Accountable Verifiable Secret Sharing and Applications

Karim Baghery[1], Ehsan Ebrahimi[2],
Omid Mirzamohammadi[1], and Mahdi Sedaghat[1,3]

[1] COSIC, KU Leuven, Leuven, Belgium
karim.baghery@kuleuven.be, omid.mirzamohammadi@esat.kuleuven.be,
ssedagha@esat.kuleuven.be
[2] University of Luxembourg, Luxembourg
eebrahimi.pqc@gmail.com
[3] Mysten Labs, Palo Alto, US

Abstract. Verifiable Secret Sharing (VSS) ensures consistent sharing and correct reconstruction, but does not identify the shareholders involved. Traceable Secret Sharing (TSS) expands secret sharing with two guarantees: *traceability*, which identifies the makers of a reconstruction box through black-box queries; and *non-imputability*, which prevents false accusations against honest shareholders. However, full traceability is practically expensive: existing Shamir-based schemes require multiple carefully chosen queries to a stateless reconstruction box. Moreover, Goyal et al. (ePrint 2025/1980) show that the strongest form of black-box tracing is incompatible with simultaneously achieving non-imputability and tracer secrecy. We introduce *Accountable VSS* (Acc-VSS), a practical extension of VSS that achieves the next-best goal: *accountability*, a strict relaxation of traceability requiring that (i) the contributors to every accepted reconstruction be publicly identifiable from its transcript with a *single* query, and (ii) no honest shareholder be falsely accused, even by an adversary holding all other shares. Accountability thus replaces black-box, multi-query tracing with a transcript check executable by any external verifier or smart-contract judge. In this paper, (i) we give a generic transformation from any computationally secure Shamir-based VSS to Acc-VSS, adding only one field element per share; it uses a long-term identity secret to each share and a conjunctive NIZK to bind partial reconstructions to the shareholder and the dealer's commitments; (ii) we instantiate it with Feldman and Pedersen VSS and prove the security of both schemes; (iii) we prove a self-contained single-query impossibility lemma motivating our relaxation; and (iv) we show how Acc-VSS uniformly yields accountable threshold ElGamal decryption, Schnorr/FROST signing, and distributed key generation. For the benchmarked VSS-based threshold-decryption protocol, accountability adds at most 20.3% end-to-end overhead for $n = 20$ and less than 6% for $n = 100$. Encrypted mempools, end-to-end-verifiable e-voting tallies, and validator-slashing in proof-of-stake systems all reduce to this framework.

Keywords: Verifiable Secret Sharing · Accountability · Threshold Cryptography · Distributed Key Generation · Shamir Secret Sharing

1 Introduction

Verifiable secret sharing (VSS) is one of the widely deployed primitives of modern cryptography. A VSS scheme [2, 3, 8, 21, 34] lets a dealer split a secret s among n shareholders so that any qualified subset of $t+1$ shareholders can reconstruct s, no smaller subset learns anything about it, and a malicious dealer cannot covertly distribute inconsistent shares. VSS is the building block for distributed key generation (DKG) [1, 24, 34], threshold signing [11, 32], threshold decryption [12–14, 37], encrypted-mempool designs [23, 38], end-to-end-verifiable e-voting, and validator-managed key custody in proof-of-stake systems. Throughout this paper we focus on Shamir-based VSS.

From traceable VSS to accountable VSS. *Traceable secret sharing* (TSS) [6, 30] targets two complementary protections: *traceability*, that a coalition publishing a reconstruction box can be traced back to at least one corrupt party from black-box access to that box (a *reconstruction box* is an algorithm, possibly stateful, that the coalition builds from its own shares; it takes as input one or more shares (i, s_i) of other parties and outputs the secret s, or $\perp$ on failure); and *non-imputability*, that an honest shareholder cannot be falsely accused. The two are usually presented together because traceability without non-imputability is dangerous (it would let a malicious tracer frame an honest party), and non-imputability without traceability would look like a retreat from the original goal of catching corrupt parties.

In practice, full traceability is expensive and impractical. Existing tracing strategies [6, 30] query a stateless reconstruction box *multiple times* with carefully crafted inputs, often with inputs that a verifiable scheme would reject (e.g., invalid shares). The Shamir-based tracing construction of Boneh, Partap, and Rotem [6] requires $2t$ black-box queries in the worst case to obtain the t evaluations needed for tracing, where t is the degree of the sharing polynomial. Concrete adversaries refresh the box's state between queries or simply self-destruct after one use, in which case the multi-query model collapses. And, on the theory side, recent work of Goyal et al. [29] proved that the strongest form of black-box tracing in the over-authorised regime is incompatible with both non-imputability and tracer secrecy unless one assumes indistinguishability Obfuscation (iO). The single-query setting, in which the tracer may query the box only once, as opposed to the multi-query tracers of [6, 30], is even harder: there are $\binom{n-1}{t}$ possible coalitions of t corrupt parties, each of which can present a stateless reconstruction box, and all of these boxes look identical to a single-query tracer. Hence single-query tracing of arbitrary boxes is impossible (Lemma 2).

What *is* achievable, and what real applications actually need, is one step weaker. Every threshold protocol that a deployment cares about, a tally, a partial decryption, a signature share, is consumed by some external verifier (a smart-contract judge, an e-voting auditor, a slashing module). That verifier always has the full audit transcript in front of it. The accountability property that matters in practice is: *from a single inspection of that transcript, can the verifier publicly read off the set of shareholders who contributed, and is the answer safe*

against framing? We call this *accountability*, and the corresponding primitive *Accountable VSS* (Acc-VSS). Accountability is a strict relaxation of traceability, it does not catch reconstruction boxes built outside the protocol, but it is exactly the property that the application layer enforces.

Accountable VSS (Acc-VSS). We develop Acc-VSS as a natural and practical extension of VSS. An Acc-VSS is a VSS scheme that, in addition to standard correctness, secrecy, and verifiability, satisfies a single *accountability* property with two clauses:

(A1) *Single-query attribution.* For every reconstruction transcript T that the protocol-bound application accepts, a public algorithm $\mathsf{VerifAudit}(T)$ returns the unique set I of shareholders who contributed to T, read off in a single query, with no interaction with any reconstruction box.

(A2) *Non-imputability.* No PPT adversary given the shares of all but one shareholder can produce a transcript T such that $\mathsf{VerifAudit}(T)$ implicates the remaining honest shareholder.

Accountability subsumes non-imputability, (A2) is the framing-protection clause inside the same definition that (A1) gives attribution, and replaces the multi-query, black-box tracing of TSS with a transcript-level check. This is what makes Acc-VSS natural for deployment: no trusted tracer, no per-query interaction, just an audit transcript that any external verifier or smart-contract judge can inspect.

Threat model: what accountability buys. A shareholder is *corrupt* if it deviates from the policy of the application that consumes the reconstruction, typically by colluding with other shareholders. Such deviations need not involve malformed messages. VSS verifiability already rejects *malformed* contributions (an inconsistent share, a wrong partial decryption). It cannot flag a *well-formed* contribution to an operation the policy forbids: decrypting a mempool transaction before its block is final, signing a message the validator set never approved, or opening a tally before the voting phase closes. Such contributions pass every check, and this is exactly the gap accountability closes. (A1) records, verifiably, which shareholders contributed to every transcript the application accepts. Whether a contribution is punishable is decided by the application, e.g. a slashing rule that compares the transcript with the protocol schedule. A shareholder that contributes in good faith to a request of a corrupt coalition is recorded as a contributor. Accountability does not decide intent; it provides the evidence. (A2) guarantees that a shareholder who did *not* contribute is never named, even if all other shareholders and the dealer collude.

Why Shamir-based VSS cannot be non-imputable on its own. In Shamir secret sharing and its verifiable variants, the indices $1, 2, \ldots, n$ used to evaluate the secret polynomial are public. Any qualified coalition of $t + 1$ shareholders can therefore compute any other party's share by Lagrange interpolation and produce a reconstruction contribution—partial decryption, signature share, or plain share reveal—that is indistinguishable from one the honest party would have produced.

Our core idea. Each shareholder P_i holds, alongside its dealer-distributed share $f_i = f(i)$, where f is the dealer's secret degree-t sharing polynomial with $f(0) = s$, a long-term identity secret γ_i whose public commitment $\mathsf{vid}_i = g^{\gamma_i}$ is registered once and reused across all of P_i's threshold deployments. Every reconstruction contribution comes with a single Non-Interactive Zero-Knowledge (NIZK) that simultaneously proves correctness of the partial output, knowledge of the dealer-side share consistent with the dealer's commitments, and knowledge of the identity secret behind vid_i. The NIZK conjunction is what makes the transcript both *attributable* (anyone can read off who signed) and *non-imputable* (the adversary cannot forge a contribution on behalf of a party whose γ_i it does not hold).

Isn't this just VSS plus a signature on each contribution? A natural alternative is to run an actively secure VSS and have each shareholder sign every partial output under a registered long-term key. We do not claim that this is insecure. Rather, it is one instance of our generic transformation: a Schnorr signature on the (session-bound) contribution is a signature of knowledge of the long-term secret γ_i behind vid_i, so its security for attribution and non-imputability follows from our generic theorem (Theorem 1). What Acc-VSS adds is (i) a precise definition of what such a scheme must guarantee (clauses (A1) and (A2) below), which prior threshold protocols provide only implicitly; (ii) a single proof, for any VSS and any knowledge-sound binding, that these guarantees hold; (iii) the observation (Lemma 1) that the binding secret must be independent of the dealer's polynomial; and (iv) instantiations in which share correctness and identity are proved in one conjunctive Σ-protocol with a shared challenge. We make this comparison explicit in Appendix C.5.

Our Contributions.

1. Accountable VSS as a primitive. We define Acc-VSS in Section 3. The new property is *accountability* (Def. 6), defined as the conjunction (A1)+(A2) above. We note that clause (A1) (single-query attribution) is already implicit in any actively-secure threshold protocol whose per-party proofs bind outputs to public keys; the novel clause is (A2) (non-imputability), which the interpolation attack of Lemma 1 shows cannot be achieved natively in any Shamir-based scheme. We also fix an algorithm vs. protocol confusion present in earlier TSS formulations [6, 30]: Π_{Share} and Π_{Verify} are interactive protocols among the parties; Reconstruct, VerifAudit, and Judge are publicly-runnable algorithms.

2. A generic VSS-to-Acc-VSS transformation, with proof. We give a black-box transformation that lifts any computationally secure Shamir-based VSS in the honest-majority setting into an Acc-VSS with one extra field element per share, and prove it secure (Theorem 1).

3. Two impossibility lemmas motivating the design. Lemma 1 establishes the load-bearing lower bound: in any Shamir-based scheme whose per-party con-

tribution depends only on the dealer-distributed share and the public commitments, any qualified coalition can frame any other party by interpolation. This is precisely why γ_i must be sampled independently of the dealer's sharing polynomial f. Lemma 2 records the complementary counting argument for single-query box tracing: each of the $\binom{n-1}{t}$ possible coalitions of size t can present a stateless box, and these boxes are indistinguishable to a single-query tracer, justifying why Acc-VSS targets accepted transcripts rather than arbitrary reconstruction boxes.

4. Concrete instantiations on Feldman and Pedersen VSS. Section 4 instantiates the transformation on Feldman VSS with the long-term identity binding; Appendix B does the same on Pedersen VSS. The Feldman variant gives unpredictability under DL; the Pedersen variant gives statistical secrecy. Each adds one extra field element per share over its underlying VSS.

5. A uniform recipe for accountable threshold cryptography. Section 5 uses Acc-VSS as the building block for three concrete protocols. Accountable threshold ElGamal decryption is fully proved (Theorem 4). Accountable threshold Schnorr/FROST signing and accountable distributed key generation are presented as sketches (see Appendices C.8 and C.9); full proofs follow the same structure and are deferred. The reconstruction-phase accountability overhead is constant: two extra scalar multiplications per party ($\approx 40\,\mu$s, or 30–37% over the reconstruction phase alone). The end-to-end overhead, which includes the identical Share and Verify phases, drops from 20.3% at $n = 20$ to under 6% at $n = 100$.

Scope. Acc-VSS adds accountability (single-query, transcript-level attribution plus non-imputability) to VSS, at the cost of one extra field element per share and one NIZK conjunction per reconstruction contribution. It does *not* catch arbitrary reconstruction boxes built outside the protocol, nor prevent a qualified subset from reconstructing s and using it outside any application; both would require either the multi-query box-tracing notion of [6,30] or the iO-based notion of [29], and Lemma 2 formalises why single-query tracing of arbitrary boxes is unattainable without such assumptions. Table 1 positions Acc-VSS relative to the relevant prior work; Appendix D expands on each comparison point. We discuss secret sharing with snitching [17] separately, as it considers a complementary form of accountability in the individual-cryptography model. Although both approaches require only a single reconstruction execution, their guarantees differ: snitching relies on a participating entity to reveal evidence enabling the punishment of another participant, whereas Acc-VSS directly and publicly attributes every valid contribution appearing in an accepted reconstruction transcript, while additionally providing VSS and non-imputability guarantees.

Outline. Section 2 reviews the required preliminaries. Section 3 introduces Acc-VSS, presents the impossibility results motivating its design, formalises its syntax and security properties, and gives the generic VSS-to-Acc-VSS transformation.

Table 1. Comparison of Acc-VSS with prior secret-sharing and tracing primitives. "Ver.": Verifiable, "Recon.": Reconstruction. "Secret reconstruction" is BB if attribution requires black-box access to a reconstruction box, and nBB otherwise. "q" is the number of reconstruction queries required to recover an attribution. "Share size" gives the private share size, where $|s|$ denotes one field element. "Non-imp." is non-imputability against an adversary holding the shares of $n-1$ shareholders. "Pub. trace" marks publicly verifiable attribution with no trusted tracer. Row $\textsc{Imp}_{1,\text{BB}}$ is the impossibility result: BB reconstruction with a single query is unachievable without either a trusted third party or binding each share to the party's identity, as Acc-VSS does.

Scheme	Ver.	Recon.	q	Share size	Non-imp.	Pub. trace
SSS [36]	✗	nBB	1	$\|s\|$	✗	n/a
VSS [21, 34]	✓	nBB	1	$\|s\|$ / $2\|s\|$	✗	n/a
TSS [30]	✗	BB	$q > t$	$\Theta(\|s\|^2)$	✓	✗
TSS [6]	✗	BB	$q > t$	$2\|s\|$	✓	✗
TSS [29] (iO)	✗	BB	$q > t$	$\mathsf{poly}(n)(\|s\|)$	partial†	✗
$\textsc{Imp}_{1,\text{BB}}$ (Lem. 2)	–	BB	1	n/a	✗‡	–
Acc-VSS (§3–4)	✓	nBB	1	$2\|s\|$ / $3\|s\|$	✓	✓

For the VSS and Acc-VSS rows, the two share sizes correspond respectively to the Feldman- and Pedersen-based variants. † Goyal et al. show that strong (over-authorised) tracing is incompatible with the conjunction of non-imputability and tracer secrecy [29, Thm. 1.2]; their construction relies on indistinguishability obfuscation. ‡ Impossible: each of the $\binom{n-1}{t}$ possible coalitions of t corrupt parties can present a stateless reconstruction box, and these boxes are identical from a single-query tracer's view (Lemma 2). Acc-VSS escapes the impossibility by binding each share to a per-party identity secret γ_i.

Section 4 instantiates the transformation with Feldman VSS, while the Pedersen-based construction is deferred to Appendix B. Section 5 applies Acc-VSS to accountable threshold cryptography, including threshold ElGamal decryption, distributed key generation, and Schnorr/FROST signing. Finally, Section 6 concludes and discusses directions for future work; supporting definitions, proofs, comparisons, and implementation results appear in the appendices.

2 Preliminaries

Notation. We let λ denotes a security parameter and 1^λ its unary representation. A function that is negligible in the security parameter λ is simply called negligible. We use the assignment operator $\leftarrow$ to denote uniform sampling from a set Ξ, e.g. $x \leftarrow \Xi$. We write $\mathbb{Z}_N := \mathbb{Z}/N\mathbb{Z}$ and $\mathbb{Z}_N[X]_t$ for polynomials of degree t in the variable X and with coefficients in $\mathbb{Z}_N$, i.e. the polynomial ring over $\mathbb{Z}_N$. For $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. PPT stands for probabilistic polynomial time and all algorithms are PPT, unless mentioned.

Definition 1 (Discrete Logarithm Problem (DLP) [15]). *Let $\mathbb{G}$ be a cyclic group of prime order q with generator g. The DLP is hard if for all PPT adversaries $\mathcal{A}$:* $\Pr[G \leftarrow\!\$\, \mathbb{G};\ x \leftarrow \mathcal{A}(g, G) : g^x = G] \leq \mathsf{negl}(\lambda)$.

Throughout, q is a prime with $q > n$, $\mathbb{G}$ is a cyclic group of prime order q with generator g in which the DLP is hard, and all polynomials are over $\mathbb{Z}_q$.

Shamir secret sharing and Feldman VSS. In (n, t)-threshold Shamir secret sharing [36], the dealer samples a random polynomial $f(X) = f_0 + a_1 X + \cdots + a_t X^t \in \mathbb{Z}_q[X]$ with $f(0) = f_0 = s$ and gives party P_i the share $f_i = f(i)$. Any set V with $|V| \geq t + 1$ recovers $s = \sum_{i \in V} L_i f_i$ with Lagrange coefficients $L_i = \prod_{j \in V \setminus \{i\}} j/(j - i)$, while any t shares are independent of s. In Feldman VSS [21] the dealer additionally broadcasts $\pi_D = (c_0, \ldots, c_t) = (g^{f_0}, g^{a_1}, \ldots, g^{a_t})$, and P_i accepts its share iff $g^{f_i} = \prod_{j=0}^{t} c_j^{i^j}$. *Complaint resolution* is the standard procedure by which disagreements are settled publicly: a party whose share fails this check broadcasts a complaint, the dealer must broadcast that share, and the dealer is disqualified if a revealed share fails the check or more than t parties complain.

NIZK proofs of knowledge. A NIZK $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for a relation $\mathbf{R}$ is complete, zero-knowledge (proofs can be simulated without the witness), and *knowledge sound*: for every PPT prover that outputs (x, π) with $\mathsf{V}(pp, x, \pi) = 1$ there is an efficient extractor that outputs w with $(x, w) \in \mathbf{R}$ except with negligible probability. All our Σ-protocols are made non-interactive with the Fiat–Shamir transform in the random oracle model.

Full definitions and further preliminaries are given in Appendix A: Secret Sharing and Shamir's Scheme (Appendix A.1), Feldman VSS (Appendix A.2), Traceable Secret Sharing (Appendix A.3), Sigma Protocols and Non-Interactive Zero-Knowledge Proofs, the Fiat-Shamir Transform (Appendix A.4), Public-Key Cryptosystems and ElGamal's Scheme (Appendix A.5), Commitment Schemes (Appendix A.6), and Distributed Key Generation Protocols (Appendix A.7).

3 Accountable Verifiable Secret Sharing

In this section we formalise *Accountable Verifiable Secret Sharing* (Acc-VSS). We work in the standard honest-majority synchronous broadcast model: there are n shareholders $P_1, \ldots, P_n$, a (possibly malicious) dealer D, an authenticated broadcast channel, and private dealer-to-shareholder channels. The threshold is $t < n/2$. As is standard, we assume a common reference string pp for the underlying commitment and NIZK schemes; in the discrete-logarithm instantiations of Section 4, pp reduces to a prime-order group description and one or two random generators.

Section 3.1 states the impossibility lemma that motivates the relaxation from full traceability to accountability. Section 3.2 gives the syntax of Acc-VSS and clarifies the algorithm-vs.-protocol distinction (see also Appendix C.1). Section 3.3 states the security games and the generic VSS-to-Acc-VSS transformation, and Theorem 1 establishes its security.

3.1 Impossibility results motivating the design

The two lemmas below justify the two key design decisions of Acc-VSS: why γ_i must be sampled independently of f (Lemma 1), and why the accountability

goal is limited to accepted transcripts rather than arbitrary reconstruction boxes (Lemma 2).

Lemma 1 (Shamir-based VSS is inherently imputable). *Let Γ' be any (n,t)-Shamir-based VSS in which each party P_i's reconstruction contribution is a deterministic function of (f_i, π_D) alone (i.e., the share and the public dealer commitment). Then for any honest-target index $i^* \in [n]$ and any qualified coalition $Q \subseteq [n] \setminus \{i^*\}$ with $|Q| = t+1$, the coalition can produce a reconstruction contribution for P_{i^*} indistinguishable from an honest one.*

Proof. By Lagrange interpolation at i^*, the coalition computes $f_{i^*} = f(i^*) = \sum_{j \in Q} \lambda_j f_j$ exactly from its own shares $\{f_j\}_{j \in Q}$. Since π_D is public and the contribution depends only on (f_{i^*}, π_D), the coalition can produce it without any interaction with P_{i^*}. □

This is the primary motivation for the independent identity secret γ_i: γ_i is sampled by P_i itself, outside the dealer's polynomial, so it lies beyond the interpolation range of any coalition that does not hold P_i's actual secret. Section 4.1 and Appendix B exploit this by requiring a NIZK conjunction over both f_i and γ_i; clause (A1) then carries attribution weight entirely through the γ_i component. In standard actively-secure threshold protocols, per-party proofs already bind outputs to public keys, achieving single-query attribution (A1); the contribution of Acc-VSS is specifically clause (A2), which the interpolation attack shows cannot be achieved natively without an independent identity secret.

Reconstruction boxes and tracing. Following [6,30], a *reconstruction box* built by a coalition J is an efficient algorithm D_J derived from the shares $\{f_j\}_{j \in J}$. On input one or more shares (i, f_i) with $i \notin J$, it outputs a candidate secret, or $\perp$. D_J is *stateless* if its answer to a query does not depend on earlier queries. A *single-query tracer* is given oracle access to a box, submits exactly one query of its choice, and must output a party $j \in J$ (an accusation).

Lemma 2 (Single-query tracing is impossible for stateless boxes). *Let Γ' be any (n,t)-Shamir-based secret-sharing scheme with $t < n-1$. No efficient tracer with black-box single-query access to a stateless reconstruction box built by an unauthorised coalition can identify any party in the coalition.*

Proof (Proof sketch). Fix any honest-shareholder index $i^\star \in [n]$ and any subset $J \subseteq [n] \setminus \{i^\star\}$ with $|J| = t$. The coalition holding $\{f_j\}_{j \in J}$ can build a stateless reconstruction box D_J that, on input (i, f_i) with $i \notin J$, Lagrange-interpolates $\{f_j\}_{j \in J} \cup \{(i, f_i)\}$ and outputs the constant term. By correctness of Shamir, every D_J outputs the secret f_0 on the excluded share $(i^\star, f_{i^\star})$. There are $\binom{n-1}{t}$ choices of J, all yielding boxes indistinguishable to a single-query tracer: querying $(i^\star, f_{i^\star})$, every D_J returns f_0. The tracer's accusation is therefore independent of J. The single escape is $t = n-1$, where $\binom{n-1}{t} = 1$ and tracing collapses to the trivial identification of all $n-1$ corrupt parties. □

The lemma complements the iO-relative impossibility of Goyal et al. [29, Thm. 1.2] for the over-authorised regime, but is independent of it. We take

it as the formal motivation for the design that follows: Acc-VSS is built around *accountability* (single-query attribution of accepted transcripts plus non-imputability), not around the strictly stronger black-box tracing notion.

3.2 Syntax

We decompose an Acc-VSS into a tuple $\Gamma = (\mathsf{Setup}, \Pi_{\mathsf{Share}}, \Pi_{\mathsf{Verify}}, \mathsf{Verify}_i, \mathsf{Reconstruct}, \mathsf{Combine}, \mathsf{VerifAudit}, \mathsf{Judge})$, where Setup is a one-time setup algorithm, Π_{Share} and Π_{Verify} are interactive protocols (with broadcast and complaint resolution, recalled in Section 2), and Verify_i, Reconstruct, Combine, VerifAudit, Judge are publicly-runnable algorithms. A more detailed discussion of why the share-distribution and verification phases must be protocols rather than single-shot algorithms is deferred to Appendix C.1.

All secrets and shares live in $\mathbb{Z}_q$ for a prime $q > n$. We let F be an injective group homomorphism from $(\mathbb{Z}_q, +)$ to a group $\mathcal{Y}$ of order q. In this paper, F is either the identity Iden (plain reconstruction) or $F(x) = c^x$ for some $c \in \mathbb{G}$ (the function used in threshold ElGamal decryption). By linearity, any set V with $|V| \geq t+1$ computes $F(s) = \prod_{i \in V} F(f_i)^{L_i}$ (written additively for Iden). A *contribution* of P_i is the pair (z_i, π_i) output by $\mathsf{Reconstruct}_i$, where $z_i = F(f_i)$ and π_i is a proof. The contribution is *valid* if π_i verifies against ak. We write V for a set of indices of valid contributions and $|V|$ for its cardinality.

Definition 2 (Accountable Verifiable Secret Sharing). *An (n,t)-Acc-VSS for a function family F consists of the components below.*

1. $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: outputs public parameters.*
2. $\Pi_{\mathsf{Share}}(\mathsf{pp}, n, t, s)$*: an interactive protocol between D and $P_1, \ldots, P_n$. At the end of an honest execution, each P_i holds an individual share s_i, the public bulletin board contains the dealer's commitment π_D, a per-party audit commitment c'_i for every $i \in [n]$, and the public audit/verification key $\mathsf{ak} = \mathsf{vk} = (\pi_D, c'_1, \ldots, c'_n)$.*
3. $\Pi_{\mathsf{Verify}}(\mathsf{ak}, \{s_i\}_{i \in [n]})$*: an interactive protocol that detects a malicious dealer or malicious shareholder by complaint resolution, and outputs a qualified set $Q \subseteq [n]$ with $|Q| \geq t+1$, or $\perp$.*
4. *Per-share verification $\mathsf{Verify}_i(\mathsf{ak}, s_i) \to \{0,1\}$: a non-interactive algorithm that lets a single party (or external observer) check the well-formedness of an individual share against ak. The interactive Π_{Verify} resolves global disagreement; Verify_i is the local primitive on which Π_{Verify} is built.*
5. $\mathsf{Reconstruct}_i(F, \mathsf{ak}, s_i) \to (z_i, \pi_i)$*: party P_i's contribution. It outputs $z_i = F(f_i)$ together with a NIZK proof π_i that z_i was computed from the underlying f_i and that P_i knows the audit secret γ_i tied to c'_i in* ak.
6. $\mathsf{Combine}(F, \mathsf{ak}, \{(z_i, \pi_i)\}_{i \in Q}) \to (z, \pi_{\mathrm{rec}})$ *or* $(\{z_j\}_{j \in V}, \pi_{\mathrm{rec}})$ *or $\perp$: a public algorithm that filters out invalid contributions, records the bitstring* who *identifying contributors, and either reconstructs $z = F(s)$ when at least $t+1$ valid contributions are present, returns the collected partials when fewer, or*

returns $\perp$ when none are valid. The string who *is used solely for efficiency: like the signer bitmap of a multi-signature, it tells the verifier which public values (y_j, vid_j) to check each proof against. Attribution rests on the proofs π_j, not on* who.

7. $\mathsf{VerifAudit}\big(z \text{ or } \{z_j\}_{j\in V}, \pi_{\mathrm{rec}}, \mathsf{ak}\big) \to (b, I, \pi_{\mathsf{Audit}})$*: a public deterministic algorithm. It returns $b = 1$, an attribution set $I \subseteq V$, and an audit proof π_{Audit}, when the reconstruction transcript is well formed; otherwise $(0, \emptyset, 0^n)$.*
8. $\mathsf{Judge}(I, \pi_{\mathsf{Audit}}, \mathsf{vk}) \to \{0,1\}$*: a public deterministic algorithm that re-checks the audit proof against the attribution set.*

We highlight two design choices: per-share verification Verify_i is exposed as a first-class primitive, and the audit-key ak *is* the verification key vk; both are discussed in Appendix C.2.

Security properties. An Acc-VSS must satisfy the four properties below: correctness, verifiability, secrecy, and *accountability*. Throughout, we write $\mathsf{negl}(\lambda)$ for a function negligible in the security parameter λ.

Definition 3 (Correctness). *For all $n > 1$, $t < n$, all secrets s, all $Q \subseteq [n]$ with $|Q| \geq t+1$, and any honest execution of Π_{Share} followed by an honest* Combine *on the partial outputs of $\{P_i\}_{i\in Q}$, the resulting reconstruction transcript is accepted by* VerifAudit *with attribution set Q.*

Definition 4 (Verifiability). *For all $n \geq 2t+1$, all PPT $\mathcal{A}$ playing as a malicious dealer (and up to t shareholders),*

$$\Pr\left[\begin{array}{l}\mathsf{ak}, \{s_i\}_{i=1}^n \leftarrow \mathcal{A}(1^\lambda, n, t); \Pi_{\mathsf{Verify}}(\mathsf{ak}, \{s_i\}_{i=1}^n) = Q \neq \perp : \\ \exists\, s\ \forall V \subseteq Q,\ |V| \geq t+1 :\ \mathsf{Combine}(\mathsf{Iden}, \mathsf{ak}, \{(s_i, \pi_i)\}_{i\in V}) = s\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

That is: once Π_{Verify} accepts a qualified set Q, the dealer is committed to a unique secret s that will be reconstructed by every honest qualified subset of Q.

Definition 5 (Secrecy). *For all $n > 1$, $t < n$ and all PPT (or unbounded, in the statistical case) $\mathcal{A}$ controlling at most t shareholders,*

$$\Pr\left[\begin{array}{l}(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, n, t);\ b \leftarrow\!\$\ \{0,1\}; \\ \mathsf{ak}, \{s_i\}_{i=1}^n \leftarrow \Pi_{\mathsf{Share}}(\mathsf{pp}, n, t, m_b); \\ Q \subset [n],\ |Q| \leq t; b' \leftarrow \mathcal{A}(\mathsf{ak}, \{s_i\}_{i\in Q}, m_0, m_1) : b' = b\end{array}\right] \leq \tfrac{1}{2} + \mathsf{negl}(\lambda).$$

We obtain unpredictability *(a weaker variant) by replacing the indistinguishability game with a guessing game on a uniformly chosen s.*

For a reconstruction transcript $T = (z \text{ or } \{z_j\}_{j\in V}, \pi_{\mathrm{rec}})$, let $\mathsf{Contrib}(T) \subseteq [n]$ denote the set of indices j for which T contains a valid contribution, i.e. a proof π_j accepted by the NIZK verifier on the statement determined by ak and z_j.

Definition 6 (Accountability). Accountability *is the conjunction of two clauses,* (A1) *single-query signer attribution and* (A2) *non-imputability, both of which an Acc-VSS must satisfy.*

(A1) Single-query attribution. *For all $n > 1$, $t < n$ and all PPT $\mathcal{A}$ controlling a set $Q \subseteq [n]$ of shareholders,*

$$\Pr\left[\begin{array}{l}\mathsf{ak}, \{s_i\}_{i\in[n]} \leftarrow \Pi_{\mathsf{Share}}(\mathsf{pp}, n, t, s);\\ T = (z \text{ or } \{z_j\}_{j\in V}, \pi_{\mathrm{rec}}) \leftarrow \mathcal{A}\big(\mathsf{ak}, \{s_i\}_{i\in Q}, F\big);\\ (b, I, \pi_{\mathsf{Audit}}) \leftarrow \mathsf{VerifAudit}(T, \mathsf{ak}) :\\ (b = 1 \wedge I = \mathsf{Contrib}(T) \wedge I \subseteq Q \wedge \mathsf{Judge}(I, \pi_{\mathsf{Audit}}, \mathsf{ak}) = 1) \vee\\ (b = 0 \wedge \mathsf{Contrib}(T) = \emptyset)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Clause (A1) captures the single-query attribution requirement: a public verifier presented with the reconstruction transcript runs VerifAudit *once and either reads off the unique set I of contributors ($b = 1$) or rejects ($b = 0$). No interaction with a reconstruction box is required. The attribution is* complete*. The auditor must return exactly $I = \mathsf{Contrib}(T)$, and it may reject only if T contains no valid contribution at all. It therefore cannot reject an attributable transcript, nor return only some of its contributors. Since* Combine *(and any application that accepts T) computes its output only from valid contributions, I is precisely the set of shareholders whose contributions determine the accepted output. The clause additionally requires that I contains only parties whose shares the adversary actually holds (no false positives outside the corrupt set) and that the public* Judge *algorithm confirms the attribution. This differs from black-box, multi-query tracing in two ways: the auditor does not interact with any external object beyond the transcript, and the guarantee only applies to transcripts that the auditor accepts.*

(A2) Non-imputability. *For all $n > 1$, $t < n$, all honest-target indices $i^\star$, and all PPT $\mathcal{A}$ controlling every P_i with $i \neq i^\star$ (and hence holding their shares),*

$$\Pr\left[\begin{array}{l}\mathsf{ak}, \{s_i\}_{i\in[n]} \leftarrow \Pi_{\mathsf{Share}}(\mathsf{pp}, n, t, s);\\ (1, I, \pi_{\mathsf{Audit}}) \leftarrow \mathcal{A}\big(\mathsf{ak}, \{s_i\}_{i\in[n]\setminus\{i^\star\}}\big) :\\ \mathsf{Judge}(I, \pi_{\mathsf{Audit}}, \mathsf{ak}) = 1 \wedge i^\star \in I\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Clause (A2) ensures that even an adversary holding the shares of all but one shareholder cannot produce a transcript that frames the honest party. The sharing phase in (A2) is run honestly; non-imputability against a dealer that additionally deviates during sharing is not claimed here. Together, (A1) and (A2) are what we mean by accountability*: every accepted transcript attributes to its contributors with a single query, and that attribution is safe against framing.*

3.3 Generic transformation: VSS ⇒ Acc-VSS

We now describe the generic transformation. Given any computationally-secure Shamir-based VSS $\Gamma' = (\mathsf{Setup}', \Pi'_{\mathsf{Share}}, \Pi'_{\mathsf{Verify}}, \mathsf{Reconstruct}')$ in the honest-majority setting, a perfectly-binding commitment scheme $\mathcal{C}$ that is one-way on

Setup: Run Setup' for Γ' and the setup of $\mathcal{C}$ and Π_{NIZK}. Publish $\mathsf{pp} = (\mathsf{pp}', \mathsf{pp}_{\mathcal{C}}, \mathsf{pp}_{\mathsf{NIZK}})$.

Π_{Share}**:** To share a secret $s := f_0$, the dealer runs Π'_{Share} on $(\mathsf{pp}', n, t, f_0)$, privately distributing $\{f_i\}_{i\in[n]}$ and broadcasting the dealer proof π_D. *Concurrently*, each shareholder P_i samples $\gamma_i \leftarrow\!\!\$\ \mathbb{Z}_q$ and broadcasts $c'_i := \mathcal{C}(\gamma_i)$. At the end, the public bulletin contains $\mathsf{ak} = (\pi_D, c'_1, \ldots, c'_n)$, and each P_i stores $s_i := (f_i, \gamma_i)$.

Π_{Verify}**:** (i) Each P_i runs the per-share check $\mathsf{Verify}'_i(\pi_D, f_i)$ of the underlying VSS, and additionally checks that every c'_k for $k \neq i$ is well-formed under $\mathcal{C}$. (ii) Mismatches trigger broadcasts of complaints, resolved by the standard conflict-resolution procedure of Γ' for π_D and by simple membership tests for the c'_k. The protocol returns a qualified set $Q \subseteq [n]$ with $|Q| \geq t+1$, or aborts.

$\mathsf{Verify}_i(\mathsf{ak}, s_i)$**:** Returns 1 iff $\mathsf{Verify}'_i(\pi_D, f_i) = 1$ and $c'_i = \mathcal{C}(\gamma_i)$ is well-formed. (This is the local primitive of step (i) above.)

$\mathsf{Reconstruct}_i$ **(per-party):** On input F and $s_i = (f_i, \gamma_i)$, P_i outputs (z_i, π_i) where $z_i = F(f_i)$ and π_i is a NIZK proof for the relation

$$R_i^{\mathrm{gen}} = \left\{ (\mathsf{ak}, F, z_i,\ wit_i = (f_i, \gamma_i)) \;\middle|\; z_i = F(f_i) \wedge \mathsf{Verify}'_i(\pi_D, f_i) = 1 \wedge c'_i = \mathcal{C}(\gamma_i) \right\}. \quad (1)$$

Combine: On a candidate $\{(z_i, \pi_i)\}_{i\in Q}$, verify each π_i using $\Pi_{\mathsf{NIZK}}.\mathsf{V}$ and ak, collect valid pairs $\{(z_j, \pi_j)\}_{j\in V}$, and build the bitstring $\mathsf{who} \in \{0,1\}^n$ that records V. Then: $|V| = 0$: return $\perp$. $|V| \geq t+1$: reconstruct $z := F(f_0)$ via the underlying VSS combine of $\{z_j\}_{j\in V}$, set $\pi_{\mathrm{rec}} = (\mathsf{who}, \{\pi_j\}_{j\in V})$, return (z, π_{rec}). $0 < |V| < t+1$: return $(\{z_j\}_{j\in V}, \pi_{\mathrm{rec}})$.

VerifAudit: Re-verify each π_j in π_{rec} against the corresponding statement in ak, set $I \subseteq V$ to the set of accepted indices, and return $(b, I, \pi_{\mathsf{Audit}})$ with $b = 1$ iff $|I| > 0$ and $\pi_{\mathsf{Audit}} = \{\pi_j\}_{j\in I}$; otherwise $(0, \emptyset, 0^n)$.

Judge: Re-verify each $\pi_j \in \pi_{\mathsf{Audit}}$ against ak and the recorded statement; output 1 iff all accept.

Fig. 1. Generic VSS-to-Acc-VSS transformation Γ from a Shamir-based VSS Γ', a perfectly-binding, one-way commitment $\mathcal{C}$, and a NIZK Π_{NIZK}.

uniform inputs (no PPT adversary given $\mathcal{C}(\gamma)$ for uniform γ outputs γ, except with negligible probability; hiding is *not* required), and a NIZK proof system Π_{NIZK} for a relation that we make explicit below, we build an Acc-VSS Γ as in Figure 1.

The high-level idea is as follows: each share is split into a dealer-generated component f_i and a per-party audit secret γ_i, the latter sampled by P_i itself. The dealer publishes the ordinary VSS commitments to $\{f_i\}$, while each P_i publishes a public commitment $c'_i = \mathcal{C}(\gamma_i)$. During reconstruction, every party outputs $z_i = F(f_i)$ together with a NIZK proof of *conjunction*: z_i was computed from the same f_i that the dealer's commitments determine, *and* P_i knows the audit secret γ_i behind c'_i.

Beyond this basic design, (i) we prove the generic transformation secure as a black-box construction (Theorem 1); and (ii) we strengthen the binding so that the audit secret γ_i is the shareholder's *long-term identity key*, registered once in a public phase rather than freshly sampled at every share-out (Section 3.3). We describe the unmodified version first, then layer the identity binding on top.

Identity binding: γ_i as long-term key. To prevent a malicious P_i from selling f_i without selling γ_i, we move γ_i out of the per-instance Share protocol and into a one-time public registration phase: every potential shareholder P_i samples $\gamma_i \leftarrow\!\!\$\ \mathbb{Z}_q$, sets $\mathsf{ik}_i = \gamma_i$ as its long-term identity key, and publishes $\mathsf{vid}_i := \mathcal{C}(\gamma_i)$ on a registration bulletin. Subsequent Acc-VSS instances reuse $(\mathsf{ik}_i, \mathsf{vid}_i)$ unmodified; the per-instance c'_i is set to vid_i, the proof π_i is bound to the session by including a unique session identifier sid in its statement (and hence in the Fiat–Shamir hash), and R_i^{gen} is updated to require knowledge of the unique γ_i behind

vid_i. Three consequences follow (selling f_i alone is useless across the application layer; leaks are cumulatively attributable across all of P_i's instances; no additional application setup is required); see Appendix C.4 for the full discussion. We instantiate this binding concretely in Section 4; it composes cleanly with both the Feldman and Pedersen variants.

Theorem 1 (Generic VSS-to-Acc-VSS). *Let Γ' be a computationally-secure Shamir-based VSS in the honest-majority setting, $\mathcal{C}$ a perfectly-binding, one-way commitment scheme, and Π_{NIZK} a NIZK proof system for the relation R_i^{gen} of (1) that satisfies completeness, zero-knowledge, and knowledge soundness. Then the construction Γ of Figure 1 is a secure Acc-VSS: it satisfies correctness (Def. 3), verifiability (Def. 4), secrecy (Def. 5, inheriting unpredictability or statistical secrecy from Γ'), and accountability (Def. 6, both clauses (A1) and (A2)).*

Proof. The proof is given in Appendix E.1, where the game-based reductions are assembled from the proofs of the concrete instantiations (Theorems 2 and 5). □

The natural baseline, an actively secure VSS in which each contribution is additionally signed under a registered long-term key, is itself an instance of this transformation (the signature acts as a signature of knowledge of γ_i), so Theorem 1 also covers it; we develop this comparison in Appendix C.5. Appendix C.3 further discusses the role of each conjunct of R_i^{gen} and delimits what Theorem 1 does and does not promise.

4 Constructions of Acc-VSS

The generic VSS-to-Acc-VSS transformation of Section 3.3 requires only a perfectly-binding, one-way commitment scheme $\mathcal{C}$ and a Shamir-based VSS Γ'. In this section, we instantiate it on the two canonical computationally-secure VSS schemes, Feldman [21] and Pedersen [34], using a discrete-logarithm commitment for $\mathcal{C}$; the Feldman-based scheme is presented below, and the Pedersen-based one is deferred to Appendix B. Both instantiations carry the per-party identity binding of Section 3.3: the audit secret γ_i is the long-term identity key of P_i, registered once and reused across Acc-VSS instances. Where the underlying VSS gives statistical secrecy (Pedersen) we get statistical secrecy; where it gives unpredictability under DL (Feldman) we get unpredictability under DL.

4.1 Acc-VSS from Feldman VSS

Feldman VSS. One of the well-known and widely used computationally secure VSS schemes is Feldman's protocol [21], in which the dealer performs Shamir secret sharing of a *high-entropy* secret $f_0 \in \mathbb{Z}_q$ and publishes DL-based commitments to the coefficients of the secret polynomial, against which each party verifies its share; due to space constraints, we recall the scheme in Appendix A.2.

Construction. Using the generic transformation of Figure 1, we instantiate Γ' with Feldman VSS and $\mathcal{C}(\cdot)$ with the DL-based commitment $\mathcal{C}(x) := g^x$, where g is the same group generator used in Feldman VSS. This map is injective, hence perfectly binding, and it is one-way on uniform inputs exactly under the DL assumption. It is not hiding, and the proofs do not need it to be. The audit secret γ_i is the *long-term identity key* of P_i (Section 3.3): in a one-time public registration phase, P_i publishes $\mathsf{vid}_i := g^{\gamma_i}$ as its identity, and every subsequent Acc-VSS instance reuses this vid_i unmodified. Per-instance, the published commitment is simply $c'_i := \mathsf{vid}_i$. To keep proofs π_i from being replayed across sessions, a unique session identifier sid is included in the statement of π_i (and hence in the Fiat–Shamir hash).

During the sharing phase, the dealer proceeds exactly as in Feldman VSS, distributing the shares $\{f_i\}_{i\in[n]}$ privately and broadcasting $\pi_D = (c_0, c_1, \ldots, c_t)$. Each P_i stores $s_i := (f_i, \gamma_i)$, where γ_i is its *long-term* identity key (unchanged across instances). The verification protocol Π_{Verify} runs Feldman's share check $g^{f_i} = \prod_{j=0}^{t} c_j^{i^j}$ and additionally requires that vid_i is the registered identity of party P_i (a one-line lookup in the identity registry); a missing or malformed vid_i disqualifies P_i.

In the Reconstruct phase of the Feldman-based Acc-VSS, any set of shareholders can publish a function of their shares; a qualified set of at least $t+1$ honest shareholders can additionally compute the plain value f_0 or any $z = F(f_0)$ that the function family supports. If a party P_i chooses to publish a function of their shares, such as $z_i = F(f_i)$, they must provide a non-interactive proof that z_i is correctly computed from f_i and they know the original share $s_i := (f_i, \gamma_i)$, sampled in the sharing phase. More formally, each party P_i needs to use a NIZK proof scheme, e.g., $\Pi^{(1)}_{\mathsf{NIZK}}$, and generate a proof for the relation $R^{(1)}_i$, defined as follows,

$$R^{(1)}_i = \left\{ (g, F(\cdot), z_i, y_i, \mathsf{vid}_i, (f_i, \gamma_i)) \;\middle|\; z_i = F(f_i) \wedge y_i = g^{f_i} \wedge \mathsf{vid}_i = g^{\gamma_i} \right\}, \quad (2)$$

where $y_i = g^{f_i} = \prod_{j=0}^{t} c_j^{i^j}$ can be publicly (pre)computed using (a valid) π_{share} and the *public index* i. Let π_i be the individual NIZK proof (or argument) published by party P_i, which is generated by a NIZK proof scheme $\Pi^{(1)}_{\mathsf{NIZK}}$. In practice, depending on the function $F(\cdot)$, the proof scheme $\Pi^{(1)}_{\mathsf{NIZK}}$ can be constructed using different families of zero-knowledge proof systems (possibly with non-malleable proofs), e.g., sigma protocols [35] and zk-SNARKs [31].

Theorem 2 (Acc-VSS from Feldman VSS). *Let $\Pi^{(1)}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a NIZK proof system for the relation $R^{(1)}_i$ of (2) that satisfies completeness, zero-knowledge, and knowledge soundness (Appendix A.4). Then, under the discrete-logarithm assumption, the protocol of Figure 2 is a secure Acc-VSS: it satisfies correctness, verifiability, unpredictability, and accountability (both clauses (A1) single-query attribution and (A2) non-imputability) as defined in Definitions 3 to 6.*

Proof. The proof is given in Appendix E.2. □

Initial: Let $\mathbb{G}$ be a group of prime order $q > n$ with generator g, where the DL problem is hard. The group generator g and the public parameters for the NIZK proof scheme $\Pi^{(1)}_{\mathsf{NIZK}}$ are shared with all the parties.

Share: Given (n, t), and a random secret $s := f_0 \in \mathbb{Z}_q$, the protocol proceeds as follows:

- The dealer samples a random degree-t polynomial $f(x) = f_0 + a_1 X + \cdots + a_t X^t$, with random coefficients from $\mathbb{Z}_q$, such that $f(0) = f_0$. Next, the dealer computes $f_i := f(i)$ for $i \in [n]$ and privately sends f_i to party P_i. Then, the dealer computes $c_0 = g^{f_0}, c_1 = g^{a_1}, \ldots, c_t = g^{a_t}$ and publishes a proof $\pi_D := (c_0, \ldots, c_t)$.
- In the same round, for $i \in [n]$, each party P_i uses its long-term identity key $\gamma_i \in \mathbb{Z}_q$, sampled once and registered as $\mathsf{vid}_i := g^{\gamma_i}$ in the one-time registration phase of Section 3.3, and broadcasts $c'_i := \mathsf{vid}_i$.
- At the end, each party P_i stores its individual share $s_i := (f_i, \gamma_i)$ and the public proof and the audit/verification keys $\mathsf{ak} = vk := (\pi_D, c'_1, \ldots, c'_n)$ which are generated jointly by the dealer and the parties.

Verify: Given $\pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$ and the shares, i.e., $s_i := (f_i, \gamma_i)$ for $i \in [n]$, the parties proceed as follows:

1) Each party P_i checks if $g^{f_i} = \prod_{j=0}^{t} c_j^{i^j}$, and broadcasts a complaint against the dealer if the verification failed. Party P_i also checks if $c'_k \in \mathbb{G}$ for $k \in [n]$, and $k \neq i$. In case a commitment c'_k is missing or it is not well-formed, party P_i disqualify P_k and broadcasts a complaint against it.
2) If the number of parties complaining against party P_k exceeds a threshold value t, the party P_k is disqualified. Similarly, if the number of shareholders complaining against the dealer exceeds a threshold value t, the dealer will be disqualified, and the verification process will result in a `false` outcome.
3) If a shareholder P_i raises a complaint against the dealer, the dealer will respond by broadcasting f_i so that everyone can verify it using the verification equation. If the verification succeeds, the protocol continues as normal. However, if it fails, the dealer will be disqualified, resulting in a 'false' verification outcome. Since the disqualification decision is based solely on publicly shared information, all honest shareholders will eventually agree either on a qualified set of parties $Q \subseteq \{P_1, P_2, \ldots, P_n\}$ or on rejecting the final verification.

Reconstruct: Given a function $F(\cdot)$, which can be the identity function as well, the proof $\pi_{share} := (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, and the shares $\{s_i := (f_i, \gamma_i)\}_{i \in Q}$, the parties proceed as follows.

- Each party P_i, for $i \in Q$, publishes $(z_i = F(f_i), \pi_i)$, where π_i is a non-interactive proof (or argument) generated by $\Pi^{(1)}_{\mathsf{NIZK}}$ for the relation $R^{(1)}_i$ defined in equation (2). The proof π ensures that z_i is computed correctly and P_i knows the original share $s_i := (f_i, \gamma_i)$. If $F(\cdot)$ is the identity, $z_i = f_i$ is public, but γ_i is never revealed: π_i only proves knowledge of it.
- Given π_{share}, the broadcasted values $\{z_i := F(f_i), \pi_i\}_{i \in Q}$ are verified using the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ and the valid ones are collected. Let $\{z_j = F(f_j), \pi_j)\}_{j \in V}$ be the set of valid pairs, V be the set of their indexes, and who be an n-bit string where each of its bit indicates whether the corresponding party has provided a valid proof.
- If $|V| = 0$, the protocol returns $\perp$;
- If $t + 1 \leq |V|$: the protocol reconstructs $z = F(f_0)$ and generates associated proof $\pi_{rec} := \{\mathsf{who}, \pi_j\}_{j \in V}$, to show that z is computed correctly and the parties indicated in who know $\{s_j\}_{j \in V}$. Finally, it returns $(z = F(f_0), \pi_{rec})$. If $F(\cdot)$ is the identity function, the published $z_j = f_j$ are the plain shares, while $\{\pi_j\}_{j \in V}$ still only prove knowledge of each γ_j.
- If $0 < |V| < t + 1$: the protocol returns $\{z_j := F(f_j)\}_{j \in V}$ and a proof $\pi_{rec} := \{\mathsf{who}, \pi_j\}_{j \in V}$ to show that $\{z_j\}_{j \in V}$ are computed correctly by the parties indicated in who and they know $\{s_j\}_{j \in V}$.

VerifAudit: Given $\{z := F(f_0), \pi_{rec}\}$ or $\{\{z_j := F(f_j)\}_{j \in V}, \pi_{rec}\}$, generated by Reconstruct, and audit key $\mathsf{ak} := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, it computes $y_j := \prod_{k=0}^{t} c_k^{j^k}$ for all the parties $j \in V$ (deducible from who) and then verifies the proofs $\pi_{rec} := \{\mathsf{who}, \pi_j\}_{j \in V}$ using the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ and commitment values $\{y_j, c'_j\}_{j \in V}$, and finally determines $I \subseteq V$ as the set of valid ones. Finally, it returns $1, I$ and $\pi_{\mathsf{Audit}} := (z = F(f_0)$ or $\{z_j = F(f_j)\}_{j \in I}, \{\pi_j\}_{j \in I})$ if $|I| > 0$; otherwise, it returns $\{0, \emptyset, 0^n\}$.

Judge: Given an alleged set I, and a proof $\pi_{\mathsf{Audit}} := (z = F(f_0)$ or $\{z_j = F(f_j)\}_{j \in I}, \{\pi_j\}_{j \in I})$, and $vk := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, using the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$, it verifies if π_{Audit} is valid and outputs `true` or `false`.

Fig. 2. The Feldman-based Acc-VSS of Section 4.1.

Efficiency. The Feldman-based Acc-VSS adds minimal overhead over Feldman VSS. During Π_{Share}, the dealer performs exactly the same computations as in the original scheme; the per-instance addition is $c'_i = \mathsf{vid}_i$ (with sid bound inside the proofs), which does not require any exponentiation since vid_i is reused from the registration phase. Per-share verification Verify_i runs the original Feldman check (4) ($t+1$ small exponentiations) plus a single membership lookup of vid_i in the public identity registry. Per-party shares are (f_i, γ_i), twice the size of a Feldman share, but γ_i is reused across all of P_i's Acc-VSS instances and so the marginal storage per instance is just $|f_i|$.

In the Reconstruct phase, each participant uses the NIZK proof system $\Pi^{(1)}_{\mathsf{NIZK}}$ to generate a proof π_i for the relation $R_i^{(1)}$. If the application requests plain shares (identity F), π_i collapses to $s_i = (f_i, \gamma_i)$ and verification reduces to two exponentiations against ak. For non-trivial F, the proof is a Schnorr-style Σ-protocol of conjunction (Section 5) of constant size.

VerifAudit and Judge run only $\Pi^{(1)}_{\mathsf{NIZK}}.\mathsf{V}$ on the per-party proofs, with cost linear in $|I|$ and *publicly* executable: there is no trusted tracer.

The Feldman-based Acc-VSS requires a high-entropy secret and achieves *unpredictability* (Def. 5, the weaker variant); for low-entropy secrets, or when statistical secrecy is required, the generic transformation can instead be instantiated on Pedersen VSS [34]. Due to space constraints, the resulting construction and its security analysis are deferred to Appendix B.

Experimental evaluation. Figure 7 benchmarks all four schemes on the ristretto255 group for $n \in [10, 100]$ with $t = \lfloor n/2 \rfloor$ (100 iterations each). The per-party accountability overhead in the reconstruction phase is a constant $\approx 40\,\mu$s (two scalar multiplications), independent of n. Appendix C.11 gives a detailed breakdown.

5 Accountable Threshold Cryptography

In this section, we use the Acc-VSS schemes of Section 4 as building blocks for *Accountable Threshold Cryptography* (Acc-TC). We present three concrete instantiations (accountable threshold ElGamal decryption, accountable distributed key generation (DKG), and accountable threshold Schnorr/FROST signing) and discuss two deployment-driven applications: encrypted-mempool MEV resistance and accountable e-voting tallies.

In a standard (n, t)-threshold cryptosystem, the key generation produces $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n)$, decryption is collaborative on at least $t+1$ partial decryptions, and the combine algorithm reconstructs the plaintext [13, 14]. Actively-secure variants additionally require per-party commitments $\mathsf{pk}_i = \mathcal{C}(\mathsf{sk}_i)$ so that each partial decryption can be checked. None of these variants offers *public attribution*: an external verifier that consumes the message m produced by Combine cannot tell which $t+1$ parties actually decrypted ct, a problem in any deployment where share-leakage carries a financial or reputational cost (encrypted mempools, election tallies, proof-of-stake validator key custody).

Our construction. Figure 4 promotes a standard ElGamal-based threshold decryption [18] into a Acc-TC by replacing the per-party VSS with our Feldman-based Acc-VSS (Section 4.1) and replacing the partial-decryption proof with the NIZK conjunction $\Pi^{Dec}_{\mathsf{NIZK}}$ of (3). The audit transcript that Combine emits is exactly the bundle of partial-decryption proofs, with the per-party identity binding of Section 3.3. We then extend the construction to threshold Schnorr/FROST signing (Section 5.3) and to a Acc-TC-flavoured Pedersen DKG (Section 5.2).

Accountable threshold encryption: syntax and requirements. An Accountable Threshold Cryptosystem (Acc-TC) is a 6-tuple (KeyGen, Enc, Dec, Combine, VerifAudit, Judge) of PPT procedures. The first four are the standard threshold-decryption components [13,14,18], lifted to take and produce audit-bound partial outputs; VerifAudit and Judge are the public attribution components inherited from Acc-VSS (Section 3). The syntax is adapted from [5,14]. Due to space constraints, the formal syntax (Def. 22) and the security requirements of Acc-TC, namely correctness, IND-CPA security, public auditability, and non-imputability (Definitions 23 to 26), are deferred to Appendix C.6; the latter two mirror clauses (A1) and (A2) of Def. 6.

5.1 An Efficient Accountable Threshold Cryptosystem

Next, we present the concrete Acc-TC protocol $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$ based on ElGamal [18]. It plugs the Feldman-based Acc-VSS of Section 4.1 into a standard threshold ElGamal decryption, replacing the per-party Chaum–Pedersen DL-equality NIZK with a NIZK conjunction $\Pi^{Dec}_{\mathsf{NIZK}}$ that additionally proves knowledge of the long-term identity secret γ_i.

Concrete Example of $\Pi^{(1)}_{\mathsf{NIZK}}$ for Use in $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$. Before presenting the full construction of $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$, we first construct a concrete NIZK proof scheme $\Pi^{Dec}_{\mathsf{NIZK}}$, which is essential for ensuring both the correctness and the public auditability of the decryption process. The NIZK proof $\Pi^{Dec}_{\mathsf{NIZK}}$ is a specific instantiation of $\Pi^{(1)}_{\mathsf{NIZK}}$ used in the reconstruction phase of the Feldman-based Acc-VSS. More precisely, $\Pi^{Dec}_{\mathsf{NIZK}}$ is tailored to a special case of the relation $R^{(1)}_i$ defined in Eq. (2), with two key modifications. First, the function $F(\cdot)$ is instantiated as the exponentiation function, meaning $d_i = c^{f_i}$, where $c \in \mathbb{G}$ and $f_i \in \mathbb{Z}_q$. Second, we write $\mathsf{pk}_i := y_i = g^{f_i}$ for the per-party verification key, which is publicly computable from the dealer's commitments, and keep the registered long-term identity $\mathsf{vid}_i = g^{\gamma_i}$ as a separate statement. In this way, the relation links γ_i directly to the identity vid_i of Section 3.3.

In summary, the new NIZK proof $\Pi^{Dec}_{\mathsf{NIZK}}$ allows a prover to generate a proof π for the relation R^{Dec}_i, defined as follows:

$$R^{Dec}_i = \{(g, c, d_i, \mathsf{pk}_i, \mathsf{vid}_i, (f_i, \gamma_i)) \mid d_i = c^{f_i} \ \wedge \ \mathsf{pk}_i = g^{f_i} \ \wedge \ \mathsf{vid}_i = g^{\gamma_i}\}, \quad (3)$$

where $x := (g, c, d_i, \mathsf{pk}_i, \mathsf{vid}_i) \in \mathbb{G}^5$ is the statement, and $w := (f_i, \gamma_i) \in \mathbb{Z}_q$ is the witness value.

Setup: Given 1^λ, the parties sample (or agree on) a cyclic group $\mathbb{G}$ of prime order q with hard DL and a generator g, and then share $pp := g$ with all the parties.

P: Given the public parameters $pp := g$, the statement $x := (d_i, c, \mathsf{pk}_i, \mathsf{vid}_i)$, a session identifier sid, and witness $w := (f_i, \gamma_i) \in \mathbb{Z}_q^2$, the prover acts as follows:

- Samples $(r_1, r_2) \xleftarrow{\$} \mathbb{Z}_q$ and commits to them as $R_1 = c^{r_1}$, $R_2 = g^{r_1}$, and $R_3 = g^{r_2}$.
- Computes the challenge $e = \mathcal{H}(\mathsf{sid}, d_i, c, \mathsf{pk}_i, \mathsf{vid}_i, R_1, R_2, R_3)$, where $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q$ is modeled as a random oracle.
- Finally, computes $z_1 = r_1 + ef_i$ and $z_2 = r_2 + e\gamma_i$ and publishes the proof $\pi := (R_1, R_2, R_3, z_1, z_2)$.

V: Given the public parameters $pp := g$, the statement $x := (d_i, c, \mathsf{pk}_i, \mathsf{vid}_i)$, sid, and the proof $\pi = (R_1, R_2, R_3, z_1, z_2)$, the verifier computes the challenge value $e = \mathcal{H}(\mathsf{sid}, d_i, c, \mathsf{pk}_i, \mathsf{vid}_i, R_1, R_2, R_3)$ and checks if:

$$c^{z_1} \stackrel{?}{=} R_1\ d_i^e, \quad g^{z_1} \stackrel{?}{=} R_2\ \mathsf{pk}_i^e, \quad \text{and} \quad g^{z_2} \stackrel{?}{=} R_3\ \mathsf{vid}_i^e$$

and returns `true` if the check passes, and `false` otherwise.

Fig. 3. $\Pi^{Dec}_{\mathsf{NIZK}}$: An efficient NIZK proof of knowledge for R_i^{Dec}, as defined in Eq. (3).

We present the details of the NIZK proof $\Pi^{Dec}_{\mathsf{NIZK}}$ in Figure 3. This proof is constructed by AND-composing, with a shared challenge, the Chaum–Pedersen DL-equality Σ-protocol [7] for $\log_c d_i = \log_g \mathsf{pk}_i = f_i$ with a Schnorr Σ-protocol [35] for $\mathsf{vid}_i = g^{\gamma_i}$. The protocol is then converted into a non-interactive proof of knowledge using the Fiat-Shamir transform in the random oracle model.

Intuitively, the new NIZK proof enables each participant in the decryption protocol to prove that their partial decryption has been done correctly, while also showing that they possess knowledge of their individual shares $s_i := (f_i, \gamma_i)$.

Theorem 3 (NIZK Proof of Knowledge for R_i^{Dec}). *Let $\mathcal{H}$ be a random oracle, defined as $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q$. The proof scheme $\Pi^{Dec}_{\mathsf{NIZK}}$, as outlined in Figure 3, is a secure NIZK proof of knowledge for the relation R_i^{Dec} in the random oracle model.*

Proof. The proof is given in Appendix E.3. □

Construction of $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$. Figure 4 outlines the construction of $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$. The protocol is a standard ElGamal-based threshold decryption that emits an audit transcript with every accepted output. The audit binding is the NIZK proof $\Pi^{Dec}_{\mathsf{NIZK}}$ of Figure 3, which proves that the partial decryption d_i was computed from the same f_i that the dealer's commitments determine *and* that P_i knows the long-term identity secret γ_i behind vid_i.

Theorem 4 (Accountable Threshold ElGamal). *Assume the DL assumption holds in $\mathbb{G}$. Let $\Pi^{Dec}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a NIZK proof system for the relation R_i^{Dec} of (3) satisfying completeness and knowledge soundness (Appendix A.4), and let ElGamal be IND-CPA-secure (Appendix A.5). Then the protocol of Figure 4 is a secure Acc-TC: it satisfies correctness, IND-CPA, and accountability (both clauses (A1) single-query attribution and (A2) non-imputability) as defined in Definitions 23 to 26.*

Proof. The proof is given in Appendix E.4. □

KeyGen: Given 1^λ, and the integer values (n, t), the algorithm proceeds as follows:
- It samples a cyclic group $\mathbb{G}$ with a prime order $q > n$ and a generator g, where DL is hard. It then sets $pp := g$ as the public parameters for the NIZK proof scheme $\Pi^{Dec}_{\mathsf{NIZK}}$.
- It takes as input the registered identities $\{\mathsf{vid}_i = g^{\gamma_i}\}_{i=1}^n$ of Section 3.3; each long-term γ_i is known only to P_i and is not sampled by KeyGen.
- Given (n, t), it samples a random secret key $f_0 \in \mathbb{Z}_q$, and then secret shares it using Shamir's scheme and obtains the shares $\{f_i\}_{i=1}^n$.
- For $i = 1, \ldots, n$, it sets the individual shares $\mathsf{sk}_i := (f_i, \gamma_i)$ (where P_i adds its own γ_i to the received f_i) and computes $\mathsf{pk}_i := g^{f_i}$.
- It sets $\mathsf{pk} = g^{f_0}$, and $\mathsf{vk} := \mathsf{tk} := (pp, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \mathsf{vid}_1, \ldots, \mathsf{vid}_n)$, and returns $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \mathsf{vk}, \mathsf{tk})$.

Note that, at the end of the key generation phase, the encryptor only stores the public key pk, while each shareholder i learns their individual share $\mathsf{sk}_i = (f_i, \gamma_i)$, in addition to the public parameters.

Enc: Given the public key pk and a message $m \in \mathbb{G}$, it first samples a randomizer $r \leftarrow \mathbb{Z}_q^*$ and then computes and returns the ElGamal ciphertext $\mathsf{ct} := (c_1, c_2) = (m \cdot \mathsf{pk}^r, g^r)$.

Dec: Given a ciphertext $\mathsf{ct} = (c_1, c_2)$ and a secret share $\mathsf{sk}_i = (f_i, \gamma_i)$, it first computes the partial decryption value $d_i := c_2^{f_i}$. Then, using $pp := g$, the statement $x := (d_i, c_2, \mathsf{pk}_i, \mathsf{vid}_i)$, and the witness value $w := (f_i, \gamma_i)$, it runs the NIZK proof of knowledge $\Pi^{Dec}_{\mathsf{NIZK}}$ (outlined in Figure 3) and obtains a proof π_i. Finally, it returns (d_i, π_i).

Combine: Given the partial decryption values and the proof $\{(d_i, \pi_i)\}_{i \in Q}$ for the set $Q \subseteq [n]$, the verification key $\mathsf{vk} := (pp, \{\mathsf{pk}_i, \mathsf{vid}_i\}_{i \in [n]})$, and the ciphertext $\mathsf{ct} = (c_1, c_2)$, it first verifies $\{(d_i, \pi_i)\}_{i \in Q}$ and stores the set of valid pairs, i.e., $\{(d_i, \pi_i)\}_{i \in V \subseteq Q}$. From the set of valid pairs, the algorithm constructs an n-bit string who, where each bit indicates whether the corresponding party has provided a valid proof. Then, the algorithm acts as follows:
- If $|V| = 0$, returns $\perp$.
- If $|V| \geq t + 1$, for each $i \in V$, it computes the Lagrange coefficients L_i. Then, it computes $m = c_1 / \left(\prod_{i \in V} d_i^{L_i}\right)$. Finally, it returns (m, π_{dec}), where π_{dec} consists of all the proofs provided by the decryptors in V and the string who, i.e., $\{\pi_i, \mathsf{who}\}_{i \in V}$.
- If $0 < |V| < t + 1$, it returns $(\{d_i\}_{i \in V}, \pi_{dec})$, where $\pi_{dec} = \{\pi_i, \mathsf{who}\}_{i \in V}$.

VerifAudit: Given a message $m \in M$ or a set of decryption shares $\{d_j\}_{j \in V}$ (where the set V can be deduced from the string who), the corresponding proof π_{dec} generated by the Combine algorithm, the ciphertext $\mathsf{ct} := (c_1, c_2)$, along with the verification key $\mathsf{vk} := (pp, \{\mathsf{pk}_i, \mathsf{vid}_i\}_{i \in [n]})$, the algorithm first verifies $\pi_{dec} = \{\pi_i, \mathsf{who}\}_{i \in V}$ using the verification algorithm V (outlined in Figure 3). If the verification fails for all indexes $j \in V$, it returns $\{0, \emptyset, 0^n\}$. Otherwise, it identifies the set of valid ones $I \subseteq V$ and returns $(1, I, \pi_{\mathsf{Audit}})$, where $\pi_{\mathsf{Audit}} := \{\mathsf{ct}, d_k, \pi_k\}_{k \in I}$, and $I \subseteq V$.

Judge: Given a set of decryptors I, a proof $\pi_{\mathsf{Audit}} := \{\mathsf{ct}, d_k, \pi_k\}_{k \in I}$, and the verification key $\mathsf{vk} := (pp, \{\mathsf{pk}_i, \mathsf{vid}_i\}_{i \in [n]})$, it runs the verification algorithm V (outlined in Figure 3) for all $k \in I$ and returns `true` if all the checks passed. Otherwise, it returns `false`.

Fig. 4. $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$: an accountable threshold cryptosystem from Feldman-based Acc-VSS and ElGamal encryption.

5.2 Accountable Distributed Key Generation

The construction in Figure 4 assumes a centralised key-generation phase, but in every deployment of practical interest the key-generation step is itself distributed. Our Acc-VSS schemes (Section 4.1 and Appendix B) extend into an accountable DKG (Acc-DKG) by having every party act as a dealer in parallel (in the style of Pedersen and GJKR [24, 34]), with each P_i's long-term identity key γ_i registered as in Section 3.3 so that the implicit joint share $\mathsf{sk}_i = (f_i, \gamma_i)$ inherits the audit binding. The resulting Acc-DKG produces $(\mathsf{pk}, \{\mathsf{sk}_i\}, \mathsf{vk}, \mathsf{tk})$ in a fully distributed manner, and every subsequent threshold operation attributes back to the registered vid_i's; Appendix C.9 gives the full construction. Due to space constraints, the detailed efficiency analysis of $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$ and its comparison with the non-accountable, actively secure threshold ElGamal baseline are deferred to Appendix C.7.

5.3 Accountable Threshold Schnorr/FROST

The same recipe applies to threshold Schnorr signing, including the FROST family [9, 11, 32]. In brief: setup mirrors $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$, each signer outputs a partial signature $\sigma_i = k_i + e\lambda_i f_i \bmod q$ together with a NIZK π_i^{Sch} proving the FROST relation plus knowledge of γ_i behind vid_i, and the audit transcript is the bundle of π_i^{Sch}'s. The accountability overhead is two exponentiations per party over standard FROST. Appendix C.8 states the relation and the protocol in full.

Three deployment scenarios motivate Acc-TC. (i) *Encrypted mempools.* Designs such as [5, 23, 38] use threshold decryption to delay transaction visibility; with $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$, the consensus layer rejects any decryption that does not carry a valid audit transcript, slashing a corrupt opener before any front-run lands. (ii) *Accountable e-voting tallies.* The audit transcript accompanying a tally publicly identifies the authorities that produced it; the underlying VSS also detects an inconsistent dealer at sharing time. (iii) *Validator key custody and slashing.* An Acc-DKG of Section 5.2 lets a validator operator distribute its key among n co-validators or HSMs; slashing rules then key on the audit binding to a registered vid_i. We do not formalise these scenarios; they are motivating examples.

6 Conclusion

We introduced *Accountable Verifiable Secret Sharing* (Acc-VSS) as a natural and practical extension of VSS that meets the next-best target after traceability: *accountability*, a strict relaxation defined by two clauses. (A1) The set of shareholders who contributed to any reconstruction the application accepts is publicly readable from the transcript via a single query, and (A2) honest shareholders cannot be falsely accused, even by an adversary that holds the shares of all other $n - 1$ shareholders. Accountability subsumes non-imputability and replaces the black-box, multi-query box-tracing strategy of TSS with a transcript-level check that any external verifier or smart-contract judge can run.

We focused on Shamir-based VSS and gave a generic transformation that lifts any computationally-secure Shamir-based VSS into an Acc-VSS with one extra field element per share. We instantiated it on Feldman and Pedersen VSS with a long-term identity binding that defeats the "sell f_i without the audit tag" attack at the construction level. We used Acc-VSS as the building block for *Accountable Threshold Cryptography* (Acc-TC), instantiating accountable threshold ElGamal decryption, accountable threshold Schnorr/FROST signing, and accountable distributed key generation.

Several directions remain open, including a constructive recovery of full traceability via a judge that observes accumulated audit transcripts, post-quantum instantiations of the generic transformation, succinct aggregation of the audit transcript, and extensions of Acc-VSS beyond Shamir-based secret sharing; Appendix D discusses them alongside the separations from prior work.

Acknowledgments

This work was supported by the Flemish Government through the Cybersecurity Research Program with grant number VOEWICS02 and through SolidLab Flanders (Flemish Government, EWI).

References

1. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: Practical robust DKG protocols for CSIDH. In: Tibouchi, M., Wang, X. (eds.) ACNS 23International Conference on Applied Cryptography and Network Security, Part II. LNCS, vol. 13906, pp. 219–247. Springer, Cham (Jun 2023). https://doi.org/10.1007/978-3-031-33491-7_9
2. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: VSS from distributed ZK proofs and applications. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part I. LNCS, vol. 14438, pp. 405–440. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8721-4_13
3. Baghery, K.: A unified framework for verifiable secret sharing. In: Jager, T., Pan, J. (eds.) Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Roros, Norway May 12-15, 2025, Proceedings. Lecture Notes in Computer Science, Springer (2025)
4. Ben-Or, M., Goldreich, O., Goldwasser, S., Håstad, J., Kilian, J., Micali, S., Rogaway, P.: Everything provable is provable in zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 37–56. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34799-2_4
5. Boneh, D., Partap, A., Rotem, L.: Accountability for misbehavior in threshold decryption via threshold traitor tracing. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part VII. LNCS, vol. 14926, pp. 317–351. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68394-7_11
6. Boneh, D., Partap, A., Rotem, L.: Traceable secret sharing: Strong security and efficient constructions. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part V. LNCS, vol. 14924, pp. 221–256. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68388-6_9
7. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 89–105. Springer, Berlin, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_7
8. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: 26th FOCS. pp. 383–395. IEEE Computer Society Press (Oct 1985). https://doi.org/10.1109/SFCS.1985.64
9. Chu, H., Gerhart, P., Ruffing, T., Schröder, D.: Practical schnorr threshold signatures without the algebraic group model. Cryptology ePrint Archive, Report 2023/899 (2023), https://eprint.iacr.org/2023/899
10. Crites, E.C., Kohlweiss, M., Preneel, B., Sedaghat, M., Slamanig, D.: Threshold structure-preserving signatures. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part II. LNCS, vol. 14439, pp. 348–382. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8724-5_11

11. Crites, E.C., Komlo, C., Maller, M.: Fully adaptive Schnorr threshold signatures. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 678–709. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_22
12. Damgård, I., Koprowski, M.: Practical threshold RSA signatures without a trusted dealer. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 152–165. Springer, Berlin, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6_10
13. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 120–127. Springer, Berlin, Heidelberg (Aug 1988). https://doi.org/10.1007/3-540-48184-2_8
14. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_28
15. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). https://doi.org/10.1109/TIT.1976.1055638
16. Dziembowski, S., Faust, S., Lizurej, T.: Individual cryptography. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 547–579. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_18
17. Dziembowski, S., Faust, S., Lizurej, T., Mielniczuk, M.: Secret sharing with snitching. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24). ACM (2024). https://doi.org/10.1145/3658644.3690296
18. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Berlin, Heidelberg (Aug 1984). https://doi.org/10.1007/3-540-39568-7_2
19. Farràs, O., Guiot, M.: Traceable secret sharing schemes for general access structures. Cryptology eprint Archive (2025)
20. Farràs, O., Guiot, M.: Information-theoretic strong traceable secret sharing schemes. Cryptology eprint Archive (2026)
21. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th FOCS. pp. 427–437. IEEE Computer Society Press (Oct 1987). https://doi.org/10.1109/SFCS.1987.4
22. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
23. Flashbots: SUAVE: A decentralized block builder network. https://writings.flashbots.net/the-future-of-mev-is-suave (2023)
24. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. Journal of Cryptology **20**(1), 51–83 (Jan 2007). https://doi.org/10.1007/s00145-006-0347-3
25. Gennaro, R., Leigh, D., Sundaram, R., Yerazunis, W.S.: Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 276–292. Springer, Berlin, Heidelberg (Dec 2004). https://doi.org/10.1007/978-3-540-30539-2_20

26. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC. pp. 25–32. ACM Press (May 1989). https://doi.org/10.1145/73007.73010
27. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987). https://doi.org/10.1145/28395.28420
28. Goyal, R., Waters, B.: Threshold traitor tracing for general access structures. Cryptology eprint Archive (2025)
29. Goyal, V., Jain, A., Partap, A.: Traceable secret sharing, revisited. Cryptology eprint Archive, Paper 2025/1980 (2026), https://eprint.iacr.org/2025/1980, https://eprint.iacr.org/2025/1980
30. Goyal, V., Song, Y., Srinivasan, A.: Traceable secret sharing and applications. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 718–747. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84252-9_24
31. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
32. Komlo, C., Goldberg, I.: FROST: Flexible round-optimized Schnorr threshold signatures. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 34–65. Springer, Cham (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0_2
33. Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract) (rump session). In: Davies, D.W. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 522–526. Springer, Berlin, Heidelberg (Apr 1991). https://doi.org/10.1007/3-540-46416-6_47
34. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Berlin, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_9
35. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_22
36. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176
37. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Berlin, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_15
38. Shutter Network: Threshold-encrypted mempool. https://shutter.network (2024)

Appendix:

A Preliminaries

A.1 Secret Sharing and Shamir's Scheme

Secret sharing schemes allow a secret to be divided into multiple shares, enabling any quorum of shareholders to reconstruct the secret, while smaller groups are unable to learn anything about the main secret.

Definition 7 (Threshold Secret Sharing [36]). *For any two positive integers (n, t), an (n, t)-threshold secret sharing scheme over $\mathbb{Z}_p$, for a prime $p > n$, consists of two PPT algorithms,* Share *and* Reconstruct. Share *is a randomized function that takes a secret $s \in \mathbb{Z}_p$, and outputs* $\mathsf{Share}(n, t, s) \rightarrow (s_1, \ldots, s_n)$. *Two main properties are correctness and security, defined as follows:*

- ***Correctness****: For any s and a set of parties $Q \subseteq [n]$ s.t. $|Q| > t$, we have,*

$$\Pr\begin{bmatrix} \forall s, (s_1, \ldots, s_n) \leftarrow \mathsf{Share}(1^\lambda, n, t, s) \\ s' \leftarrow \mathsf{Reconstruct}\left(\{s_i\}_{i \in Q}\right) : s' = s \end{bmatrix} = 1 \ .$$

- ***Security****: For any secret s and a set of parties $Q \subseteq [n]$, for all unbounded adversaries $\mathcal{A}$ we have,*

$$\Pr\begin{bmatrix} Q \leftarrow \mathcal{A}(1^\lambda, n, t), (s_1, \ldots, s_n) \leftarrow \mathsf{Share}(1^\lambda, n, t, s) \\ s^* \leftarrow \mathcal{A}(\{s_i\}_{i \in Q}) : |Q| \leq t \wedge s^* = s \end{bmatrix} = 1/p \ .$$

Shamir Secret Sharing (SSS) scheme [36] is a widely used protocol, which is formally defined below.

Definition 8 (Shamir Secret Sharing (SSS) [36]). *Let p be a prime with $p > n$. In an (n, t)-SSS, to distribute a secret $s \in \mathbb{Z}_p$ among n parties, the dealer creates a random polynomial $f(x) \in \mathbb{Z}_p[X]$ of degree t such that $f(0) = s$. The dealer then evaluates $f(x)$ at each shareholder's public index and securely provides each shareholder with their share $s_i = f(i)$ for $i \in [n]$. To reconstruct the secret, a subset $S \subseteq [n]$ can be used to compute the Lagrange coefficients $L_i := \prod_{j \in S, j \neq i} j/(j - i)$, which are then combined with the shares to recover the secret: $s = f(0) = \sum_{i \in S} s_i L_i$.*

We restrict to prime fields for simplicity and because all our instantiations work in prime-order groups. Shamir's scheme works over any finite field with more than n elements.

A.2 Feldman VSS

In Feldman's scheme [21], given (n, t) and a generator g of a cyclic group $\mathbb{G}$ of prime order $q > n$, to share a *high-entropy* secret $f_0 \in \mathbb{Z}_q$, the dealer acts as follows. It first does Shamir secret sharing using a secret polynomial $f(X) := f_0 + a_1 X + \cdots + a_t X^t$, and privately sends the shares $s_i := f_i = f(i)$ to each party P_i. Then, it publishes commitments $c_0 = g^{f_0}, c_1 = g^{a_1}, \ldots, c_t = g^{a_t}$ as the proof π_{share}. The proof π_{share} allow the parties to verify the validity of their shares in the sharing phase, as well as the published shares during the reconstruction phase.

To verify the share f_i, given, $\pi_{share} := (c_0, c_1, \ldots, c_t)$, party P_i checks if

$$g^{f_i} = \prod_{j=0}^{t} c_j^{i^j}, \tag{4}$$

and outputs `true` or `false`. For $n \geq 2t + 1$, if *all* n parties return `true`, then the final Verification returns `true`. Otherwise, any conflict between the dealer and the parties is resolved using a well-known conflict resolution approach, as employed in earlier VSS schemes such as Feldman and Pedersen.

A.3 Traceable Secret Sharing

As discussed in Section 1, Goyal et al. [30] introduced the concept of Traceable Secret Sharing (TSS), which enables traceability to identify the shareholders who participate in the secret reconstruction phase. Below, we recall the formal definition of TSS by synthesizing definitions from existing TSS schemes in the literature [6,30] and slightly modify them.

Definition 9 (Traceable Secret Sharing (TSS) schemes). *An (n, t)-TSS consists of five PPT algorithms of (Initial, Share, Reconstruct, Trace, Judge) defined as follows:*

- $\mathsf{Initial} \rightarrow (\mathsf{pp})$*: The initialization phase generates the set of common public parameters,* pp*.*
- $\mathsf{Share}(1^\lambda, n, t, s) \rightarrow (\{s_i\}_{i=1}^n)$*: Given the security parameter 1^λ, the secret s, and integers (n, t), the algorithm shares the secret s and obtains the shares $s_1, \cdots, s_n$.*
- $\mathsf{Reconstruct}(\{s_i\}_{i \in Q}) \rightarrow (s/\perp)$*: Given a set of shares s_i for shareholders $i \in Q$, it returns the secret s if $|Q| > t$, otherwise it outputs $\perp$.*
- $\mathsf{Trace}^{\mathsf{Rec}}(\{F(s_j)\}_{j \in V}, F(s), tk) \rightarrow ((I, \pi_{trace})/\perp)$*: Given a function of main secret obtained from the* Reconstruct *algorithm and shares, and an optional (public or private) tracing key tk, this algorithm has access to a reconstruction oracle* Rec*. It then outputs either a subset $I \subset [n]$ along with an associated proof π_{trace} or $\perp$.*
- $\mathsf{Judge}(vk, I, \pi_{trace}) \rightarrow$ `true/false`*: It is a deterministic algorithm that, given a verification key vk, a subset I, and an associated proof π_{trace}, outputs either* `true`*, indicating acceptance of the proof that the parties in I are malicious, or* `false`*, indicating rejection.*

A.4 Sigma Protocols and Non-Interactive Zero-Knowledge Proofs

Given a security parameter λ, let $X = X(\lambda)$ and $W = W(\lambda)$ be two sets and $\mathbf{R}$ be a relation on $X \times W$ that defines the following language,

$$Ł = \{x \in X : \exists\ \omega \in W,\ \mathbf{R}(x, \omega) = 1\}.$$

For an element $x \in Ł$, a corresponding element $\omega \in W$ is referred to as a witness if $\mathbf{R}(x, \omega) = 1$. Let $\mathcal{R}(\lambda)$ be a PPT algorithm that generates pairs (x, ω) satisfying $\mathbf{R}$, i.e. $\mathbf{R}(x, \omega) = 1$.

A sigma-protocol (Σ-protocol) for the relation $\mathbf{R}$ is a 3-round interactive protocol between two PPT algorithms: a prover P and a verifier V. P holds a witness ω for $x \in Ł$ and V is given x. P first sends a value a (so called the commitment) to V, and then V answers with a challenge c , and finally P answers with z. V accepts or rejects the proof. The triple $\mathsf{trans} = (a, c, z)$ is called a transcript of the Σ-protocol. A Σ-protocol is supposed to satisfy *Completeness*, *Honest Verifier Zero-Knowledge* (HVZK), and *Special Soundness* defined below.

Definition 10 (Completeness). *A Σ-protocol Π_Σ with parties (P, V) is* complete for $\mathcal{R}$*, if for all $(x, \omega) \in \mathbf{R}$ we have,*

$$\Pr\left[\mathsf{trans}(\mathsf{P}(\mathbf{R}, x, \omega) \leftrightarrow \mathsf{V}(\mathbf{R}, x)) \text{ is accepted by } \mathsf{V}\right] = 1\ ,$$

where trans *denotes the transcript of the protocol.*

Definition 11 (Honest Verifier Zero-Knowledge). *A Σ-protocol Π_Σ satisfies* HVZK for $\mathcal{R}$*, if there exists a PPT algorithm $\mathcal{S}$ that given $x \in X$, can simulate the* trans *of the scheme, s.t. for all $x \in Ł$, $(x, \omega) \in \mathbf{R}$ we have,*

$$\mathsf{trans}(\mathsf{P}(\mathbf{R}, x, \omega) \leftrightarrow \mathsf{V}(\mathbf{R}, x)) \approx \mathsf{trans}(\mathcal{S}(\mathbf{R}, x) \leftrightarrow V(\mathbf{R}, x))$$

where $\mathsf{trans}(\mathsf{P}(\cdot) \leftrightarrow \mathsf{V}(\cdot))$ *indicates the transcript of Π_Σ with (P, V), and $\approx$ denotes the indistinguishability of transcripts.*

Definition 12 (Special Soundness). *A Σ-protocol Π_Σ with parties (P, V) is* special sound for $\mathcal{R}$*, if there exists a PPT extractor* Ext*, such that for any $x \in Ł$, given two valid transcripts (a, c, z) and (a, c', z') for the same message, a, but $c \neq c'$, then $\mathsf{Ext}(a, c, z, c', z')$ outputs a witness ω for the relation $\mathbf{R}$.*

Public-coin Σ-Protocols. A sigma protocol Π_Σ is called *public-coin* if the challenge value c is a public and random string rather than something derived in a complex or private way.

Non-Interactive Zero-Knowledge (NIZK) Proof Systems. As previously noted, zero-knowledge proofs are interactive protocols, while Non-Interactive Zero-Knowledge (NIZK) proofs [4,27] eliminate the need for interaction between the prover and the verifier. A NIZK for the relation $\mathbf{R}$ consists of three PPT algorithms $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$, that are defined as follows.

- $\mathsf{Setup}(1^\lambda) \to pp$: Given the security parameter λ in its unary representation, outputs public parameters pp.
- $\mathsf{P}(pp, x, w) \to \pi$: Given the public parameters pp, a statement x, and a witness w, it generates a non-interactive proof π.
- $\mathsf{V}(pp, x, \pi) \to \texttt{false}/\texttt{true}$: Given the public parameters pp, a statement x, and a proof π, it outputs either `true` (accept) or `false` (reject).

A NIZK proof scheme must satisfy Completeness, Soundness, and Zero-Knowledge (ZK), as defined below. The definition of completeness for NIZK is analogous to the completeness property of a Σ-protocol (as defined in Def. 10), which can also be expressed as follows:

Definition 13 (Completeness). *A NIZK proof system, Π_{NIZK}, satisfies completeness if for all $(x, w) \in \mathbf{R}$, honestly generated proofs are always accepted by the verifier, i.e.,*

$$\forall (x, w) \in \mathbf{R}, \quad \Pr\left[\mathsf{V}(pp, x, \mathsf{P}(pp, x, w)) = \texttt{true}\right] = 1.$$

Definition 14 (Zero-Knowledge (ZK)). *A NIZK proof system Π_{NIZK} satisfies zero-knowledge if there exists a PPT simulator $\mathcal{S}$ such that for all $x \in L$, where L is the language corresponding to relation $\mathbf{R}$, the simulator can generate a proof that is indistinguishable from the real proof:*

$$\forall\ \mathsf{PPT}\,\mathcal{A}, \exists\,\mathcal{S}, \quad \{\mathcal{S}(pp, x)\} \approx \{\mathsf{P}(pp, x, w)\}\,.$$

Note that, Zero-Knowledge (ZK) is an amplified version of HVZK, defined in Def. 11, as it guarantees privacy against even a malicious verifier.

Definition 15 (Soundness). *A NIZK proof system, Π_{NIZK}, satisfies soundness if no PPT adversary $\mathcal{A}$ can generate a valid proof for an invalid statement $x \notin L$ with more than negligible probability. More formally, for all PPT adversaries $\mathcal{A}$ we have,*

$$\Pr\left[\mathsf{Setup}(1^\lambda) \to pp,\ \mathcal{A}(pp, x, w) \to \pi : \mathsf{V}(pp, x, \pi) = \texttt{true}\ \wedge x \notin L\right] \leq \mathsf{negl}(\lambda).$$

NIZK Proof of Knowledge. In addition to the standard soundness property, a NIZK proof system can satisfy a stronger notion known as *knowledge soundness*. When a NIZK proof system satisfies knowledge soundness, it is referred to as a NIZK proof of knowledge. Unlike standard soundness, which merely ensures that a prover cannot convince the verifier of a false statement, knowledge soundness guarantees that if a prover can generate a valid proof for a statement, then there exists an efficient extractor that can extract the corresponding witness from the prover. This strengthens the proof system by ensuring not only that a valid proof implies the truth of the statement, but also that the prover actually possesses the knowledge (i.e., the witness) required to construct the proof.

Definition 16 (Knowledge Soundness). *A NIZK proof system Π_{NIZK} satisfies knowledge soundness if for any PPT adversary $\mathcal{A}$ that can produce a valid*

proof for some statement x, there exists an efficient extractor, $\mathsf{Ext}_{\mathcal{A}}$, that can extract the corresponding witness w. Formally, for all PPT adversaries $\mathcal{A}$, if:

$$\Pr\begin{bmatrix}\mathsf{Setup}(1^\lambda)\to pp,\ \mathcal{A}(pp,x)\to\pi,\ \mathsf{Ext}_{\mathcal{A}}(pp,x,\pi)\to w:\\ \mathsf{V}(pp,x,\pi)=\mathtt{true}\ \wedge(x,w)\notin\mathbf{R}\end{bmatrix}\leq\mathsf{negl}(\lambda)\ .$$

Fiat-Shamir Transform. The *Fiat-Shamir* transform [22] enables the conversion of a secure public-coin Σ-protocol for a relation $\mathbf{R}$ into a NIZK proof scheme for the same relation. This transformation is achieved by replacing the challenge issued by the verifier with a challenge sampled from a random oracle, thereby eliminating the interaction between the prover and the verifier. In a public-coin Σ-protocol, the verifier sends a random challenge c to the prover in the second round. The Fiat-Shamir Transform substitutes this interactive challenge with a hash function $H(\cdot)$, modeled as a random oracle, that takes the public parameters, the statement being proved, and the prover's previous messages as inputs. Specifically, the challenge is computed as $c = H(pp, x, a)$, where pp represents the public parameters, x is the statement, and a is the prover's initial commitment. In the random oracle model, this transformation ensures that the challenge is both unpredictable and uniformly random, thereby preserving the security properties of the original protocol and achieving ZK instead of HVZK.

A.5 Public Key Cryptosystems and ElGamal's Scheme

Public-Key Encryption (PKE) is a fundamental cryptographic technique that allows a user to encrypt a message using a recipient's public key, ensuring confidentiality. The corresponding secret key, which is held securely by the recipient, is used to decrypt the ciphertext and recover the original plaintext.

Definition 17 (PKE Schemes). *For a given λ, message space M, and ciphertext space C, a PKE scheme is defined by the following PPT algorithms:*

- $\mathsf{KeyGen}(1^\lambda)\to(\mathsf{pk},\mathsf{sk})$*: Given the security parameter λ in its unary representation, it outputs the public/secret key pair* $(\mathsf{pk},\mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk},m)\to\mathsf{ct}$*: Given a public key* pk*, a message $m\in M$, the encryption algorithm outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk},\mathsf{ct})\to m/\perp$*: Given the secret key* sk*, a ciphertext $\mathsf{ct}\in C$, it returns either the message m or a failure symbol $\perp$ if the ciphertext is invalid.*

A PKE scheme must satisfy the following security properties.

Definition 18 (Correctness). *A PKE scheme is called correct if we have,*

$$\Pr\left[\forall\ \lambda, m\in M, (\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KGen}(1^\lambda):\mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\mathsf{pk},m))=m\right]=1\ .$$

Definition 19 (IND-CPA **Security).** *A PKE scheme is said to be IND-CPA secure if for all PPT adversaries $\mathcal{A}$ we have,*

$$\Pr\begin{bmatrix}(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KGen}(1^\lambda),(m_0,m_1,\mathsf{st})\leftarrow\mathcal{A}(\mathsf{pk}),b\leftarrow\{0,1\}\\ \mathsf{ct}_b\leftarrow\mathsf{Enc}(\mathsf{pk},m_b),b'\leftarrow\mathcal{A}(\mathsf{st},\mathsf{pk},\mathsf{ct}_b):b'=b\wedge m_0\neq m_1\end{bmatrix}\leq\frac{1}{2}+\mathsf{negl}(\lambda).$$

ElGamal Cryptosystem. ElGamal cryptosystem (a.k.a., encryption) [18] is a well-known PKE scheme, defined over a cyclic group $\mathbb{G}$ of prime order p with generator g. The scheme operates as follows:

- $\mathsf{KeyGen}(1^\lambda)$: Choose a secret key $\mathsf{sk} \leftarrow \mathbb{Z}_p^*$ and compute the public key as $\mathsf{pk} = g^{\mathsf{sk}}$. Output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: To encrypt a message $m \in \mathbb{G}$ using the public key pk, pick a random $r \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute and output the ciphertext $\mathsf{ct} := (c_1, c_2) = (m \cdot \mathsf{pk}^r, g^r)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: To decrypt a ciphertext $\mathsf{ct} = (c_1, c_2)$ using the secret key sk, compute and output: $m = c_1 / c_2^{\mathsf{sk}}$.

The correctness of ElGamal encryption is trivial and its IND-CPA security can be proved based on Decisional Diffie-Hellman (DDH) problem.

A.6 Commitment Schemes

Definition 20 (Commitment Scheme). *A commitment scheme, $\mathcal{C}$, consists of the following PPT algorithms:*

- $\mathsf{Initial}(1^\lambda) \rightarrow pp_c$*: The initial algorithm outputs the set of public parameters pp.*
- $\mathsf{Com}(pp, m, \tau) \rightarrow \mathsf{cm}$*: The commitment algorithm takes as input pp and a message m along with a trapdoor τ, and outputs a commitment* cm*.*
- $\mathsf{Verify}(pp, \mathsf{cm}, m', \tau') \rightarrow$ `true`/`false`*: Verification is a deterministic algorithm which takes as input pp, a commitment* cm*, a message m', along with an opening value τ', and outputs either* `true` *(accept) or* `false` *(reject).*

Informally, the main security properties required for a commitment scheme are correctness, hiding, and binding. Correctness means that commitments generated correctly will pass the verification phase. The hiding property ensures that the commitment hides all information about the committed value. Meanwhile, binding guarantees that once a commitment is made, the committer cannot open it to reveal two different messages.

Pedersen Commitment Scheme [34]. Over a cyclic group $\mathbb{G}$ of prime order p with generator g, the Pedersen commitment scheme allows to commit to a scalar message $m \in \mathbb{Z}_p$ and is perfectly hiding and computationally binding. It consists of the following polynomial-time algorithms:

- $\mathsf{Initial}(1^\lambda)$: Sample $r \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $h = g^r$. Output $pp := (g, h)$.
- $\mathsf{Com}(pp, m, \tau)$: Compute $\mathsf{cm} \leftarrow g^\tau h^m$. Output cm.
- $\mathsf{CVerify}(pp, \mathsf{cm}, m', \tau')$: Compute $\mathsf{cm}' \leftarrow g^{\tau'} h^{m'}$. Return $\mathsf{cm} == \mathsf{cm}'$.

A.7 Distributed Key Generation Protocols

Distributed Key Generation (DKG) protocols [34] allow multiple parties to collaboratively and securely generate key pairs without relying on a trusted third party. To carry out any operation involving the secret key, a sufficiently large subset of participants must cooperate, while any smaller group cannot gain any knowledge of the secret key. These generated keys can be used in various applications, particularly in threshold cryptosystems, such as threshold signatures and threshold encryption.

Definition 21 (Distributed Key Generation (DKG) [34]). *More formally, an (n,t)-DKG is an interactive protocol among a set of parties $(P_1,\ldots,P_n)$ that generates a tuple of public keys $(\mathsf{pk},\mathsf{pk}_1,\ldots,\mathsf{pk}_n)$ and a corresponding tuple of secret key shares $(\mathsf{sk}_1,\ldots,\mathsf{sk}_n)$ such that only party P_i knows the share sk_i.*

- *Consistency: A DKG is referred to as (t)-Consistent if, by the end of the protocol, all honest parties agree on a consistent public key pk and the vector of public keys $(\mathsf{pk}_1,\ldots,\mathsf{pk}_n)$, even when up to t parties are corrupted.*
- *Security: Additionally, as long as fewer than $t+1$ parties are corrupted, there exists a polynomial $f(x) \in \mathbb{Z}_p[X]_t$ of degree $d \leq t$ such that $\forall i \in [n]: \ \mathsf{sk}_i = f(i)$. In case of DKG for discrete logarithm, the global public key can then be computed as $\mathsf{pk} = g^{f(0)}$, where g is the generator of a cyclic group $\mathbb{G}$.*

B Acc-VSS from Pedersen VSS

The Feldman-based Acc-VSS of Section 4.1 requires a high-entropy secret and achieves *unpredictability* (Def. 5, the weaker variant). For low-entropy secrets, or when statistical secrecy is required, we instantiate the generic transformation on Pedersen VSS instead.

Pedersen VSS. In [34], Pedersen introduced his well-known commitment scheme and subsequently proposed a variant of the Feldman VSS scheme that can function with a *low-entropy* secret f_0 as well and achieve information-theoretic secrecy. To share a given secret f_0, Pedersen's VSS follows a similar approach to Feldman's scheme, where the dealer commits to the coefficients of the secret polynomial $f(X) := f_0 + a_1X + \cdots + a_tX^t$ using a discrete logarithm-based homomorphic commitment. However, unlike Feldman's scheme, in this case the dealer uses Pedersen commitment. More precisely, the dealer additionally samples another secret polynomial $r(X) := r_0 + b_1X + \cdots + b_tX^t$ and privately sends the shares $s_i := (f_i, r_i) = (f(i), r(i))$ to each party P_i. Then, given two random group generators (g,h), the dealer publishes commitments $c_0 = g^{f_0}h^{r_0}, c_1 = g^{a_1}h^{b_1}, \ldots, c_t = g^{a_t}h^{b_t}$ as the proof π_{share}. Similar to Feldman VSS, this proof π_{share} enables the parties to verify the validity of their shares during the sharing phase and to validate the published shares during the reconstruction phase. To verify the share $s_i := (f_i, r_i)$, given $\pi_{share} := (c_0, c_1, \ldots, c_t)$,

party P_i checks if

$$g^{f_i} h^{r_i} = \prod_{j=0}^{t} c_j^{i^j}, \tag{5}$$

and outputs `true` or `false`, similar to the verification procedure in Feldman VSS. Compared to Feldman's scheme, Pedersen's VSS doubles the size of individual shares, but it allows the protocol to operate with low-entropy secrets and provides a stronger security guarantee. During the reconstruction phase, each party P_j must publish their share $s_j := (f_j, r_j)$. These published shares are then verified using π_{share} and Eq. (5) and only the valid shares are used to reconstruct the secret f_0.

Construction. We instantiate the generic transformation of Figure 1 on Pedersen VSS, with the same identity-binding setup as in Section 4.1: γ_i is P_i's long-term identity key, with public identity $\mathsf{vid}_i = g^{\gamma_i}$ registered once. Each P_i stores $s_i := (f_i, r_i, \gamma_i)$, where (f_i, r_i) are the Pedersen share components. Verification of the dealer's (f_i, r_i) uses the standard Pedersen check (5); the audit-side check is the membership of vid_i in the registry. Conflict resolution proceeds as in Pedersen VSS: a complaint by P_i is resolved by the dealer broadcasting (f_i, r_i) for public verification.

In the $\mathsf{Reconstruct}$ phase, each P_i outputs $z_i = F(f_i)$ together with a NIZK proof π_i for the relation

$$\begin{aligned} R_i^{(2)} = \big\{ (g, h, F(\cdot), z_i, y_i, \mathsf{vid}_i, (f_i, r_i, \gamma_i)) \;\big|\; \\ z_i = F(f_i) \;\wedge\; y_i = g^{f_i} h^{r_i} \;\wedge\; \mathsf{vid}_i = g^{\gamma_i} \big\}, \end{aligned} \tag{6}$$

where $y_i = g^{f_i} h^{r_i} = \prod_{j=0}^{t} c_j^{i^j}$ can be publicly (pre)computed using π_{share} and the public index i. As in the Feldman-based $\mathsf{Acc\text{-}VSS}$ of Figure 2, depending on F, the NIZK system $\Pi_{\mathsf{NIZK}}^{(2)}$ can be a Schnorr-style Σ-protocol of conjunction (with Fiat–Shamir), a zk-SNARK, or any system with knowledge soundness for $R_i^{(2)}$. If P_i reveals plain share values, π_i collapses to (f_i, r_i, γ_i) and is checked directly against ak and vid_i. The collective audit proof π_{rec} aggregates the individual proofs and certifies that the reconstruction was carried out by an attributable set of shareholders. Figure 5 outlines the Pedersen-based $\mathsf{Acc\text{-}VSS}$. Its security is established below.

Theorem 5 (Acc-VSS from Pedersen VSS). *Let $\Pi_{\mathsf{NIZK}}^{(2)} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a NIZK argument for the relation $R_i^{(2)}$ of (6) satisfying completeness, zero-knowledge, and knowledge soundness (Appendix A.4). Under the discrete-logarithm assumption, the protocol of Figure 5 is a secure $\mathsf{Acc\text{-}VSS}$ with statistical secrecy: it satisfies correctness, verifiability, statistical secrecy, and accountability (both clauses (A1) single-query attribution and (A2) non-imputability) as defined in Definitions 3 to 6.*

Proof. The proof is given in Appendix E.5. □

Initial: Let $\mathbb{G}$ be a group of prime order $q > n$ with hard DL, and two random generators (g, h). The group generators (g, h) and all necessary parameters for the NIZK argument $\Pi^{(2)}_{\mathsf{NIZK}}$ are shared with all the parties.

Share: Given (n, t), and a secret $s := f_0 \in \mathbb{Z}_q$ the protocol proceeds as follows:

- Given secret f_0, the dealer samples a random secret $r_0 \in \mathbb{Z}_q$ and a two random degree-t polynomials $f(x) = f_0 + a_1 X + \cdots + a_t X^t$ and $r(x) = r_0 + b_1 X + \cdots + b_t X^t$, with $f(0) = f_0$, $r(0) = r_0$ and random coefficients $\{a_j, b_j\}_{j=1}^{t}$ from $\mathbb{Z}_q$. Next, the dealer computes $f_i := f(i)$ and $r_i := r(i)$ for $i = 1, \ldots, n$ and privately sends (f_i, r_i) to party P_i. Then, the dealer computes commitments $c_0 = g^{f_0} h^{r_0}, c_1 = g^{a_1} h^{b_1}, \ldots, c_t = g^{a_t} h^{b_t}$ and publishes a proof $\pi_D := (c_0, \ldots, c_t)$.
- In the same round, for $i \in [n]$, each party P_i uses its long-term identity key $\gamma_i \in \mathbb{Z}_q$, sampled once and registered as $\mathsf{vid}_i := g^{\gamma_i}$ in the one-time registration phase of Section 3.3, and broadcasts $c'_i := \mathsf{vid}_i$.
- At the end, each party P_i stores its individual share $s_i := (f_i, r_i, \gamma_i)$ and the public proof and the audit/verification keys $\mathsf{ak} = vk := (\pi_D, c'_1, \ldots, c'_n) = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$.

Verify: Given $\pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, and the shares, i.e., $s_i := (f_i, r_i, \gamma_i)$ for $i = 1, \ldots, n$, the parties proceed as follows:

1) Each party P_i checks if $g^{f_i} h^{r_i} = \prod_{j=0}^{t} c_j^{i^j}$, and broadcasts a complaint against the dealer if the verification failed. Party P_i also check if $c'_k \in \mathbb{G}$ for $k = 1, \ldots, n$, and $k \neq i$. The rest of this step and step 2 are the same as in Figure 2.
3) In case a shareholder P_i complains against the dealer, the dealer will broadcast (f_i, r_i) to enable everyone to verify it using the verification equation from Eq. (5). The rest of this step is the same as in Figure 2.

Reconstruct: Given $F(\cdot)$, which can be the identity function as well, the proof $\pi_{share} := (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, and the shares $\{s_i := (f_i, r_i, \gamma_i)\}_{i \in Q}$, the parties act as below:

- Each party P_i, for $i \in Q$, publishes $(z_i = F(f_i), \pi_i)$, where π_i is a non-interactive argument generated by $\Pi^{(2)}_{\mathsf{NIZK}}$ for the relation $R^{(2)}_i$ in equation (6). The proof ensures that z_i is computed correctly and P_i knows the original share $s_i := (f_i, r_i, \gamma_i)$. If $F(\cdot)$ is the identity, $z_i = f_i$ is public, but γ_i is never revealed: π_i only proves knowledge of it.
- Given π_{share}, the published values $\{z_i := F(f_i), \pi_i\}_{i \in Q}$ are verified using the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ and the valid ones are collected. Let $\{z_j = F(f_j), \pi_j)\}_{j \in V}$ be the set of valid pairs, V be the set of their indexes, and who be an n-bit string where encodes the indexes in set V and each of its bit indicates whether the corresponding party has provided a valid proof.
- If $|V| = 0$, the protocol returns $\perp$;
- If $t + 1 \leq |V|$: the protocol reconstructs $z := F(f_0)$ and generates associated proof $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j \in V}$, to show that z is computed correctly the the parties indicated by who and they know $\{s_j\}_{j \in V}$. Finally, it returns $(z = F(f_0), \pi_{rec} = \{\pi_j, \mathsf{who}\}_{j \in V})$. If $F(\cdot)$ is the identity function, the published $z_j = f_j$ are the plain shares, while $\{\pi_j\}_{j \in V}$ still only prove knowledge of each γ_j.
- If $0 < |V| < t + 1$: the protocol returns $\{z_j = F(f_j)\}_{j \in V}$ and a proof $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j \in V}$ to show that $\{z_j\}_{j \in V}$ are computed correctly by the parties indicated by who and they know $\{s_j\}_{j \in V}$.

VerifAudit: Given $\{z := F(f_0), \pi_{rec}\}$ or $\{\{z_j := F(f_j)\}_{j \in V}, \pi_{rec}\}$, generated by Reconstruct, and audit key $\mathsf{ak} := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, it computes $y_j := \prod_{k=0}^{t} c_k^{j^k}$ for $j \in V$ (which can be deduced from who) and then verifies the proofs $\pi_{rec} := \{\pi_j\}_{j \in V}$ using the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ and the commitment values $\{y_j, c'_j\}_{j \in V}$ and determines $I \subseteq V$ as the set of valid ones. Finally, it returns $1, I$ and $\pi_{\mathsf{Audit}} := (z = F(f_0)$ or $\{z_j = F(f_j)\}_{j \in I}, \{\pi_j\}_{j \in I})$ if $|I| > 0$; otherwise, it returns $\{0, \emptyset, 0^n\}$.

Judge: Given an alleged set I, and a proof $\pi_{\mathsf{Audit}} := (z = F(f_0)$ or $\{z_j = F(f_j)\}_{j \in I}, \{\pi_j\}_{j \in I})$, and $vk := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, using the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$, it verifies if π_{Audit} is valid and outputs `true` or `false`.

Fig. 5. The Pedersen-based Acc-VSS of Appendix B.

Efficiency. The Pedersen-based Acc-VSS has efficiency nearly identical to the Feldman-based one and adds minimal overhead over Pedersen VSS itself. The dealer's work in Π_{Share} is unchanged. Per-party shares are (f_i, r_i, γ_i), i.e. three field elements, one more than a Pedersen share (f_i, r_i), with γ_i reused across

instances under the identity-binding of Section 3.3. Per-share verification runs Pedersen's check (5) ($t + 1$ small exponentiations) plus a registry lookup of vid_i. Reconstruct, VerifAudit, and Judge run the NIZK system $\Pi^{(2)}_{\mathsf{NIZK}}$ with cost linear in $|I|$ and *publicly* executable, with no trusted tracer. In Section 5 we use the constructions of Figures 2 and 5 as building blocks for three concrete Acc-TC protocols: accountable threshold ElGamal decryption, accountable threshold Schnorr signing, and accountable distributed key generation.

C Deferred material

C.1 Algorithms vs. protocols

Earlier formulations of traceable secret-sharing schemes (including [6, 30]) describe the *whole* primitive as a tuple of "algorithms." This is convenient typographically, but for an honest-majority VSS-based scheme it is technically inaccurate: the share-distribution phase and the verification phase necessarily involve broadcast and complaint rounds among the parties, and treating them as single-shot algorithms hides the conflict-resolution structure that determines which parties end up qualified.

Throughout the paper we therefore decompose an Acc-VSS into a tuple $\Gamma = (\mathsf{Setup}, \Pi_{\mathsf{Share}}, \Pi_{\mathsf{Verify}}, \mathsf{Verify}_i, \mathsf{Reconstruct}, \mathsf{Combine}, \mathsf{VerifAudit}, \mathsf{Judge})$ in which:

- Setup is a one-time PPT *algorithm* that produces public parameters pp;
- Π_{Share} is an interactive *protocol* between the dealer D and the n shareholders that produces, when honestly executed, an individual share s_i for each party P_i together with a public audit-key ak and verification key vk;
- Π_{Verify} is an interactive *protocol* (with broadcast and complaints) that, on a public transcript and individual shares, returns a qualified set $Q \subseteq [n]$ or aborts;
- Reconstruct is a (per-party) PPT *algorithm* that, on a function $F(\cdot)$ and a party's individual share, produces a partial output and a NIZK proof of correctness, and Verify_i (local share check) and Combine (collection of valid contributions) are public algorithms;
- VerifAudit and Judge are deterministic public algorithms: the former verifies a collected reconstruction transcript and outputs the attribution set I and an audit proof π_{Audit}; the latter is the externally-runnable judge.

For brevity, we will sometimes refer to the entire scheme by its tuple of components and write $\Gamma.\mathsf{Share}$ instead of $\Gamma.\Pi_{\mathsf{Share}}$, but the protocol-vs.-algorithm distinction is operative in every security argument.

C.2 Design choices in the Acc-VSS syntax

We highlight two design choices of the syntax of Def. 2.

- Per-share verification Verify_i is exposed as a first-class primitive. This addresses the natural "who runs Verify?" question: any single party (or a public verifier observing the bulletin board) checks its own share locally; only when complaints are raised does the Π_{Verify} protocol coordinate a global decision.
- The audit-key ak *is* the verification key vk. We do not separate "tracing" and "verification" into two keys, since both are public; this also makes Judge executable by any external observer (an on-chain verifier, an e-voting auditor) without trusted-tracer setup.

C.3 Further discussion of the generic transformation

Discussion of the relation R_i^{gen}. The conjunction $\mathsf{Verify}'_i(\pi_D, f_i) = 1$ is what binds f_i to the *dealer's* commitments published in π_D; the conjunction $c'_i = \mathcal{C}(\gamma_i)$ is what binds γ_i to the *shareholder's* commitment published during Π_{Share}. Together, they ensure that a valid π_i commits the prover to two pieces of secret information that *no other party can produce*: the dealer-side share f_i and the shareholder-side audit secret γ_i.

What the proof of Theorem 1 does not *say.* Theorem 1 does not promise that an adversary holding $\{f_i\}_{i\in V}$ for some unauthorised V cannot reconstruct s, nor does it promise tracing of arbitrary reconstruction boxes built outside the protocol. The first is impossible in any threshold VSS by the very correctness of Shamir secret sharing; the second is impossible in the single-query model by Lemma 2.

C.4 Identity binding: γ_i as long-term key

In any $\mathsf{Acc\text{-}VSS}$ construction, a malicious P_i might try to sell f_i to a buyer without selling γ_i, hoping that the buyer can reconstruct s with $t+1$ such f_i's and use s in an application that does not enforce the audit check. We defeat this strategy at the construction level.

We move γ_i out of the per-instance Share protocol and into a one-time *public registration* phase. Concretely, before any $\mathsf{Acc\text{-}VSS}$ instance runs, every potential shareholder P_i executes:

- $\mathsf{Reg}_i \to (\mathsf{ik}_i, \mathsf{vid}_i)$: P_i samples $\gamma_i \leftarrow\!\!\$\ \mathbb{Z}_q$, sets $\mathsf{ik}_i = \gamma_i$ as its long-term identity key, and publishes $\mathsf{vid}_i := \mathcal{C}(\gamma_i)$ on a registration bulletin.

Subsequent $\mathsf{Acc\text{-}VSS}$ instances reuse $(\mathsf{ik}_i, \mathsf{vid}_i)$ unmodified: in Π_{Share}, the per-instance c'_i is simply set to $\mathsf{vid}_i = \mathcal{C}(\gamma_i)$ (the long-term identity commitment), and the relation R_i^{gen} requires that P_i proves knowledge of the unique γ_i behind vid_i.

Consequences.

- Selling f_i alone is useless across the application layer: a buyer who acquires f_i and reconstructs s cannot make the application accept any partial output, because producing π_i requires γ_i, which is bound to vid_i. To use the reconstructed secret in any Acc-VSS-fed application instance, the buyer must *also* obtain γ_i, the same γ_i across *all* of P_i's threshold deployments. Selling it once is selling all of P_i's identity at once.
- Cumulative auditability across instances: if P_i leaks γ_i, every audit trail that P_i ever participated in becomes derivable to anyone holding γ_i; the deterrent on P_i is therefore the cumulative reputation/financial value of vid_i, not the per-instance value.
- No additional setup for the application: the application layer's only new requirement is that it consume vid_i as a public party identity, exactly as it already does in PoS validator sets, federated-decryption committees, and DAOs.

C.5 Acc-VSS vs. VSS with authenticated contributions

The most natural way to obtain attribution for a threshold transcript is to take an actively secure VSS and have each shareholder sign every contribution (z_i, π_i) (together with a session identifier) under a long-term signing key $\mathsf{sk}_i^{\mathrm{sig}}$ whose public key is registered once. Public attribution then comes from signature verification, and non-imputability from EUF-CMA unforgeability. We do not claim that this baseline is insecure. Our point is that it is an *instance* of Acc-VSS, and that what Acc-VSS contributes lies elsewhere.

The baseline is an instance of the generic transformation. Take $\mathcal{C}(\gamma) = g^{\gamma}$ and let the binding for γ_i in R_i^{gen} be proved by a Schnorr signature on $(\mathsf{sid}, z_i, \pi_i^{\mathrm{VSS}})$ under $\mathsf{vid}_i = g^{\gamma_i}$, where π_i^{VSS} is the VSS's own proof of correct contribution. The result is exactly "VSS + signed contributions". In the random oracle model a Schnorr signature is a signature of knowledge of γ_i, so the pair $(\pi_i^{\mathrm{VSS}}, \sigma_i)$ is a knowledge-sound proof for R_i^{gen}, and Theorem 1 applies unchanged. For other signature schemes, the same argument goes through by replacing knowledge soundness with unforgeability in the (A1)/(A2) reductions.

What Acc-VSS adds. Given that the baseline is covered, the contribution of Acc-VSS is not a new mechanism that signatures cannot provide. It consists of the following:

- *A definition.* Actively secure threshold protocols attribute outputs to public keys only implicitly, and no prior VSS notion states what an auditor must be able to conclude from a transcript. Def. 6 makes this precise: complete attribution of every accepted transcript (A1) and protection against a dealer colluding with all other shareholders (A2).
- *A generic theorem.* Theorem 1 shows that *any* Shamir-based VSS combined with *any* knowledge-sound binding to a registered identity satisfies this definition, so each new instantiation (signature-based or not) does not need its own proof.

- *A necessity result.* Lemma 1 shows that, in any Shamir-based VSS, the binding secret *must* be independent of the dealer's polynomial. In particular, one cannot simply use the share f_i (or g^{f_i}) as the signing key, which is what an actively secure threshold protocol with per-party keys $\mathsf{pk}_i = g^{f_i}$ does.
- *Efficient conjunctive instantiations.* In our constructions, share correctness and the identity binding are proved in a single Σ-protocol with a shared challenge (Figure 3). This costs one extra group element and one extra scalar per contribution over the non-accountable Chaum–Pedersen proof, and needs no separate signing infrastructure.

In deployments where a signing-key infrastructure already exists, the signature-based instantiation is a perfectly reasonable choice, and our results apply to it directly.

C.6 Acc-TC: formal syntax and security requirements

Definition 22 (Accountable Threshold Cryptosystem (Acc-TC)). *An (n,t)-Acc-TC $\Pi_{\text{Acc-TC}}$ over message space M and ciphertext space C consists of the following PPT procedures:*

- $\mathsf{KeyGen}(1^\lambda, n, t) \to (\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n, \mathsf{vk}, \mathsf{tk})$*: Given the security parameter λ and integers (n,t), this algorithm generates a public key* pk*, individual secret shares $\{\mathsf{sk}_i\}_{i=1}^n$, and a pair of (public) verification and audit keys* $(\mathsf{vk}, \mathsf{tk})$*. As in our Acc-VSS instantiation,* $\mathsf{tk} = \mathsf{vk}$*.*
- $\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$*: Given a public key* pk *and a message $m \in M$ as inputs, this algorithm returns a ciphertext $\mathsf{ct} \in C$.*
- $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}) \to (d_i, \pi_i)$*: Given a secret key share* sk_i *and a ciphertext* ct*, the decryption algorithm returns a partial decryption d_i along with an associated proof π_i, which is generated to prove knowledge of s_i and the correctness of the partial decryption value d_i.*
- $\mathsf{Combine}(\{(d_i,\pi_i)\}_{i\in Q}, \mathsf{ct}, \mathsf{vk}) \to (\{m, \pi_{dec}\}/\{\{d_j\}_{j\in V}, \pi_{dec}\}/\perp)$*: Given the partial decryption values along with the associated proofs for a set of $Q \subseteq [n]$, i.e. $\{(d_i,\pi_i)\}_{i\in Q}$, the verification key* vk*, and the ciphertext* $\mathsf{ct} \in C$*, the algorithm first verifies the proofs $\{\pi_i\}_{i\in Q}$ based on $\{d_i\}_{i\in Q}$, and stores the set of valid pairs, i.e., $\{(d_i,\pi_i)\}_{i\in V\subseteq Q}$. From the set of valid pairs, it forms an n-bit string* who *where each bit indicates whether the corresponding party has provided a valid proof, exactly as in the Acc-VSS syntax of Def. 2. It then acts as follows:*
  - *If $|V| = 0$, it returns $\perp$.*
  - *If $|V| \geq t+1$, it uses $\{d_j\}_{j\in V}$ to decrypt the ciphertext* ct*, computes the message m, and generates an associated proof π_{dec}, and returns (m, π_{dec}). Additionally, the string* who *is included in the proof π_{dec}.*
  - *If $0 < |V| < t+1$, it returns the set $\{d_j\}_{j\in V}$ along with an associated proof π_{dec} for their validity. Similarly, the string* who *is included in π_{dec}.*
- $\mathsf{VerifAudit}\,((m,\pi_{dec})/(\{d_j\}_{j\in V},\pi_{dec}), \mathsf{ct}, \mathsf{ak}) \to (b, I, \pi_{\mathsf{Audit}})$*: given the* $\mathsf{Combine}$*output, the ciphertext, and the (public) audit key, this algorithm verifies π_{dec}, returns $(0, \emptyset, 0^n)$ on failure, and otherwise returns $(1, I, \pi_{\mathsf{Audit}})$ for the unique attribution set $I \subseteq V$ of valid contributors.*

1. $(Q, \mathsf{st}) \leftarrow \mathcal{A}\left(1^{\lambda}, n, t\right)$;
2. $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^{\lambda}, n, t)$;
3. $(m_0, m_1) \leftarrow \mathcal{A}\left(\mathsf{st}, \{\mathsf{sk}_i\}_{i\in Q}, \mathsf{pk}, \mathsf{vk}, \mathsf{tk}\right)$;
4. $b \leftarrow \{0,1\},\ \mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$;
5. $b' \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{pk}, \mathsf{vk}, \mathsf{ct}_b)$;
6. **return** $(b == b' \ \wedge\ |Q| < t+1)$;

Fig. 6. $\mathbf{Game}_{\mathcal{A}}^{\mathrm{CPA}}$: The IND-CPA security game.

- $\mathsf{Judge}(I, \pi_{\mathsf{Audit}}, \mathsf{vk}) \to \{\mathtt{true}, \mathtt{false}\}$*: a public deterministic algorithm that re-checks the audit proof against the attribution set.*

Security Requirements. An accountable threshold cryptosystem $\Pi_{\mathsf{Acc\text{-}TC}}$ must satisfy the following security properties:

Definition 23 (Correctness). *For a given integer $n > 1$ and $t < n$, a Acc-TC $\Pi_{\mathsf{Acc\text{-}TC}}$ is called correct, if we have:*

$$\Pr\left[\begin{array}{l}\forall m \in M, Q \subseteq [n], |Q| \geq t+1; (\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^{\lambda}, n, t); \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m); (m', \pi_{dec}) \leftarrow \mathsf{Combine}(\{\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})\}_{i\in Q}, \mathsf{ct}, \mathsf{vk}) : m' = m\end{array}\right] = 1$$

Additionally,

$$\Pr\left[\begin{array}{l}\forall m \in M, Q \subseteq [n], (\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^{\lambda}, n, t); \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m); \\ (1, Q', \pi_{\mathsf{Audit}}) \leftarrow \mathsf{VerifAudit}(\mathsf{Combine}\left(\{\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})\}_{i\in Q}, \mathsf{ct}, \mathsf{vk}\right), \mathsf{ct}, \mathsf{tk}) : \\ Q' = Q\end{array}\right] = 1$$

Definition 24 (IND-CPA Security). *Let the advantage of an adversary $\mathcal{A}$ against the IND-CPA security $\mathbf{Game}_{\mathcal{A}}^{CPA}$ (cf. Figure 6), be defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{CPA}(\lambda) = \Pr\left[\mathbf{Game}_{\mathcal{A}}^{CPA} = 1\right] \ .$$

A Acc-TC $\Pi_{\mathsf{Acc\text{-}TC}}$ is called IND-CPA secure if for all PPT adversaries $\mathcal{A}$, we have, $\mathsf{Adv}_{\mathcal{A}}^{CPA}(\lambda) \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Definition 25 (Public Auditability). *For $n > 1$ and $t < n$, a Acc-TC $\Pi_{\mathsf{Acc\text{-}TC}}$ satisfies public auditability if for all adversaries $\mathcal{A}$, and any set $Q \subseteq [n]$, we have:*

$$\Pr\left[\begin{array}{l}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^{\lambda}, n, t); \\ (\mathsf{ct}, T = (m, \pi_{dec})/(\{d_j\}_{j\in Q}, \pi_{dec})) \leftarrow \mathcal{A}(1^{\lambda}, n, t, \mathsf{pk}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i\in Q}); \\ (b, I, \pi_{\mathsf{Audit}}) \leftarrow \mathsf{VerifAudit}((m, \pi_{dec})/(\{d_j\}_{j\in Q}, \pi_{dec}), \mathsf{ct}, \mathsf{tk}) : \\ (b = 1 \wedge I = \mathsf{Contrib}(T) \wedge I \subseteq Q \wedge \mathsf{Judge}(I, \pi_{\mathsf{Audit}}, \mathsf{tk}) = \mathtt{true}) \vee (b = 0 \wedge \mathsf{Contrib}(T) = \emptyset)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda$$

Here $\mathsf{Contrib}(T)$ is the set of indices j for which T contains a partial decryption d_j with a proof π_j that verifies for (ct, d_j) under vk, exactly as in Def. 6.

Definition 26 (Non-imputability). *For any integers $n > 1$ and $t < n$, a Acc-TC $\Pi_{\mathsf{Acc\text{-}TC}}$ satisfies the non-imputability if for all adversaries $\mathcal{A}$ and ciphertext $\mathsf{ct} \in C$, we have:*

$$\Pr\left[\begin{array}{l}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^\lambda, n, t);\\ (1, I, \pi_{\mathsf{Audit}}) \leftarrow \mathcal{A}(1^\lambda, n, t, \mathsf{pk}, \mathsf{vk}, \mathsf{tk}, \{\mathsf{sk}_i\}_{i\in[n]\setminus i^*}, \mathsf{ct}) :\\ \mathsf{Judge}(I, \pi_{\mathsf{Audit}}, \mathsf{vk}) = \mathtt{true} \wedge i^* \in I\end{array}\right] \leq \mathsf{negl}(\lambda)\ .$$

C.7 Efficiency of $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$ and comparisons

Although $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$ introduces accountability, it adds minimal overhead over a non-accountable, actively secure, robust threshold ElGamal protocol. In the non-accountable case, parties still publish a NIZK proof for the correctness of decryption: each party uses the Chaum-Pedersen DL equality NIZK protocol [7] and generates a proof to show that $d_i = c^{f_i} \wedge \mathsf{pk}_i = g^{f_i}$. $\Pi^{Dec}_{\mathsf{NIZK}}$ is exactly this proof AND-composed with a Schnorr proof for $\mathsf{vid}_i = g^{\gamma_i}$ (relation (3)), so the only extra work is one group element and one scalar in the proof, one exponentiation for the prover, and two for the verifier.

More concretely, during key generation of $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$ the parties produce exactly the same $(pp, \mathsf{pk}, \mathsf{pk}_1 = g^{f_1}, \ldots, \mathsf{pk}_n = g^{f_n})$ as in the non-accountable protocol. The audit/verification key is $\mathsf{vk} = \mathsf{tk} = (pp, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \mathsf{vid}_1, \ldots, \mathsf{vid}_n)$, where the vid_i are the long-term identities registered once and reused across instances, and the encryptor still uses only (pp, pk).

The encryption phase operates identically to the standard ElGamal scheme. In the decryption phase, each party P_i computes a partial decryption d_i, similar to the standard ElGamal scheme. However, as in the non-accountable actively secure threshold decryption protocol based on ElGamal, each party also generates a proof π_i using $\Pi^{Dec}_{\mathsf{NIZK}}$, which, as mentioned earlier, has asymptotically the same cost as in the non-accountable case, and concretely adds a constant number of exponentiations in group $\mathbb{G}$ to each party's cost.

When combining the partial decryptions, as in the non-accountable actively secure case, the validity of the proofs $\{\pi_i\}_{i\in Q}$ is verified using the public keys and the verification algorithm for $\Pi^{Dec}_{\mathsf{NIZK}}$. This again has asymptotically the same cost as in the non-accountable case, and concretely requires $6|Q|$ exponentiations instead of $4|Q|$ (in Chaum-Pedersen DL equality proof [7]). Once valid proofs $|V| \geq t+1$ are collected, similar to threshold ElGamal, each party computes the Lagrange coefficients and performs $|V|$ exponentiations and $|V|+1$ multiplications to obtain the message m.

In our case, public auditability requires that the partial-output verification be performed even when $|V| < t+1$ (so that minority-set leakage is still attributable, even when the secret cannot be reconstructed). VerifAudit and Judge run only the verifier of $\Pi^{Dec}_{\mathsf{NIZK}}$, with cost at most $6|I|$ exponentiations. In our benchmarks (Appendix C.11) the resulting per-party overhead is $\approx 40\,\mu$s, i.e. 30–37% of the reconstruction phase and at most 20.3% end-to-end ($n = 20$).

C.8 Accountable threshold Schnorr/FROST

The same recipe applies to threshold Schnorr signing, including the FROST family of two-round threshold signatures [9,11,32]. Below we sketch the natural Acc-TC instantiation; a full treatment follows the same structure as Section 5.1.

The setup phase is identical to that of $\Pi^{\mathsf{EG}}_{\mathsf{Acc\text{-}TC}}$: a long-term joint public key $\mathsf{pk} = g^{f_0}$, individual shares $\mathsf{sk}_i = (f_i, \gamma_i)$, and public per-party identities $\mathsf{vid}_i = g^{\gamma_i}$ are produced by an instance of the Acc-DKG of Section 5.2. To sign a message μ:

- Each signer P_i samples a per-instance nonce k_i and broadcasts the FROST commitment D_i, E_i as in standard FROST. The aggregator computes the joint nonce R and challenge $e = H(R, \mathsf{pk}, \mu)$.
- Each signer outputs the partial signature $\sigma_i = k_i + e\,\lambda_i\,f_i \bmod q$, where λ_i is the standard Lagrange coefficient, together with a NIZK proof π_i^{Sch} for the relation

$$R_i^{\mathrm{Sch}} = \left\{ (g, R, \mathsf{pk}, e, \lambda_i, \sigma_i, \mathsf{vid}_i, (f_i, \gamma_i)) \;\middle|\; g^{\sigma_i} = R_i \cdot (g^{\lambda_i f_i})^e \;\wedge\; \mathsf{vid}_i = g^{\gamma_i} \right\},$$

  where $R_i = g^{k_i}$ is the per-party nonce contribution and the public statement $g^{\lambda_i f_i}$ is recoverable from the dealer's commitments.
- The aggregator combines the σ_i into the aggregate signature $\sigma = \sum_{i \in V} \sigma_i$ exactly as in FROST. The audit transcript is $\pi_{\mathrm{rec}}^{\mathrm{Sch}} = (\mathsf{who}, \{\pi_i^{\mathrm{Sch}}\}_{i \in V})$.
- VerifAudit and Judge are the standard NIZK verifier on each π_i^{Sch} together with a check against the registered vid_i.

The proof π_i^{Sch} is a Schnorr-style Σ-protocol of conjunction whose cost is comparable to the proof already used by [11] for adaptively-secure FROST. The accountability overhead is the per-party identity-binding component, which adds two exponentiations (one in the prover, one in the verifier). Without our public auditability layer, a corrupt validator that signs an unauthorised message can leak σ_i without leaving any record of who signed what. With the auditability layer, the slashing module in a proof-of-stake protocol can verify not only that an unauthorised signature was produced, but also which validators signed it, with a public proof that does not require trusted tracing.

C.9 Accountable Distributed Key Generation

Using a DKG protocol, the secret polynomial and the corresponding commitments to the shares are generated in a fully distributed manner by the parties themselves, rather than by a trusted authority. A standard approach, based on Feldman (or Pedersen) VSS, is described in [33]. Each party P_i acts as the dealer once, generating a polynomial $f^{(i)}$ of the correct degree, and privately sending the share $f^{(i)}(j)$ to party P_j for $j \in \{1, \ldots, n\} \setminus \{i\}$. Each party then computes its own share by summing all the shares received. This implicitly defines $f = \sum_{i=1}^{n} f^{(i)}$, with P_j's share $f(j) = \sum_{i=1}^{n} f^{(i)}(j)$, and the secret $f_0 = f(0) = \sum_{i=1}^{n} f^{(i)}(0)$ remains unknown to all players in the honest-majority setting.

Since some parties (up to t) may behave maliciously, all parties verify that the shares they receive originate from a polynomial of degree at most t. Once verification completes (using the Feldman or Pedersen VSS check), the non-disqualified parties compute the joint public key

$$\mathsf{pk} = g^{f_0} = g^{f(0)} = \prod_{i \in Q} g^{L_{0,i}^{Q} s_i},$$

where $L_{0,i}^{Q} = \prod_{j \in Q,\ j \neq i} \frac{j}{j-i}$ is the Lagrange coefficient and Q is the qualified set.

In our Acc-VSS-based Acc-DKG, each P_i runs the dealer steps of Figure 2 or Figure 5 in parallel using the long-term γ_i registered in advance, and the joint audit/verification key $\mathsf{vk} = \mathsf{tk}$ is assembled from the vid_i's and the dealer commitments. The audit transcript of every subsequent threshold operation (decryption, signing, partial reconstruction) attributes back to the vid_i's established in the Acc-DKG, so a slashing module that observes the audit transcript can identify and penalise validators that contributed to an unauthorised partial output, without any trusted tracer.

C.10 Other related lines

Traceable secret sharing for general access structures [19, 20, 28]. Farràs and Guiot [19] give a TSS for general access structures in a model where the access structure is hidden from the adversary; their scheme has share size and reconstruction time exponential in n. A follow-up [20] aims for information-theoretic strong tracing. Goyal and Waters [28] concurrently developed threshold traitor tracing for general access structures. None of these works address verifiability or threshold-protocol accountability; Acc-VSS complements them.

Accountable threshold signatures [6, 11, 32]. The signature variants of Boneh et al. and the FROST line of Komlo and Goldberg [32] and Crites et al. [10, 11] focus on the unforgeability and aggregation of threshold signatures, not on the attribution question. Our Acc-TC-Schnorr/FROST sketch in Appendix C.8 adds the attribution layer and enables on-chain slashing of corrupt signers in proof-of-stake deployments.

Verifiable secret sharing surveys [2, 3]. Recent unifications of VSS frameworks [2, 3] underpin the choice of Γ' in our generic transformation. Any of these can be plugged into Figure 1, yielding an Acc-VSS with the corresponding security profile (computational vs. statistical secrecy, plain DL vs. ROM-based hash commitments).

Comparison with secret sharing with snitching. Dziembowski et al. [17] study a different model in which all n shareholders may be corrupt. Their model is the *individual cryptography* model of [16]: hashes can be evaluated cheaply locally but are slow inside an MPC, and reconstruction by an unauthorised set is possible only by performing many MPC hash evaluations. They define *punishability* (any

successful illegal reconstruction yields a proof for an external judge) and *unframability* (no honest party can be falsely accused), and instantiate their scheme on top of an MPC-hard hash function plus a smart-contract judge. Crucially, their scheme captures *MPC attacks*, in which colluders compute the reconstruction inside an MPC and never produce a tangible reconstruction box; both TSS and Acc-VSS leave this outside their model.

Snitching is more powerful than Acc-VSS in two respects: (i) it tolerates a fully corrupt set of shareholders (instead of an honest majority), and (ii) it catches MPC attacks. It is also heavier: it relies on an explicit blockchain-based punishment infrastructure, on hashing-MPC-hardness assumptions, and on every shareholder being incentivised to snitch. Acc-VSS and snitching are essentially orthogonal:

- Acc-VSS is a property of the threshold protocol's *outputs*: it makes any accepted partial output single-query-attributable, regardless of the machinery the adversary used to compute it.
- Snitching is a property of the share-sharing layer: it makes any complete illegal reconstruction (even one performed in MPC) provably punishable, conditional on at least one colluder choosing to snitch.

The two can compose: a system that uses Acc-VSS as its threshold-cryptography layer and a snitching-style judge as its slashing layer would resist (i) all output-attribution attacks (by Acc-VSS), (ii) bribery of honest-majority openers (by Acc-VSS + on-chain enforcement), and (iii) MPC-style covert reconstructions (by snitching). We do not formalise this composition here.

C.11 Benchmark details

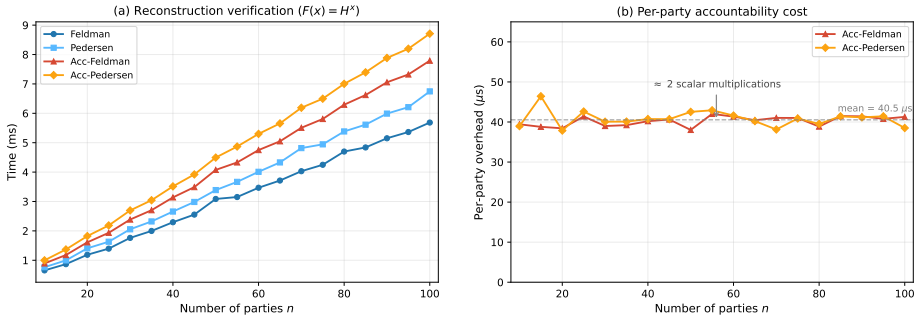


Fig. 7. Accountable threshold-decryption benchmarks on ristretto255 ($t{=}\lfloor n/2 \rfloor$, 100 iterations). (a) Reconstruction verification time for $F(x) = H^x$. (b) Per-party accountability overhead: a constant $\approx 40\,\mu$s ≈ 2 scalar multiplications, independent of n.

We implement the Feldman and Pedersen Acc-VSS of Section 4.1 and Appendix B together with their non-accountable baselines on the ristretto255 group

and benchmark on an Apple M4 processor. For each configuration (n, t) with $t = \lfloor n/2 \rfloor$ and $n \in \{10, 15, \ldots, 100\}$, we run 100 iterations and report the average time. Four phases are measured independently: (i) Share (dealer generates the degree-t polynomial, evaluates n shares, and computes the commitment vector); (ii) Verify (all n parties check their shares against the commitments); (iii) Reconstruct-id (t+1 parties reveal plain shares, with identity verification for Acc-VSS); (iv) Reconstruct-TDec (t+1 parties produce partial decryptions $d_i = c_2^{f_i}$ with Schnorr-style DLEQ proofs, extended to the Acc-DLEQ conjunction of (2) for Acc-VSS).

Table 2 reports representative numbers for $n \in \{20, 50, 100\}$. The Share and Verify columns confirm that these phases are cryptographically identical across the accountable and non-accountable variants: the Acc-VSS share-time increase is due entirely to the one-time identity registration (n exponentiations for $\mathsf{vid}_i = g^{\gamma_i}$), amortised across all future instances. The reconstruction overhead isolates cleanly. Each party's accountable contribution requires exactly two additional scalar multiplications for the identity clause of the NIZK conjunction ($g^{s_\gamma} \stackrel{?}{=} R_\gamma \cdot \mathsf{vid}_i^c$), yielding a constant per-party cost of $\approx 40\,\mu$s regardless of n (Figure 7(b)). Because Share and Verify dominate the total protocol time and are identical across variants, the end-to-end overhead (all three phases combined) drops from 20.3% (Acc-Feldman, n=20) to 5.9% (n=100) and from 17.1% (Acc-Pedersen, n=20) to 5.0% (n=100). Isolating the reconstruction phase alone, the overhead is $\approx$ 30–37%; this is the constant two-exponentiation cost divided by the roughly-constant per-party baseline cost, and it does not grow with n.

Table 2. Average per-phase times (μs) and total-protocol overhead for selected n (ristretto255, t=$\lfloor n/2 \rfloor$, 100 iterations). Recon = Reconstruct-TDec ($F(x) = H^x$, DLEQ/Acc-DLEQ verification + Lagrange combine). Overhead = total (Share+Verify+Recon) ratio of Acc-VSS over the baseline.

	$n = 20,\ t = 10$			$n = 50,\ t = 25$			$n = 100,\ t = 50$		
Scheme	Share	Verify	Recon	Share	Verify	Recon	Share	Verify	Recon
Feldman	211	4 233	1 187	579	24 855	3 086	1 298	93 717	5 686
Acc-Feldman	568	4 597	1 610	1 499	25 921	4 074	3 135	95 717	7 790
overhead		20.3%			10.4%			5.9%	
Pedersen	427	4 607	1 406	1 189	26 296	3 388	2 613	95 743	6 746
Acc-Pedersen	775	4 946	1 823	2 083	26 784	4 494	4 381	97 246	8 709
overhead		17.1%			8.1%			5.0%	

D Related Work, Separations, and Open Problems

This section places Acc-VSS and Acc-TC on the map of accountability primitives for threshold cryptography. We open with the comparison most likely to be raised

by readers, against a VSS whose contributions are signed under long-term keys, and then summarise the box-tracing line [5,6,29,30] and the snitching line [17]. Acc-VSS occupies the strongest single-query corner of this map among schemes that do not rely on iO: it is the only point that combines verifiability of the dealer with single-query, transcript-level accountability.

D.1 Acc-VSS vs. VSS with signed contributions

A natural alternative is an actively secure VSS in which each shareholder signs every contribution under a registered long-term key. This baseline is sound, and it is in fact an instance of our generic transformation: a Schnorr signature on the session-bound contribution is a signature of knowledge of the identity secret γ_i, so Theorem 1 covers it. The contribution of Acc-VSS is therefore not a mechanism that signatures cannot provide. It is (i) the definition (A1)+(A2) of what such a scheme must guarantee, (ii) a single generic proof covering every VSS and every knowledge-sound identity binding, (iii) Lemma 1, which shows that the binding secret must be independent of the dealer's polynomial, and (iv) conjunctive instantiations whose overhead over the non-accountable proof is one group element and one scalar. Appendix C.5 elaborates.

D.2 Comparison with the box-tracing line

Goyal et al. [30]. The original TSS scheme of Goyal et al. handles threshold structures and bases its tracing strategy on the Goldreich–Levin hardcore-bit [26]. The scheme assumes an honest dealer (the dealer engages in a two-party protocol with each shareholder), the share size grows quadratically in the secret, and the tracing algorithm queries the reconstruction box on inputs that depend on previous queries, a feature that implicitly assumes the box is stateless and answers many queries. A concrete adversary can ship a stateful box (or one that self-destructs after a single use), which collapses the guarantee. Acc-VSS is incomparable: we do not promise tracing of arbitrary boxes, but we guarantee single-query, transcript-level attribution of every reconstruction the application accepts, and we add verifiability of the share-distribution phase, neither of which is present in [30].

Boneh et al. [5, 6]. Boneh et al. reduced the share size to $O(|s|)$ for threshold structures and streamlined the tracing strategy: for Shamir secret sharing, it requires $2t$ black-box queries to the reconstruction box in the worst case (two per evaluation, for the t evaluations needed for tracing); the dealer and the tracer are separated but both still trusted, and the dealer is honest. Their tracing requires *invalid* shares to be queried into the box, which is incompatible with verifiability: a VSS-based scheme rejects invalid shares. The companion accountable-decryption paper [5] introduces a confirmation procedure that identifies decoders from their behaviour on adversarial ciphertexts; like the share-tracing version, it requires multiple queries against a decoder that, in any deployment, can be made stateful. Acc-TC, by contrast, embeds the accountability into each *partial*

output that the decryption protocol publishes, so a single corrupt decryption attempt is already attributable without subsequent querying.

Goyal et al. [29]. The recent revisitation identifies two shortcomings of all prior box-tracing definitions: (i) the box is assumed to expect minimal share inputs, and (ii) the tracing only handles boxes that output the full secret. They propose two strengthened notions: $\emptyset$-strong traceability and strong traceability. For $\emptyset$-strong traceability they give a construction for monotone-circuit access structures based on iO and one-way functions; for strong traceability they show the notion is incompatible with tracer non-imputability or tracer secrecy, and give a construction without those properties from one-way functions only. Acc-VSS does not target either of these notions; instead, our combinatorial Lemma 2 shows that the simpler single-query tracing of stateless boxes already fails. Section 3 discusses how the two impossibilities are related but independent.

D.3 What we do not solve

Tracing the post-reconstruction secret. Acc-VSS does not prevent a corrupt subset from reconstructing s from leaked f_i's and using s outside any Acc-VSS-mediated application. This is unavoidable for any threshold VSS: $t + 1$ shares determine s. Closing this gap requires either the multi-query box-tracing notion of [6,30], the iO-based notion of [29], or the snitching model of [17].

Post-quantum accountability. Our constructions instantiate the generic transformation on Feldman/Pedersen VSS in discrete-log groups. Composing the generic transformation with a lattice-based or hash-based VSS (e.g., the framework of [3]) plus a post-quantum NIZK should yield a post-quantum Acc-VSS; we leave a fully-vetted instantiation to future work.

Aggregating audit transcripts. The audit proof π_{rec} is currently a concatenation of $|I|$ per-party Schnorr-style proofs. Because the proofs share the same algebraic structure, a verifier can batch-verify all $|I|$ proofs via two multi-exponentiations of size $2|I|$ using the random-linear-combination technique [25], reducing the verification cost by a factor of roughly $\log |I|$ without changing the proof format. This does not reduce the proof *size*; compressing the audit transcript into a single succinct proof would require folding-style or Bulletproofs-style aggregation, which we leave to future work.

Beyond Shamir. A constructive recovery of full traceability via a smart-contract or trusted-third-party judge that observes accumulated Acc-VSS audit transcripts would bridge Acc-VSS with the snitching model of [17] and with the box-tracing line [6,29]. Extending Acc-VSS to non-Shamir secret-sharing schemes is a further natural direction.

E Security Proofs

E.1 Proof of Theorem 1

Proof. **Correctness** follows by inspection: when every party honestly executes Π'_{Share}, samples γ_i, broadcasts c'_i, and runs $\mathsf{Reconstruct}_i$, completeness of Γ', $\mathcal{C}$, and Π_{NIZK} ensures that every π_i verifies, $\mathsf{Combine}$ reconstructs the right $z = F(s)$, and $\mathsf{VerifAudit}$ accepts with attribution set Q.

Verifiability reduces to verifiability of Γ'. Π_{Verify} in Γ accepts iff Π'_{Verify} accepts and every c'_k is well-formed; if the former accepts, the underlying VSS guarantees a unique secret s that any qualified subset will reconstruct. The well-formedness of the c'_k does not enter the secret extraction.

Secrecy reduces to secrecy of Γ' alone; no hiding property of $\mathcal{C}$ is needed. Each γ_i is sampled by P_i independently of the dealer's polynomial f, so the commitments $c'_i = \mathcal{C}(\gamma_i)$ are independent of the secret. Given an adversary against Γ, a distinguisher for Γ' samples all γ_i itself, computes all c'_i honestly, and embeds its own challenge share-out of Γ'. The adversary's view is identical to its view in Γ, so the two advantages coincide (in the statistical case the reduction is unbounded as well).

Accountability, clause (A1) (single-query attribution). First, $I = \mathsf{Contrib}(T)$ whenever $b = 1$, and $b = 0$ only when $\mathsf{Contrib}(T) = \emptyset$. This holds with probability 1 by construction, because $\mathsf{VerifAudit}$ is deterministic and sets I to exactly the set of indices whose proofs verify. It remains to show $I \subseteq Q$. The reduction is to knowledge soundness of Π_{NIZK} and to one-wayness of $\mathcal{C}$. Suppose $\mathcal{A}$ controlling Q produces a transcript π_{rec} with $b = 1$, attribution I, and $i^\star \in I \setminus Q$. By validity of $\pi_{i^\star}$ and knowledge soundness, an extractor outputs a witness $(\hat{f}_{i^\star}, \hat{\gamma}_{i^\star})$ such that $\mathsf{Verify}'_{i^\star}(\pi_D, \hat{f}_{i^\star}) = 1$ and $\mathcal{C}(\hat{\gamma}_{i^\star}) = c'_{i^\star}$. Since $\hat{f}_{i^\star}$ may be computable by the coalition (Lemma 1), the contradiction must come from $\hat{\gamma}_{i^\star}$. The reduction receives a one-wayness challenge $C^\star = \mathcal{C}(\gamma)$ for uniform γ and embeds it as $c'_{i^\star}$. This embedding is perfect, because $\gamma_{i^\star}$ is uniform and independent of everything else, and $P_{i^\star}$ never reveals it (honest contributions of $P_{i^\star}$, if any, are simulated with the NIZK simulator). By perfect binding, $\hat{\gamma}_{i^\star}$ is a preimage of $C^\star$, which breaks one-wayness. Hence $I \subseteq Q$ except with negligible probability. The conjunction $\mathsf{Verify}'_{i^\star}(\pi_D, \hat{f}_{i^\star}) = 1$ ensures Judge also accepts on I.

Accountability, clause (A2) (non-imputability). The reduction is to one-wayness of $\mathcal{C}$ on $\gamma_{i^\star}$. The adversary holds all shares except $s_{i^\star}$, and in particular sees $c'_{i^\star}$ but not $\gamma_{i^\star}$. The reduction embeds a one-wayness challenge as $c'_{i^\star}$, exactly as in (A1). If $\mathcal{A}$ outputs $(1, I, \pi_{\mathsf{Audit}})$ with $i^\star \in I$ and Judge accepts, then π_{Audit} contains a valid $\pi_{i^\star}$, and knowledge soundness of Π_{NIZK} yields $\hat{\gamma}_{i^\star}$ with $\mathcal{C}(\hat{\gamma}_{i^\star}) = c'_{i^\star}$, contradicting one-wayness. This holds even though the adversary holds the shares of all other parties and can therefore interpolate $f_{i^\star}$ whenever $n-1 \geq t+1$. □

E.2 Proof of Theorem 2

Proof. **Correctness of the Reconstruction Algorithm.** By the correctness of the Feldman VSS, the secret value s is constructed with a probability 1.

Correctness of the Audit Algorithm. We show the correctness by analyzing two cases based on the size of Q:

- When $0 < |Q| \leq t$, the reconstruction algorithm returns $\{F(f_j), \pi_j, \mathsf{who}\}_{j\in Q}$ where π_j is a non-interactive proof (or argument) generated by $\Pi^{(1)}_{\mathsf{NIZK}}$ for the relation $R^{(1)}_j$ defined in equation (2). The audit algorithm given $\{F(f_j), \pi_j, \mathsf{who}\}_{j\in Q}$ and the audit key $\mathsf{ak} := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, computes $y_j := \prod_{k=0}^{t} c_k^{j^k}$ for $j \in Q$ (which can be deduced from the n-bit string who) and then verifies the proofs $\{\pi_j\}_{j\in Q}$ using the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ and commitment values $\{y_j, c'_j\}_{j\in Q}$. By the completeness property of $\Pi^{(1)}_{\mathsf{NIZK}}$, all proofs will be accepted. Finally, it returns Q and $\pi_{\mathsf{Audit}} := (\{F(f_j)\}_{j\in Q}, \{\pi_j\}_{j\in Q})$.
- When $|Q| \geq t+1$, the reconstruction algorithm returns $\{F(s), \pi_j, \mathsf{who}\}_{j\in Q}$ where π_j is a non-interactive proof (or argument) generated by $\Pi^{(1)}_{\mathsf{NIZK}}$ for the relation $R^{(1)}_j$ defined in equation (2). Similarly, by the completeness property of $\Pi^{(1)}_{\mathsf{NIZK}}$, all proofs $\{\pi_j\}_{j\in Q}$ will be accepted. Therefore, the audit algorithm returns Q and $\pi_{\mathsf{Audit}} := (\{F(s)\}_{j\in Q}, \{\pi_j\}_{j\in Q})$.

Verifiability. It holds directly by the verifiability of the Feldman VSS. (Note that, the only difference between the verification in Figure 2 and Feldman scheme is that the parties check if c'_j are broadcast and well-formed. Since Q contains honest parties, this extra check does not affect the output of verification algorithm in Figure 2.)

Unpredictability. The unpredictability of the protocol in Figure 2 holds clearly by the unpredictability of the Feldman VSS which is proven under the discrete logarithm assumption. The reason is that, for each i, the extra component of s_i, which is γ_i, is chosen by the party i randomly and, independently from f_i. Therefore, the adversary can sample them itself and broadcast g^{γ_i} for all $i \in Q$ instead of receiving them from the share algorithm. This reduces the unpredictability of Figure 2 scheme to the unpredictability of the Feldman VSS.

Accountability, clause (A1) (single-query attribution). The Share algorithm given $(1^\lambda, n, t, s)$, where s is a random secret from $\mathbb{Z}_q$, first samples a random degree-t polynomial $f(x) = s + a_1 X + \cdots + a_t X^t$, with random coefficients from $\mathbb{Z}_q$. Then it outputs $s_i := (f_i, \gamma_i)$, where $f_i = f(i)$, and $\pi_{share} = \mathsf{ak} = vk := (c_0, \ldots, c_t, c'_1, \ldots, c'_n)$ where $c_0 = g^s, c_1 = g^{a_1}, \ldots, c_t = g^{a_t}$, and for any $i \in [n]$, $c'_i = g^{\gamma_i}$. Then the adversary given $\{s_i\}_{i\in Q}, \pi_{share}, F(\cdot)$ for a set $Q \subseteq [n]$ returns either $\{z := F(s), \pi_{rec}\}$ or $\{\{z_j := F(f_j)\}_{j\in Q}, \pi_{rec}\}$ where $\pi_{rec} = \{\pi_j, \mathsf{who}\}_{j\in Q}$. The audit algorithm $\mathsf{VerifAudit}$ in Figure 2 given the audit

key $\mathsf{ak} := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, it computes $y_j := \prod_{k=0}^{t} c_k^{j^k}$ for $j \in Q$ (which can be deduced from the n-bit string who) and then verifies the proofs $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j\in Q}$ using the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ and commitment values $\{y_j, c'_j\}_{j\in[n]}$. For any statement $(c_i, c'_i, F(\cdot))$, which the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ on at least one of the proofs in $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j\in Q}$ returns 1, the audit algorithm puts i in I. If there is no statement $(c_i, c'_i, F(\cdot))$ which has an accepting proof in $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j\in Q}$, it returns $\{0, \emptyset, 0^n\}$. If I is non-empty, the audit algorithm returns $1, I, \pi_{\mathsf{Audit}} = \{\pi_i\}_{i\in I}$. Since $\mathsf{VerifAudit}$ is deterministic and puts into I exactly the indices whose proofs verify, $I = \mathsf{Contrib}(T)$ whenever $b = 1$, and $b = 0$ only if $\mathsf{Contrib}(T) = \emptyset$, with probability 1; it remains to show $I \subseteq Q$. Analyzing the following cases concludes the proof:

- If the audit algorithm returns $\{0, \emptyset, 0^n\}$, nothing is left to prove.
- Assume that with a non-negligible probability ϵ, there is an index $i^* \in I$ such that $i^* \notin Q$. That is, the adversary does not know the corresponding witness $s_{i^*} := (f_{i^*}, \gamma_{i^*})$ for the statement $(c_{i^*}, c'_{i^*}, F(\cdot))$, however the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ on π_{i^*} returns 1 with a non-negligible probability ϵ. Since $\Pi^{(1)}_{\mathsf{NIZK}}$ is knowledge sound (with respect to the Definition 16), there is an efficient extractor that recovers $(\hat{f}_{i^*}, \hat{\gamma}_{i^*})$ with non-negligible probability. Since $f_{i^*} = f(i^*)$ is computable from the adversary's own shares by Lagrange interpolation (Lemma 1), the binding comes entirely from $\hat{\gamma}_{i^*}$: it satisfies $g^{\hat{\gamma}_{i^*}} = c'_{i^*}$, i.e., the extractor solves the discrete-logarithm instance $c'_{i^*} = g^{\gamma_{i^*}}$, which contradicts the DL assumption.
- It is trivial that Judge in Figure 2 outputs `true` because it executes the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ on $\pi_{\mathsf{Audit}} = \{\pi_i\}_{i\in I}$ similar to the audit algorithm.

Accountability, clause (A2) (non-imputability). The Share algorithm given $(1^\lambda, n, t, s)$, where s is a random secret from $\mathbb{Z}_q$, first samples a random degree-t polynomial $f(x) = s + a_1 X + \cdots + a_t X^t$, with random coefficients from $\mathbb{Z}_q$. Then it outputs $s_i := (f_i, \gamma_i)$, where $f_i = f(i)$, and $\pi_{share} = \mathsf{ak} = vk := (c_0, \ldots, c_t, c'_1, \ldots, c'_n)$ where $c_0 = g^s, c_1 = g^{a_1}, \ldots, c_t = g^{a_t}$, and for any $i \in [n]$, $c'_i = g^{\gamma_i}$. Then the adversary given $\{s_i\}_{i\in[n]\setminus i^*}, \pi_{share}, F(\cdot)$ returns $(1, I, \pi_{\mathsf{Audit}} = \{\pi_i\}_{i\in I})$. The Judge algorithm in Figure 2 given the verification key $vk := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, it uses pre-computed values $y_j := \prod_{k=0}^{t} c_k^{j^k}$ for $j \in [n]$ and then verifies the proofs $\pi_{rec} := \{\pi_j\}_{i\in I}$ using the verification algorithm of $\Pi^{(1)}_{\mathsf{NIZK}}$ and commitment values $\{y_j, c'_j\}_{j\in[n]}$. If all the verification pass, it returns `true`. Otherwise, it returns `false`.

Assume that the Judge algorithm returns `true` and $i^* \in I$. That means the adversary has returned an accepting proof π_{i^*} for the statement $(c_{i^*}, c'_{i^*}, F(\cdot))$, without knowing the corresponding witness $s_{i^*} := (f_{i^*}, \gamma_{i^*})$. Since $\Pi^{(1)}_{\mathsf{NIZK}}$ is knowledge sound (with respect to the Definition 16), there is an efficient extractor that recovers $(\hat{f}_{i^*}, \hat{\gamma}_{i^*})$ with non-negligible probability. Since f_{i^*} is publicly interpolable (Lemma 1), the binding comes from $\hat{\gamma}_{i^*}$ satisfying $g^{\hat{\gamma}_{i^*}} = c'_{i^*}$, i.e., the extractor solves the DL instance $c'_{i^*} = g^{\gamma_{i^*}}$, contradicting the DL assump-

tion. *Remark:* non-imputability holds as long as γ_{i^*} remains secret; a party that voluntarily discloses γ_{i^*} loses this protection. This completes the proof. □

E.3 Proof of Theorem 3

Proof. We present the proof for the interactive protocol, as the final NIZK proof is obtained by directly applying Fiat-Shamir transform on the Σ-protocol.

Completeness. When both P and V are honest, $R_1 = c^{r_1}$, $R_2 = g^{r_1}$, $R_3 = g^{r_2}$, $z_1 = r_1 + ef_i$ and $z_2 = r_2 + e\gamma_i$. Substituting P's response into V's checks:

$$c^{z_1} = c^{r_1}c^{ef_i} = R_1 d_i^e, \qquad g^{z_1} = g^{r_1}g^{ef_i} = R_2\ \mathsf{pk}_i^e, \qquad g^{z_2} = g^{r_2}g^{e\gamma_i} = R_3\ \mathsf{vid}_i^e.$$

All three checks hold, so V accepts, and completeness is satisfied.

Honest Verifier Zero-Knowledge (HVZK). Given the statement and a challenge e, the simulator samples $z_1', z_2' \leftarrow\!\$\ \mathbb{Z}_q$ and sets $R_1' = c^{z_1'}d_i^{-e}$, $R_2' = g^{z_1'}\mathsf{pk}_i^{-e}$, and $R_3' = g^{z_2'}\mathsf{vid}_i^{-e}$. All three verification equations hold by construction. In a real transcript, (z_1, z_2) is uniform in $\mathbb{Z}_q^2$ (because r_1, r_2 are uniform) and (R_1, R_2, R_3) is the unique triple determined by (e, z_1, z_2) and the verification equations. The same holds for the simulated transcript, so the two distributions are identical.

Special Soundness. Given two accepting transcripts $(R_1, R_2, R_3, e, z_1, z_2)$ and $(R_1, R_2, R_3, e', z_1', z_2')$ with $e \neq e'$, dividing the corresponding verification equations gives

$$c^{z_1 - z_1'} = d_i^{e-e'}, \qquad g^{z_1 - z_1'} = \mathsf{pk}_i^{e-e'}, \qquad g^{z_2 - z_2'} = \mathsf{vid}_i^{e-e'}.$$

Hence $\hat{f}_i = (z_1 - z_1')/(e - e') \bmod q$ satisfies $d_i = c^{\hat{f}_i}$ and $\mathsf{pk}_i = g^{\hat{f}_i}$, and $\hat{\gamma}_i = (z_2 - z_2')/(e - e') \bmod q$ satisfies $\mathsf{vid}_i = g^{\hat{\gamma}_i}$. Thus $(\hat{f}_i, \hat{\gamma}_i)$ is a witness for R_i^{Dec}. Since q is prime, extraction is unconditional and needs no computational assumption, so $\Pi_{\mathsf{NIZK}}^{Dec}$ is a proof (not merely an argument) of knowledge.

Since the Σ-protocol is public-coin and satisfies completeness, HVZK, and special soundness, the Fiat–Shamir transform (with sid and the full statement included in the hash) yields a NIZK proof of knowledge in the random oracle model. □

E.4 Proof of Theorem 4

Proof. To prove this theorem, we apply a technique similar to the one used in the proof of Theorem 2.

Correctness of the Combine Algorithm. By the correctness of the threshold ElGamal encryption [13,14], with a sufficiently large set of partial decryptions, the plaintext m can be retrieved with a probability 1.

Correctness of the Audit Algorithm. We show the correctness of the trace algorithm by analyzing two cases based on the cardinality of the set Q. W.l.o.g, we assume $Q = V = I$.

- When $0 < |Q| \leq t$, the combine algorithm returns $(\{d_j, \pi_j\}_{j\in Q}, \mathsf{who})$, where π_j is a non-interactive proof (or argument) generated by $\Pi^{Dec}_{\mathsf{NIZK}}$ for the relation R^{Dec}_j defined in Eq. (3) obtained from π_{dec}. The audit algorithm $\mathsf{VerifAudit}$ given $\{d_j, \pi_j\}_{j\in Q}$ and the verification key $\mathsf{vk} = (pp, \{\mathsf{pk}_i\}_{i\in[n]})$, verifies the proofs $\{\pi_j\}_{j\in Q}$ using the verification algorithm of $\Pi^{Dec}_{\mathsf{NIZK}}$. By the completeness property of $\Pi^{Dec}_{\mathsf{NIZK}}$, all proofs will be accepted. Finally, it returns $\pi_{\mathsf{Audit}} := \{\mathsf{ct}, d_j, \pi_j\}_{j\in Q}$.
- When $|Q| \geq t + 1$, the combine algorithm returns (m, π_{dec}), where $\pi_{dec} := \{\pi_i, \mathsf{who}\}_{i\in Q}$ and π_j is a non-interactive proof (or argument) generated by $\Pi^{Dec}_{\mathsf{NIZK}}$ for the relation R^{Dec}_j defined in Eq. (3). Similarly, by the completeness property of $\Pi^{Dec}_{\mathsf{NIZK}}$, all proofs $\{\pi_j\}_{j\in Q}$ will be accepted. Therefore, the audit algorithm $\mathsf{VerifAudit}$ returns $(1, Q, \pi_{\mathsf{Audit}})$, where $\pi_{\mathsf{Audit}} := \{\mathsf{ct}, d_j, \pi_j\}_{j\in Q}$.

IND-CPA. Given the IND-CPA security of ElGamal encryption (cf. Appendix A.5) and information theoretical security of Shamir Secret sharing (cf. Appendix A.1), next we prove the IND-CPA security of the proposed ATC (cf. Figure 4).

Let there is an adversary $\mathcal{A}$ that can break the IND-CPA security of ATC, we will use $\mathcal{A}$ to build an adversary $\mathcal{B}$ that can break the IND-CPA security of standard ElGamal with the same probability. The adversary $\mathcal{B}$ interacts with the standard ElGamal challenger in the following way. First, the adversary $\mathcal{A}$ provides the list of corrupted parties Q to $\mathcal{B}$. Once $\mathcal{B}$ receives public the group description $(\mathbb{G}, g)$ and a public key pk' from the standard ElGamal challenger sets this the global public key of ATC $\mathsf{pk} := \mathsf{pk}'$ and simulates the public keys, $\{\mathsf{pk}_i\}_{i\in[n]}$, as follows:

- W.l.o.g., let $|Q| = t$. For all $i \in Q$, $\mathcal{B}$ samples random values $f_i \leftarrow \mathbb{Z}_q$, and defines the corresponding public key as $\mathsf{pk}_i := g^{f_i}$. Note that, due to the information-theoretic security of the Shamir secret sharing scheme, any t shares are uniform and independent of f_0, so these values are distributed exactly as those generated by the standard Share algorithm.
- To generate the public key of the honest parties $k \in H, H := [n] \setminus Q$, $\mathcal{B}$ proceeds as follows:
  1. For all $i \in \tilde{T} := Q \cup \{0\}$, it computes the Lagrange coefficients, $L_{ki} = \prod_{j\in\tilde{T}, j\neq i} (j - k)/(j - i)$.
  2. It takes the public keys of corrupted parties $\{\mathsf{pk}_i\}_{i\in Q}$ and the global public key pk and then computes $\mathsf{pk}_k := \mathsf{pk}^{L_{k0}} \prod_{i\in Q} \mathsf{pk}_i^{L_{ki}}$.
- For all $i \in [n]$, $\mathcal{B}$ samples the identity secret $\gamma_i \leftarrow \mathbb{Z}_q$ itself, sets $\mathsf{vid}_i := g^{\gamma_i}$ and, for $i \in Q$, $\mathsf{sk}_i := (f_i, \gamma_i)$. $\mathcal{B}$ then sets $\mathsf{vk} := \mathsf{tk} := (pp, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \mathsf{vid}_1, \ldots, \mathsf{vid}_n)$, where $pp := g$.

This simulation follows KeyGen of Figure 4 exactly. pk is uniform, the corrupt shares are uniform, every honest $\mathsf{pk}_k = g^{f(k)}$ is the value determined by the unique degree-t polynomial through $(0, f_0)$ and $\{(i, f_i)\}_{i\in Q}$, and the γ_i are sampled independently of f as in the real protocol. The simulated $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in Q}, \mathsf{vk}, \mathsf{tk})$ is therefore identically distributed to the real one. After

receiving $(\{\mathsf{sk}_i\}_{i\in Q}, \mathsf{pk}, \mathsf{vk}, \mathsf{tk})$, $\mathcal{A}$ chooses two distinct messages m_0 and m_1 and sends them to $\mathcal{B}$. Then $\mathcal{B}$ forwards (m_0, m_1) to the standard ElGamal challenger, who encrypts one of them based on the random bit β, and returns the ciphertext ct_β to $\mathcal{B}$. $\mathcal{B}$ then passes this ciphertext to $\mathcal{A}$. Afterwards $\mathcal{A}$ returns the bit b', and $\mathcal{B}$ uses $\mathcal{A}$'s guess as its own guess in the standard ElGamal IND-CPA game.

Since the encryption process in both standard and threshold ElGamal is identical, the ciphertext that $\mathcal{A}$ sees is indistinguishable from what it would see in the ATC setting. As a result, if $\mathcal{A}$ can successfully distinguish between encryptions in IND-CPA game for the ATC, then $\mathcal{B}$ can also do so against the IND-CPA game for the standard ElGamal, thus breaking the IND-CPA security of standard ElGamal. This concludes the IND-CPA security of our proposed ATC.

Public Auditability (A1). In the public-auditability game (A1), defined in Definition 25, the adversary given a set of secret keys $\{\mathsf{sk}_i\}_{i\in Q}$ returns either $\{m, \pi_{dec}\}$ or $\{\{d_j\}_{j\in Q}, \pi_{dec}\}$ where $\pi_{dec} = \{\pi_j, \mathsf{who}\}_{j\in Q}$. The audit algorithm $\mathsf{VerifAudit}$ verifies the proofs $\pi_{dec} := \{\pi_j\}_{j\in Q}$ using the verification algorithm of $\Pi^{Dec}_{\mathsf{NIZK}}$ and partial ciphertext d_j and ciphertext ct. For any statement x, which the verification algorithm of $\Pi^{Dec}_{\mathsf{NIZK}}$ on at least one of the proofs in $\pi_{dec} := \{\pi_j\}_{j\in Q}$ accepts, the audit algorithm updates $I = I \cup \{i\}$, otherwise it returns $\{0, \emptyset, 0^n\}$. If I is non-empty, $\mathsf{VerifAudit}$ returns $(1, I, \pi_{\mathsf{Audit}} = \{\pi_i\}_{i\in I})$. Hence $I = \mathsf{Contrib}(T)$ whenever $b = 1$, and $b = 0$ only if $\mathsf{Contrib}(T) = \emptyset$, with probability 1. Analyzing the following cases concludes the proof:

- If the audit algorithm $\mathsf{VerifAudit}$ returns $\{0, \emptyset, 0^n\}$, nothing is left to prove.
- Assume that with a non-negligible probability ϵ, there is an index $i^* \in I$ s.t. $i^* \notin Q$. Then π_{i^*} is an accepting proof for the statement $(g, c_2, d_{i^*}, \mathsf{pk}_{i^*}, \mathsf{vid}_{i^*})$, and by knowledge soundness of $\Pi^{Dec}_{\mathsf{NIZK}}$ an extractor obtains $(\hat{f}_{i^*}, \hat{\gamma}_{i^*})$ with $\mathsf{vid}_{i^*} = g^{\hat{\gamma}_{i^*}}$. A reduction that embeds a DL challenge as vid_{i^*} simulates KeyGen perfectly, because γ_{i^*} is independent of all other keys, and thereby solves DL with probability close to ϵ. This contradicts the DL assumption. (Note that $\hat{f}_{i^*}$ alone yields no contradiction, since f_{i^*} may be interpolable by the coalition.) Therefore, $I \subseteq Q$.
- It is trivial that Judge in Figure 4 outputs `true` because it re-runs the verification algorithm of $\Pi^{Dec}_{\mathsf{NIZK}}$ on $\pi_{\mathsf{Audit}} := \{\mathsf{ct}, d_j, \pi_j\}_{j\in I}$ exactly as $\mathsf{VerifAudit}$ does.

Non-imputability. In the non-imputability game, defined in Definition 26, the adversary given a set of secret keys of all parties except the challenger party i^*, i.e. $\{\mathsf{sk}_i\}_{i\in[n]\setminus\{i^*\}}$ returns $(1, I, \pi_{\mathsf{Audit}})$. The Judge algorithm in Figure 4 given the verification key $\mathsf{vk} = (pp, \{\mathsf{pk}_i\}_{i\in[n]})$, verifies the proofs $\pi_{\mathsf{Audit}} := \{\mathsf{ct}, d_j, \pi_j\}_{j\in I}$ using the verification algorithm of $\Pi^{Dec}_{\mathsf{NIZK}}$ and ciphertext ct. If all the verifications pass, it returns `true`. Otherwise, it returns `false`.

Assume that the Judge algorithm returns `true` and a subset I such that $i^* \in I$. That means the adversary has returned an accepting proof π_{i^*} for the statement $(g, c, d_{i^*}, \mathsf{pk}_{i^*}, \mathsf{vid}_{i^*})$, although it does not know γ_{i^*}. By knowledge soundness of $\Pi^{Dec}_{\mathsf{NIZK}}$, the extractor outputs $\hat{\gamma}_{i^*}$ with $\mathsf{vid}_{i^*} = g^{\hat{\gamma}_{i^*}}$. Embedding a

DL challenge as vid_{i^*} (all other keys, including f_{i^*}, are sampled by the reduction, which is a perfect simulation) therefore solves DL, contradicting the DL assumption. The adversary knows f_{i^*} by interpolation, so security rests entirely on γ_{i^*}. This concludes the proof. □

E.5 Proof of Theorem 5

Proof. **Correctness of the Reconstruction Algorithm.** By the correctness of the Pedersen VSS, the secret value s is constructed with a probability 1.

Correctness of the Audit Algorithm. We show the correctness by analyzing two cases based on the size of Q:

- When $0 < |Q| \leq t$, the reconstruction algorithm returns $\{F(f_j), \pi_j, \mathsf{who}\}_{j \in Q}$ where π_j is a non-interactive proof (or argument) generated by $\Pi^{(2)}_{\mathsf{NIZK}}$ for the relation $R^{(2)}_j$ defined in equation (6). The audit algorithm $\mathsf{VerifAudit}$ given $\{F(f_j), \pi_j, \mathsf{who}\}_{j \in Q}$ and the audit key $\mathsf{ak} := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, computes $y_j := \prod_{k=0}^{t} c_k^{j^k}$ for $j \in Q$ (which can be deduced from the n-bit string who) and then verifies the proofs $\{\pi_j\}_{j \in Q}$ using the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ and commitment values $\{y_j, c'_j\}_{j \in Q}$. By the completeness property of $\Pi^{(2)}_{\mathsf{NIZK}}$, all proofs will be accepted. Finally, it returns Q and $\pi_{\mathsf{Audit}} := (\{F(f_j)\}_{j \in Q}, \{\pi_j\}_{j \in Q})$.
- When $|Q| \geq t + 1$, the reconstruction algorithm returns $\{F(s), \pi_j, \mathsf{who}\}_{j \in Q}$ where π_j is a non-interactive proof (or argument) generated by $\Pi^{(2)}_{\mathsf{NIZK}}$ for the relation $R^{(2)}_j$ defined in equation (6). Similarly, by the completeness property of $\Pi^{(2)}_{\mathsf{NIZK}}$, all proofs $\{\pi_j\}_{j \in Q}$ will be accepted. Therefore, the $\mathsf{VerifAudit}$ audit algorithm returns Q and $\pi_{\mathsf{Audit}} := (\{F(s)\}_{j \in Q}, \{\pi_j\}_{j \in Q})$.

Verifiability. It holds directly by the verifiability of the Pedersen VSS. (Note that, the only difference between the verification in Figure 5 and in the Pedersen scheme is that the parties check if c'_j are broadcast and well-formed. Since Q contains honest parties, this extra check does not affect the output of verification algorithm in Figure 5.)

Statistical Secrecy. The statistical secrecy of the protocol in Figure 5 is guaranteed by the statistical secrecy properties of the Pedersen VSS, which stem from the perfect hiding property of the Pedersen commitment scheme. The reason is that, for each i, the additional component of s_i, denoted as γ_i, is chosen by party i randomly and independently of f_i. Therefore, the adversary can sample them itself and broadcast g^{γ_i} for all $i \in Q$ instead of receiving them from the share algorithm. This reduces the secrecy of Figure 5 scheme to the secrecy of the Pedersen VSS.

Accountability, clause (A1) (single-query attribution). The Share algorithm given $(1^\lambda, n, t, s)$, where s is a secret from $\mathbb{Z}_q$, samples a random secret

$r_0 \in \mathbb{Z}_q$ and a two random degree-t polynomials $f(x) = s + a_1X + \cdots + a_tX^t$ and $r(x) = r_0 + b_1X + \cdots + b_tX^t$, and random coefficients $\{a_j, b_j\}_{j=1}^t$ from $\mathbb{Z}_q$. Then it outputs $s_i := (f_i, r_i, \gamma_i)$, where $f_i = f(i)$, $r_i = r(i)$, and $\pi_{share} = \mathsf{ak} = vk := (c_0, \ldots, c_t, c'_1, \ldots, c'_n)$ where $c_0 = g^s h^{r_0}, c_1 = g^{a_1} h^{b_1}, \ldots, c_t = g^{a_t} h^{b_t}$, and for any $i \in [n]$, $c'_i = g^{\gamma_i}$. Then the adversary given $\{s_i\}_{i \in Q}, \pi_{share}, F(\cdot)$ for a set $Q \subseteq [n]$ returns either $\{z := F(s), \pi_{rec}\}$ or $\{\{z_j := F(f_j)\}_{j \in Q}, \pi_{rec}\}$ where $\pi_{rec} = \{\pi_j, \mathsf{who}\}_{j \in Q}$. The audit algorithm $\mathsf{VerifAudit}$ in Figure 5 given the audit key $\mathsf{ak} := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, computes $y_j := \prod_{k=0}^t c_k^{j^k}$ for $j \in Q$ (which can be deduced from the n-bit string who) and then verifies the proofs $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j \in Q}$ using the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ and commitment values $\{y_j, c'_j\}_{j \in [n]}$. For any statement $(c_i, c'_i, F(\cdot))$, which the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ on at least one of the proofs in $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j \in Q}$ returns 1, the audit algorithm puts i in I. If there is no statement $(c_i, c'_i, F(\cdot))$ which has an accepting proof in $\pi_{rec} := \{\pi_j, \mathsf{who}\}_{j \in Q}$, it returns $\{0, \emptyset, 0^n\}$. If I is non-empty, the audit algorithm returns $1, I, \pi_{\mathsf{Audit}} = \{\pi_i\}_{i \in I}$. Since $\mathsf{VerifAudit}$ is deterministic and puts into I exactly the indices whose proofs verify, $I = \mathsf{Contrib}(T)$ whenever $b = 1$, and $b = 0$ only if $\mathsf{Contrib}(T) = \emptyset$, with probability 1; it remains to show $I \subseteq Q$. Analyzing the following cases finishes the proof:

- If the audit algorithm returns $\{0, \emptyset, 0^n\}$, nothing is left to prove.
- Assume that with a non-negligible probability ϵ, there is an index $i^* \in I$ such that $i^* \notin Q$. That is, the adversary does not know the corresponding witness $s_{i^*} := (f_{i^*}, r_{i^*}, \gamma_{i^*})$ for the statement $(c_{i^*}, c'_{i^*}, F(\cdot))$, however the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ on π_{i^*} returns 1 with a non-negligible probability ϵ. Since $\Pi^{(2)}_{\mathsf{NIZK}}$ is knowledge sound (with respect to the Definition 16), there is an efficient extractor that recovers $(\hat{f}_{i^*}, \hat{r}_{i^*}, \hat{\gamma}_{i^*})$ with non-negligible probability. Since f_{i^*} and r_{i^*} are computable by Lagrange interpolation from the adversary's shares (Lemma 1), the binding comes from $\hat{\gamma}_{i^*}$ satisfying $g^{\hat{\gamma}_{i^*}} = c'_{i^*}$, i.e., the extractor solves the DL instance $c'_{i^*} = g^{\gamma_{i^*}}$, contradicting the DL assumption.
- It is trivial that Judge in Figure 5 outputs `true` because it executes the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ on $\pi_{\mathsf{Audit}} = \{\pi_i\}_{i \in I}$ similar to the audit algorithm.

Accountability, clause (A2) (non-imputability). The Share algorithm given $(1^\lambda, n, t, s)$, where s is a secret from $\mathbb{Z}_q$, samples a random secret $r_0 \in \mathbb{Z}_q$ and a two random degree-t polynomials $f(x) = s + a_1X + \cdots + a_tX^t$ and $r(x) = r_0 + b_1X + \cdots + b_tX^t$, and random coefficients $\{a_j, b_j\}_{j=1}^t$ from $\mathbb{Z}_q$. Then it outputs $s_i := (f_i, r_i, \gamma_i)$, where $f_i = f(i)$, $r_i = r(i)$, and $\pi_{share} = \mathsf{ak} = vk := (c_0, \ldots, c_t, c'_1, \ldots, c'_n)$ where $c_0 = g^s h^{r_0}, c_1 = g^{a_1} h^{b_1}, \ldots, c_t = g^{a_t} h^{b_t}$, and for any $i \in [n]$, $c'_i = g^{\gamma_i}$. Then the adversary given $\{s_i\}_{i \in [n] \setminus i^*}, \pi_{share}, F(\cdot)$ returns $(1, I, \pi_{\mathsf{Audit}} = \{\pi_i\}_{i \in I})$. The Judge algorithm in Figure 5 given the verification key $vk := \pi_{share} = (c_0, c_1, \ldots, c_t, c'_1, \ldots, c'_n)$, uses pre-computed values $y_j := \prod_{k=0}^t c_k^{j^k}$ for $j \in [n]$ and then verifies the proofs $\pi_{rec} := \{\pi_j\}_{i \in I}$ using the verification algorithm of $\Pi^{(2)}_{\mathsf{NIZK}}$ and commitment values $\{y_j, c'_j\}_{j \in [n]}$. If all the verification pass, it returns `true`. Otherwise, it returns `false`.

Assume that the Judge algorithm returns `true` and $i^* \in I$. That means the adversary has returned an accepting proof π_{i^*} for the statement $(c_{i^*}, c'_{i^*}, F(\cdot))$, without knowing the corresponding witness $s_{i^*} := (f_{i^*}, r_{i^*}, \gamma_{i^*})$. Since $\Pi^{(2)}_{\mathsf{NIZK}}$ is knowledge sound (with respect to the Definition 16), there is an efficient extractor that recovers $(\hat{f}_{i^*}, \hat{r}_{i^*}, \hat{\gamma}_{i^*})$ with non-negligible probability. Since f_{i^*} and r_{i^*} are publicly interpolable (Lemma 1), the binding comes from $\hat{\gamma}_{i^*}$ satisfying $g^{\hat{\gamma}_{i^*}} = c'_{i^*}$, i.e., the DL instance $c'_{i^*} = g^{\gamma_{i^*}}$ is solved, contradicting the DL assumption. *Remark:* non-imputability holds as long as γ_{i^*} remains secret; voluntary disclosure of γ_{i^*} by P_{i^*} removes this protection. This completes the proof. □