation I, the setup committee size is fixed to $N = 64$. To highlight our strengths, we compare the proposed PVSS schemes with DHPVSS [6], the most recent state-of-the-art construction. Figures 6 to 10 report the online costs of sharing, public verification, partial decryption, partial-decryption verification, and reconstruction, averaged over ten repetitions. One-time setup, setup verification, and reusable interpolation precomputation are excluded from these online measurements.

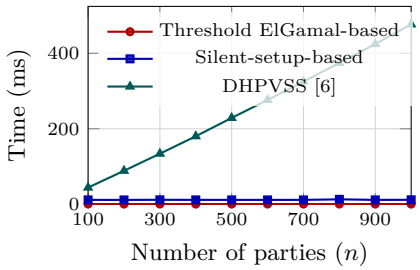


Fig. 6. Cost of PVSS.Share

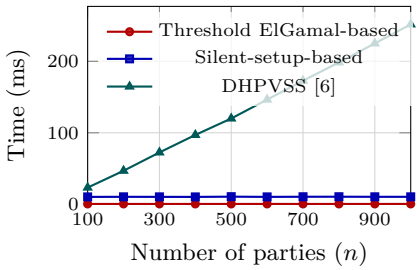


Fig. 7. Cost of PVSS.Verify

The measurements confirm the intended online behavior. The two proposed instantiations keep PVSS.Share and PVSS.Verify essentially constant as n grows, while DHPVSS grows linearly in both algorithms. Partial-decryption generation and verification are also constant per participating party. Reconstruction scales with the recovery threshold, as expected; the silent-setup instantiation has a larger reconstruction constant due to its pairing-based aggregation step.

B Feldman VSS

Feldman VSS scheme [39] enables the dealer to be privately verifiable.

– $(\mathbb{G}, p, g) \leftarrow \mathsf{VSS.Setup}(1^\lambda)$: g is a generator of group $\mathbb{G}$ with prime order p.

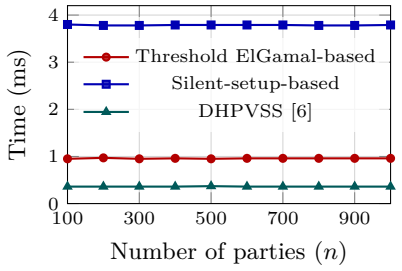


Fig. 8. Cost of PVSS.PreRecon

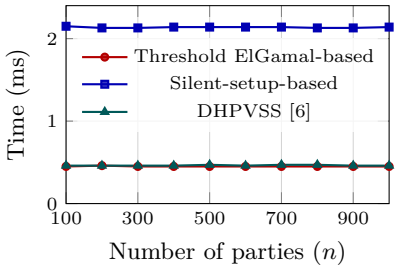


Fig. 9. Cost of PVSS.DecShareVrf

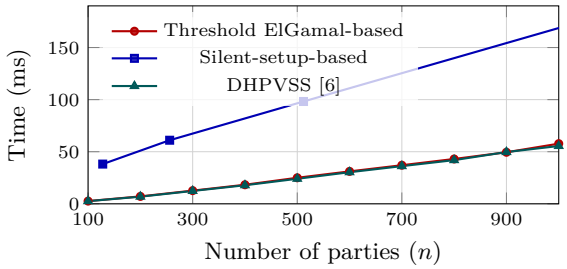


Fig. 10. Cost of PVSS.Recon

- $(\{i, s_i, \{gs_i\}_{i\in[1,n]}, \{Com_j\}_{j\in[0,t-1]}\}) \leftarrow \mathsf{VSS.Share}(s, n, t)$: Choose a polynomial $f(x) = s + \sum_{i=1}^{t-1} a_i x^i \pmod p$, compute $s_i = f(i)$ and $Com_j = g^{a_j} \bmod p$. The dealer broadcasts $(\{gs_i = g^{s_i}\}, \{Com_j\})$ and sends (i, s_i) securely to the shareholder i.
- true or $\mathsf{false} \leftarrow \mathsf{VSS.Verify}(i, s_i, \{gs_i\}, \{Com_j\})$: Each shareholder tests whether $gs_i \stackrel{?}{=} g^{s_i} \stackrel{?}{=} \prod_{j=0}^{t-1} (Com_j)^{i^j} \ mod\ p$ holds. If not, the shareholder broadcasts a complaint against the dealer. Note that if a dealer is malicious, it needs to publish the received s_i in the complaint phase.
- $s \leftarrow \mathsf{VSS.Reconstruct}(\{(j, s_j)\}_{j\in Q})$: With $|Q| \geq t$, it recovers s using the Lagrange interpolation algorithm Intpl.

C Committee-Based Setup Implementation

This appendix gives one concrete realization of Definition 6 and records the corresponding algebraic checks. The material is included only to make the setup assumption concrete; it is not claimed as a new standalone DKG contribution. [4]

[4] Any setup or DKG protocol satisfying Definition 6 can be substituted. Our focus is the $O(1)$ online PVSS phase rather than setup optimization. Alternative DKG constructions [47,48,49,45] may improve setup efficiency but do not affect online complexity.

C.1 The Setup Implementation

Figure 11 gives one simple way to realize the setup assumption using Feldman VSS [39] (refer to Appendix B). This realization assumes a synchronous network, authenticated channels from committee members to parties, a public bulletin board for commitments and complaints, fewer than t corrupted parties, and at least one honest committee contribution. The n parties do not communicate directly with one another during this setup.

Functionality Committee-Based Setup Realization

$(\mathsf{pp}, (\{\mathsf{sk}_i, \mathsf{pk}_i\}) \leftarrow \mathsf{CommitteeSetup}(\lambda, t, n)$

1. Each committee member i selects a random secret $k_i \in \mathbb{Z}_p$ and initiates a Feldman VSS sharing algorithm $\mathsf{VSS.Share}(k_i, n, t)$ to distribute shares $\{k_{i,j}\}_{j=1}^{n}$ to the n parties. Each share is signed with member i's signing key to prevent malicious complaints. Note that the commitment g^{k_i} is also published if the dealer is honest.
2. After all N committee members complete their sharing phases, all honest parties validate the correctness of the received shares via $\mathsf{VSS.Verify}$. If the verification fails, the receiving party broadcasts a complaint. Let $I \subseteq [1, N]$ denote the set of accepted committee contributions, i.e., those against which no valid complaint is recorded in the public transcript.
3. The master public key pk is computed as the product of all individual commitments $\mathsf{pk} = \prod_{i \in I} g^{k_i}$.
4. Each party P_j calculates its private key share as $\mathsf{sk}_j = \sum_{i \in I} k_{i,j}$ and computes the corresponding public key share as $\mathsf{pk}_j = g^{\mathsf{sk}_j}$.
5. Optionally, the master secret key can be calculated with at least t partial secret keys $\{\mathsf{sk}_j\}$ using Lagrange interpolation algorithm, i.e., $\mathsf{sk} = \mathsf{VSS.Reconstruct}((\{(j, \mathsf{sk}_j)\}_{j \in Q})$, where $Q \subseteq [1, n]$ and $|Q| \geq t$.
6. Denote $\mathsf{pp} = (g, t, n, \mathsf{pk})$ as public parameters.

Fig. 11. Committee-based realization of the setup assumption

C.2 Setup Guarantees Used by Instantiation I

The online Threshold ElGamal construction only uses the guarantees stated in Definition 6. In the realization of Figure 11, these guarantees follow from Feldman VSS, authenticated complaints, and the linearity of Lagrange interpolation under the stated setup assumptions.

Correctness. The master key $\mathsf{sk} = \sum_{i \in I} k_i$ is uniquely defined by the accepted contributions, every qualified set of party shares interpolates to the same sk, and the public key satisfies $\mathsf{pk} = g^{\mathsf{sk}}$.

Secrecy. If at least one accepted committee contribution contains honest entropy, then sk is uniform. Under the discrete-log hardness assumption, Feldman commitments do not reveal the underlying field shares, so coalitions of $< t$ parties cannot recover sk.

Public Auditability. Feldman commitments enable public consistency checks for pk and the party public key shares; signed shares and public complaints provide an auditable setup transcript via the bulletin board.

C.3 Security Properties

The following checks justify that the realization satisfies Definition 6 under the stated setup assumptions.

- **Correctness (C1)**: The master secret key sk is uniquely defined by all accepted committee contributions.
  Proof : Each committee member employs Feldman VSS to distribute shares, enabling every shareholder to verify the validity of their received share using the provided public commitments. Under the synchronous-network and bulletin-board assumptions, a termination timeout allows parties to submit complaints against malformed shares. The accepted contribution set I is therefore determined by the public transcript. For each accepted contribution $i \in I$, let $f_i(x)$ be the Feldman-valid sharing polynomial with $f_i(0) = k_i$ and $f_i(j) = k_{i,j}$. For any $Q \subseteq [1, n]$ with $|Q| \geq t$,

$$\begin{aligned}\mathsf{Intpl}(\{(j, \mathsf{sk}_j)\}_{j\in Q}) &= \mathsf{Intpl}(\{(j, \sum_{i\in I} k_{i,j})\}_{j\in Q}) \\ &= \sum_{i\in I} \mathsf{Intpl}(\{(j, k_{i,j})\}_{j\in Q}) = \sum_{i\in I} f_i(0) = \sum_{i\in I} k_i.\end{aligned}$$

  □
- **Correctness (C2)**: The master public key pk equals g^{sk}, where sk is the master secret key.
  Proof : As proved in Correctness (C1), $\mathsf{sk} = \sum_{i\in I} k_i$, then $\mathsf{pk} = \prod_{i\in I} g^{k_i} = g^{\sum_{i\in I} k_i} = g^{\mathsf{sk}}$ □
- **Correctness (C3)**: The master private key sk is uniformly distributed in $\mathbb{Z}_p$ and the master public key is uniformly distributed in $\mathbb{G}$, assuming at least one accepted contribution contains honest entropy.
  Proof : If at least one accepted committee member $i^\star \in I$ selects $k_{i^\star}$ uniformly at random during setup, then the sum $\mathsf{sk} = \sum_{i\in I} k_i$ is uniformly distributed over $\mathbb{Z}_p$ even conditioned on all other contributions. Further, the public key $\mathsf{pk} = g^{\mathsf{sk}}$ is uniformly distributed over the group $\mathbb{G}$. □
- **Secrecy**: The secret key sk must remain hidden from any adversary that controls fewer than t parties, provided that the setup transcript includes at least one honest contribution.
  Proof : Correctness (C3) establishes that sk is uniformly distributed over

its domain under the accepted honest-entropy assumption. Any coalition of at most $t-1$ parties obtains fewer than t field shares of each accepted contribution and therefore cannot interpolate any honest k_i. The public Feldman commitments reveal only group encodings of the polynomial coefficients and shares; under the discrete-log hardness assumption in $\mathbb{G}$, these encodings do not reveal the underlying field values. Hence such a coalition cannot recover sk. □

– **Robustness**: The corresponding secret key sk can be reconstructed by any subset of at least t sub keys $\{\mathsf{sk}_j\}$.
Proof : Let $Q \subseteq [1, n]$ be any set with $|Q| \geq t$, and let $L_j(0)$ be the corresponding Lagrange coefficient for point 0. Since $\mathsf{sk}_j = \sum_{i \in I} k_{i,j}$,

$$\sum_{j \in Q} L_j(0)\mathsf{sk}_j = \sum_{j \in Q} L_j(0) \sum_{i \in I} k_{i,j} = \sum_{i \in I} \sum_{j \in Q} L_j(0) k_{i,j} = \sum_{i \in I} k_i = \mathsf{sk}.$$

□

– **Public Auditability**: All commitments and complaints are published via a bulletin board, allowing any external party to audit the setup transcript without learning private shares.
Proof : VSS enables the n parties to verify their respective shares against the public commitments. A committee contribution is discarded if a valid complaint is raised. To prevent malicious complaints, we require that all shares distributed to parties are properly signed by the committee member. Consequently, any complaint can be publicly checked by verifying the member's signature on the disputed share. Let $Com_{i,r} = g^{a_{i,r}}$ be the published Feldman commitments for accepted contribution i, where $Com_{i,0} = g^{k_i}$. Any external verifier can check the master public key and each published party public key by recomputing $\mathsf{pk} \stackrel{?}{=} \prod_{i \in I} Com_{i,0}, \mathsf{pk}_j \stackrel{?}{=} \prod_{i \in I} \prod_{r=0}^{t-1} Com_{i,r}^{j^r} = g^{\sum_{i \in I} k_{i,j}}$. Thus the bulletin-board transcript determines the accepted contributions, the master public key, and all accepted public key shares, providing the public auditability required by Definition 6. □

C.4 Epoch-Based Reconfiguration

We do not claim fully adaptive dynamic-membership security. Instead, reconfiguration is modeled by epochs. In each epoch e, there is a public setup transcript st_e, a party set P_e, threshold t_e, public parameters pp_e, and key shares $\{\mathsf{sk}_{i,e}, \mathsf{pk}_{i,e}\}_{i \in P_e}$. All PVSS instances created in epoch e are verified and reconstructed only with respect to st_e. The security theorems apply independently inside each epoch, assuming st_e satisfies Definition 6.

A transition from epoch e to $e+1$ must publish a new setup transcript st_{e+1} that satisfies the same public-auditability and threshold-secrecy requirements for the new party set P_{e+1}. This can be done either by running a fresh setup ceremony or by a publicly auditable resharing procedure. In a fresh setup, the new epoch generates an independent master public key pk_{e+1}; old PVSS instances remain tied to pk_e, and new PVSS instances use pk_{e+1}. In contrast, a

proactive resharing procedure may preserve the same master secret and public key while changing only the sharing of that secret. In that case, decryption capability is deliberately migrated to the new party set, so old ciphertexts are not cryptographically isolated from the new committee. Long-lived secrecy across such resharing steps requires erasure of obsolete key material and the assumption that fewer than the relevant threshold number of shares are exposed in each epoch.

For the committee-based realization, a standard proactive resharing step can instantiate the second type of transition. Let $J \subseteq P_e$ denote an authorized set of at least t_e existing parties, and let $L = P_{e+1}$ be the new group with $|L| = n'$. The resharing transcript is accepted as the next epoch setup only if it is publicly auditable.

1. Each party $P_j \in J$ runs the Feldman VSS algorithm $\mathsf{VSS.Share}(\mathsf{sk}_{j,e}, n', t_{e+1})$, generating shares $\{\mathsf{sk}_{j,l}\}_{l\in[1,n']}$ for each new party $P_l \in L$.
2. The original setup transcript provides a public commitment to each accepted share $\mathsf{sk}_{j,e}$; in the Feldman realization this commitment is obtained by evaluating the accepted coefficient commitments at index j. Let this commitment be $\mathsf{pk}_{j,e} = g^{\mathsf{sk}_{j,e}}$. In the resharing transcript, party P_j publishes Feldman commitments $A_{j,r} = g^{a_{j,r}}$ to a polynomial $\phi_j(X) = \mathsf{sk}_{j,e} + \sum_{r=1}^{t_{e+1}-1} a_{j,r}X^r$, $A_{j,0} = \mathsf{pk}_{j,e}$. This equality proves that the resharing polynomial has the committed old share as its constant term. Furthermore, each party $P_l \in L$ invokes $\mathsf{VSS.Verify}$ on its received share $\mathsf{sk}_{j,l} = \phi_j(l)$ by checking $g^{\mathsf{sk}_{j,l}} \stackrel{?}{=} \prod_{r=0}^{t_{e+1}-1} A_{j,r}^{l^r}$. Hence every accepted $\mathsf{sk}_{j,l}$ is consistent with the publicly committed polynomial of degree at most $t_{e+1}-1$ and constant term $\mathsf{sk}_{j,e}$.
3. Suppose that the authorized resharing set J successfully completes the sharing process and no valid complaints are raised.
4. Then each party P_l in L computes its epoch-$(e+1)$ secret key as $\mathsf{sk}_{l,e+1} = \mathsf{Intpl}(\{(j, \mathsf{sk}_{j,l})\}_{j\in J})$. Equivalently, if λ_j^J denotes the Lagrange coefficient for reconstructing point 0 from the old authorized set J, then $\mathsf{sk}_{l,e+1} = \sum_{j\in J} \lambda_j^J \mathsf{sk}_{j,l} = \sum_{j\in J} \lambda_j^J \phi_j(l)$. Define $\Phi(X) = \sum_{j\in J} \lambda_j^J \phi_j(X)$. Then $\mathsf{sk}_{l,e+1} = \Phi(l)$ and $\Phi(0) = \sum_{j\in J} \lambda_j^J \phi_j(0) = \sum_{j\in J} \lambda_j^J \mathsf{sk}_{j,e} = \mathsf{Intpl}(\{(j, \mathsf{sk}_{j,e})\}_{j\in J}) = \mathsf{sk}_e$. Thus $\{\mathsf{sk}_{l,e+1}\}_{l\in L}$ is a degree-$(t_{e+1}-1)$ sharing of the same master secret, and every qualified set $Q' \subseteq L$ with $|Q'| \geq t_{e+1}$ satisfies $\mathsf{Intpl}(\{(l, \mathsf{sk}_{l,e+1})\}_{l\in Q'}) = \Phi(0) = \mathsf{sk}_e$. The corresponding public key share $\mathsf{pk}_{l,e+1} = g^{\mathsf{sk}_{l,e+1}}$ is derived and checked from the resharing commitments using the same Lagrange coefficients.

D Power-of-Tau CRS Procedures

The silent-setup instantiation uses a publicly verifiable Power-of-Tau ceremony to obtain powers of an unknown trapdoor τ. This appendix records the abstract CRS update and verification procedures used by Section 5.1; they are standard setup machinery and are not claimed as a separate contribution. Algorithm 1

abstracts a CRS update using a private contribution $\beta \in \mathbb{Z}_p$, while Algorithm 2 gives the public consistency check for the updated CRS.

Algorithm 1 *PowerOfTau*(CRS, β)

Parse CRS as $\{g_1^{\alpha^i}, g_2^{\alpha^i}\}_{i\in[1,n]}$
for $i \in [1,n]$ **do**
    $g_1^{\tau^i} \leftarrow (g_1^{\alpha^i})^{\beta^i}$, $g_2^{\tau^i} \leftarrow (g_2^{\alpha^i})^{\beta^i}$
end for
Destroy β
return $\{g_1^{\tau^i}, g_2^{\tau^i}\}_{i\in[1,n]}$

Algorithm 2 *VerifyCRS*(CRS)

Parse CRS as $\{g_1^{\tau^i}, g_2^{\tau^i}\}_{i\in[1,n]}$
for $i \in [1,n]$ **do**
    if $e(g_1, g_2^{\tau^i}) \neq e(g_1^{\tau^i}, g_2)$ **then**
        return false
    end if
end for
return true

E CPA-Secure Silent Setup Threshold Encryption [9]

Definition 7 (Evaluation of Polynomial on $\mathbb{G}$). *Let $\tau \xleftarrow{\$} \mathbb{Z}_p$ be the unknown secret trapdoor. Given a polynomial $f(x) = \sum_{i=0}^{d} a_i x^i$, the evaluation of polynomial on $\mathbb{G}$ at τ is defined and calculated as: $g^{f(\tau)} := g^{\sum_{i=0}^{d} a_i \tau^i} = \prod_{i=0}^{d} \left(g^{\tau^i}\right)^{a_i}$, where g^{τ^i} are publicly known. Apparently, this evaluation requires $d+1$ exponentiations.*

Definition 8 (Vanishing Polynomial). *Let $\omega \in \mathbb{Z}_p$ be an ℓ-th primitive root of unity, i.e. $\omega^\ell = 1$ over $\mathbb{Z}_p$. ω generates the multiplicative subgroup of roots of unity $\mathbb{H} = \{\omega, \omega^2, \ldots, \omega^\ell\}$. Let $Z(x) = \prod_{i=1}^{\ell}(x - \omega^i)$ be the vanishing polynomial on $\mathbb{H}$. Since $\mathbb{H}$ is a multiplicative subgroup, $Z(x) = x^\ell - 1$ and $L_i(x) = \frac{\omega^i}{\ell} \cdot \frac{x^\ell - 1}{x - \omega^i}$. Note that $L_i(0) = \frac{1}{\ell}$. Sometimes, we will refer to ω^ℓ as ω_0 and $L_0 = L_\ell$, as $\mathbb{H}$ is cyclic.*

Lemma 8 (Univariate Sumcheck[50]). *Let $A(x) = \sum_{i=0}^{n} a_i L_i(x)$ and $B(x) = \sum_{i=0}^{n} b_i L_i(x)$, Then:*

$$A(x)B(x) = \frac{\sum_i a_i b_i}{n} + Q_z(x)Z(x) + Q_x(x)x$$

holds, where $Q_x(x) = \sum_{i=0}^{n} a_i b_i \frac{L_i(x)-L_i(0)}{x}$ *and* $Q_Z(x) = \sum_{i=0}^{n} a_i b_i \frac{L_i^2(x)-L_i(x)}{Z(x)}$ $+ \sum_{j=1,j\neq i}^{n} a_i b_j \frac{L_i(x)L_j(x)}{Z(x)}$.

Figure 12 demonstrates the CPA-secure STE concrete construction. The construction builds on a univariate sumcheck protocol (Lemma 8) and a silent setup threshold signature (STS) scheme. It has been shown that a STE scheme can be constructed from any signature scheme whose verification process forms a public linear constraint system. Consequently, the core of the STE scheme lies in finding a matrix $\mathbf{A}$ that encapsulates the verification logic of a threshold signature. The encryption key in the STE scheme corresponds to the verification key of STS, while decryption involves aggregating a valid signature that satisfies STS verification. Specifically, the matrix $\mathbf{A}$ in STE encryption serves as the STS verification key, and the decryption vector $\mathbf{w}$ represents a valid aggregated STS signature, augmented with elements derived from the univariate sumcheck protocol used during signature verification.

Though, [9] presents a method to achieve CCA-security: a tag-based NIZK proof which proves knowledge of γ. However, this method does not guarantee well-formed ciphertext, thus unsuitable for our CCATE scheme.

Functionality CPA-Secure Threshold Encryption via Silent Setup

$\underline{(\mathsf{pp}, \{\mathsf{sk}_i, \mathsf{pk}_i\}) \leftarrow \mathtt{Setup}(\lambda, t, n)}$

As introduced in Figure 4

$\underline{bool \leftarrow \mathtt{VerifySetup}(\mathsf{pp}, \{\mathsf{pk}_i\})}$

As introduced in Figure 4

$\underline{(C) \leftarrow \mathtt{Encrypt}(s, \mathsf{pp})}$

Sample $\gamma \xleftarrow{\$} \mathbb{Z}_p$, parse Z_0 from pp, set $\mathbf{b} = (1_{\mathbb{G}_T}, 1_{\mathbb{G}_T}, 1_{\mathbb{G}_T}, 1_{\mathbb{G}_T}, g_t)^\top$, and set $\mathbf{A}$ (the same as in Figure 5)

Sample $s_1, \cdots, s_5 \xleftarrow{\$} \mathbb{Z}_p^*$ and set $\mathbf{v} = (s_1, \cdots, s_5)^\top$

$C = (g_2^\gamma, \mathbf{A}^\mathbf{v}, \mathbf{b}^\mathbf{v} \cdot g_t'^s)$ ($\mathbf{A}^\mathbf{v}$,$\mathbf{b}^\mathbf{v}$ denotes matrix scalar exponentiation)

$\underline{D_i \leftarrow \mathsf{CPA.PartialDec}(C, \mathsf{sk}_i)}$

$D_i = C[0]^{\mathsf{sk}_i} = g_2^{\gamma \mathsf{sk}_i}$

$\underline{bool \leftarrow \mathtt{VerifyPD}(D_i, \mathsf{pk}_i, C)}$

$e(\mathsf{pk}_i, g_2^\gamma) \stackrel{?}{=} e(g_1, D_i)$

$\underline{S \leftarrow \mathtt{Decrypt}(C, \{(i, D_i)\}_{i \in P})}$

- Using ak to compute the subset with valid public keys in P, i.e., $P_I = P \cap I$. Interpolate a polynomial $B(X)$ such that $B(\omega^0) = 1, B(\omega^i) = 0$ for $i \notin P_I$ and set $B = g_2^{\sum_{i=0}^n B(\omega^i) L_i(\tau)}$.
- Compute $\mathsf{pk}' = (g_1 \prod_{i \in P_I} \mathsf{pk}_i^{B(\omega^i)})^{\frac{1}{n+1}}$ as aggregated public key.
- Compute polynomial $Q_x(X)$ and $Q_z(X)$ such that $\mathsf{SK}(X)B(X) = \frac{\mathsf{sk}'}{n+1} + Q_z(X)Z(X) + Q_x(X)X$ where $\mathsf{SK}(X) = \sum_{i=1}^n \mathsf{sk}_i L_i(X)$ and $\mathsf{sk}' = \sum_{i=0}^n \mathsf{SK}(\omega^i)B(\omega^i) = \sum_{i \in P_I} \mathsf{sk}_i B(\omega^i) + 1$. According to Lemma 8 univariate sumcheck: $Q_x(X) = \sum_{i=0}^n B(\omega^i)(\mathsf{sk}_i \frac{L_i(X) - L_i(0)}{X})$, $Q_Z(X) = \sum_{i=0}^n B(\omega^i)(\mathsf{sk}_i \frac{L_i^2(X) - L_i(X)}{Z(X)}) + \sum_{i=0}^n B(\omega^i) \sum_{j=0, j \neq i}^n (\mathsf{sk}_j \frac{L_i(X) L_j(X)}{Z(X)})$. Compute $Q_Z = g_1^{Q_Z(\tau)}$.
- Compute $Q(x) = g_1^{Q_x(\tau)}$ and $\hat{Q}(x) = g_1^{Q_x(\tau) \cdot \tau}$.
- Set $D = \frac{g_2^\gamma \prod_{i \in P_I} D_i^{B(\omega^i)}}{n+1}$ as aggregated partial decryption.
- Compute $\hat{B} = g_1^{\tau^t B(\tau)}$.
- Compute $Q_0(X)$ such that $B(X) - 1 = Q_0(X)(X - \omega^0)$ and its commitment $Q_0 = g_1^{Q_0(\tau)}$.
- $\mathbf{w}$ is set as $(B, -\mathsf{pk}', -Q_Z, -Q_x, \hat{Q}_x, D, -\hat{B}, -Q_0)$.
- Output $S = C[2] \cdot e(C[1], \mathbf{w})^{-1} = \mathbf{b}^\mathbf{v} \cdot g_t'^s \cdot e(\mathbf{A}^\mathbf{v}, \mathbf{w})^{-1} = \mathbf{b}^\mathbf{v} \cdot g_t'^s \cdot e(\mathbf{A}, \mathbf{w})^{-\mathbf{v}} = g_t'^s$.

Fig. 12. CPA-Secure Threshold Encryption via Silent Setup