Towards Public Tracing: Collaborative Traceable Secret Sharing

Pousali Dey, Rittwik Hajra, Subha Kar, and Soumit Pal

Indian Statistical Institute, Kolkata, India
{deypousali95,rittwik1729,subhakar9732,soumitpal378}@gmail.com

Abstract. In a (t, n)-threshold secret sharing scheme, secrecy holds as long as fewer than t servers collude. If $f < t$ parties are corrupt and they sell their shares, there is no mechanism to hold them accountable in classical secret sharing schemes. Goyal–Song–Srinivasan [CRYPTO'21] introduced Traceable Secret Sharing (TSS) and later Boneh–Partap–Rotem [CRYPTO'24] made it practical: $f < t$ corrupt servers produce a reconstruction box $\mathcal{R}$ that, given $t - f$ extra shares, outputs the secret. The task is to trace $\mathcal{R}$ back to the corrupted servers, given black-box access to $\mathcal{R}$. Prior works on TSS rely on a designated tracer with private trace keys for tracing and/or verification.
We remove the dependence on any designated tracer and propose Collaborative Traceable Secret Sharing (CTSS), which eliminates the private trace key and the private verification key. Instead, tracing requires collaboration from a threshold number of parties, and verification is fully public. We define the CTSS framework, along with its security notions, and present two efficient collaborative traceable secret sharing schemes based on the classical Shamir and Blakley schemes. Both achieve secrecy, traceability, and non-imputability, with minimal share size overhead and polynomial-time tracing effectively eliminating the need for a designated tracing authority.

Keywords: Secret Sharing, Traceable Secret Sharing, Non-imputability

1 Introduction

Secret Sharing, introduced by Shamir [Sha79] and Blakley [BLA79], is a core cryptographic primitive. In a (t, n)-threshold secret sharing scheme, a dealer splits a secret among n servers; any t can reconstruct it, but any fewer than t learn nothing. It is widely used, for example, to distribute an encryption key across cloud servers to prevent a single point of failure. Despite its security guarantee, traditional secret sharing lacks accountability. For instance, Alice shares her key across n servers. An adversary could bribe a small coalition $f < t$ to buy their shares. To recover the secret, the attacker only needs to compromise $t - f$ more servers. A leaked share is an anonymous token; it contains no intrinsic information that can be used to forensically link it back to the specific server that leaked it. The fundamental problem is that traditional secret sharing schemes provide no accountability.

Traceable Secret Sharing. To mitigate this accountability issue, Goyal, Song, and Srinivasan (GSS) [GSS21] introduced the concept of Traceable Secret Sharing (TSS), which was impractical and demands a quadratic share size. Boneh, Partap, and Rotem [BPR24] refined the model of GSS: a designated tracer, with trace keys and verification keys, has black-box access to an adversarial reconstruction box $\mathcal{R}$ hard-coded with $f < t$ shares and, given $t - f$ extra shares, outputs the secret formed by those t shares. A scheme is *traceable* if the tracer can identify all f leaking parties, i.e., shares hard-coded in $\mathcal{R}$ and produce a proof verifiable with the verification key. A scheme is *non-imputable* if the tracer should not be able to generate a false proof, implicating an honest server who was not involved in any leak. Practical TSS schemes were achieved for major secret sharing methods, such as Shamir, Blakley [BPR24], Mignotte [Hof24], and Asmuth–Bloom [LD25] secret sharing schemes, confirming the utility and efficiency of the black-box model.

Private and Semi-Public Tracing. The traceability notion proposed by Boneh, Partap, and Rotem [BPR24] utilizes private tracing and verification keys, making the tracing process private; hence, the tracing process is not public. The current TSS constructions [BPR24], [LD25], [HKMP25] are privately traceable: a designated tracer holds a private trace key tk and a private verification key vk. In contrast, Hoffman [Hof24] introduced semi-public tracing, where anyone can trace, but verification remains private.

Motivation for Public Tracing. The dealer or tracer has to maintain a private trace keys tuple to perform trace, which is a circular problem, because to trace the corrupt parties, a secret should be maintained. We are eliminating the need for maintaining separate secrets by making trace collaborative. A key motivation for secret sharing is the desire to avoid storing the secret, whether because of storage limitations or the risk of accidental loss. Keeping that in mind two major motivation behind the distributed/collaborative tracing are the following:

1. In private tracing, the tracer is an entity that possesses a private trace key. Here, we emphasize single-point failure to refer specifically to the risk of the tracer losing the trace key, rather than the tracer's inability to perform tracing or a failed tracing attempt.
2. While secret sharing is intended to avoid storing the secret, private tracing requires the tracer to store secret information of size $O(n)$ times the secret, which defeats this purpose.

Technical Challenges of Public Tracing. Boneh et al. [BPR24] and Hoffman [Hof24] show that fully public tracing is incompatible with the notion of current TSS. If the tracing key is public, an adversarial reconstruction box $\mathcal{R}$ can detect tracer queries and return $\perp$. For example, in traceable Shamir [BPR24] the tracing key cosists of $F(x_1), \ldots, F(x_n)$ for a one-way function F, while shares are (x_i, y_i). Then, $\mathcal{R}$ can check a random query (x^*, y^*) by comparing $F(x^*)$ to the trace keys and output $\perp$ when there is no match; one-wayness of F makes matches negligible, so random probing is detectable by $\mathcal{R}$ and tracing is not

possible. Hoffman [Hof24] proposed a semi-public TSS where tracing is public, and verification is private. The current TSS schemes are private or semi-public, leaving fully public traceability as a significant open problem.

Pitfalls of Private and Semi-Public Tracing. A significant limitation of all existing TSS schemes is their dependence on private or semi-public tracing. The black-box model makes tracing efficient, but tracing and/or verification remain centralized: in private tracing, both are private, and in semi-public tracing, only verification is private. Relying on a designated tracer with a private tracing key tk (and a private verification key vk in the semi-public model) creates a trust bottleneck, as the designated tracer becomes a powerful centralized authority. If its private tracing key is compromised, an adversary can evade detection (e.g., always output $\perp$), creating a *single point of failure*. Tracing outputs are not publicly verifiable for both private [BPR24,LD25,HKMP25] and semi-public [Hof24] models, so the dealer must depend on the tracer.
More practically, in decentralized settings involving multiple independent organizations (e.g., different cloud providers), no single entity can be universally trusted as a tracer. Boneh et al. [BPR24] suggested public tracing as future work. This leads to our central question:

Can there be a publicly traceable threshold secret sharing scheme?

Existing semi-public schemes [Hof24] still need private verification keys, so they are not fully public.

1.1 Our Contribution and Technical Overview

We introduce the first formal framework for *Collaborative Traceable Secret Sharing* (CTSS), where tracing is done *collaboratively* by a coalition of parties and verification is public. Our model eliminates the need for a designated tracing authority by making tracing a collaborative operation. Our contributions are threefold:

1. We introduce new syntax and security definitions for collaborative, traceable secret sharing, where tracing is performed by a threshold number of parties and verification can be conducted by anyone.
2. We present a new and efficient collaborative tracing mechanism for standard Shamir secret sharing.
3. We construct an efficient, collaborative, and traceable version of the Blakley secret sharing scheme.

Work	TSS model	Trace	Verify
[BPR24], [LD25], [HKMP25]	Private	Private	Private
[Hof24]	Semi-Public	Public	Private
[This Work]	Collaborative	Collaborative	Public

Our Model of Collaborative Traceable Secret Sharing. In Section 2, we formalize CTSS, a paradigm shift from centralized TSS models [BPR24], [Hof24] that rely on a designated tracer holding private trace key and/or private verification key, creating a single point of failure. In our model of CTSS, tracing is collaborative, and verification is public, making it ideal for multi-organization decentralized systems. A CTSS scheme extends a (t, n)-threshold secret sharing with a collaborative trace and a public verification. CTSS consists of four PPT algorithms: Share, Recon, Trace, Verify. The Share algorithm outputs shares sh_i, a unique trace credential φ_i for each $i \in [n]$, and a public value pub. Crucially, pub enables public verification, but alone does not allow tracing. Tracing requires a coalition $\mathcal{C} \subseteq [n]$ of t parties pooling their credentials $\{\varphi_i\}_{i\in\mathcal{C}}$. Our core innovation is the trace credential φ_i. Any collection of trace credentials φ_i must not leak the secret s. Each trace credential φ_i is authenticated to prevent parties from providing false trace credentials during collaborative tracing. Trace algorithm is run by a coalition $\mathcal{C} \subseteq [n]$ with black-box access to a pirate reconstruction box $\mathcal{R}$ that may hard-code $f < t$ shares (and possibly those corrupt parties' trace credentials). Given pub and valid trace credentials $\varphi_{\mathcal{C}}$, Trace outputs a set of corrupt party's identities $\mathcal{I} \subseteq [n]$ and a proof π. Verify is deterministic and allows anyone to check the accusation $\mathcal{I}$ using pub and π, removing reliance on a designated tracer with private verification keys. We define three security properties. *Secrecy:* An adversary with pub, all trace credentials, and any $t-1$ shares reveal nothing about the secret. *Traceability:* A coalition of t parties can trace a reconstruction box $\mathcal{R}$ with $f < t$ leaked shares to identify the corrupt set $\mathcal{I}$. *Non-imputability:* no adversary can forge a valid π that falsely accuses an honest $i^* \notin \mathcal{I}$.

One-time Nature of Collaborative Tracing. Collaborative tracing is a one-time procedure. Since tracing is an expensive operation even for the case of private tracing in previous works like [BPR24] and, unlike signature generation, is not performed frequently. It is reasonable to assume that once tracing is executed and corruption is detected, the *trace credentials* must be refreshed. In particular, the polynomial q must be updated to preserve non-imputability and other security properties. Moreover, as f shares may have been leaked, the shares themselves must also be renewed to protect the secret.

Honest Majority. In the tracing phase, t parties collaboratively reconstruct the polynomial q and the proof π consists of the $(C_{i_j}, x_{i_j}, d_{i_j})$. Then it seems that n should be at least $2t-1$. Otherwise, a corrupted user can refuse to join the collaborative tracing procedure. An honest-majority assumption can be employed when a party refuses to participate in tracer collaboration. To avoid relying on this assumption, one may instead adopt accountability upon refusal, whereby a party that declines to participate in tracing can be accused of being corrupt. Although this might create a problem of getting falsely identified as corrupt parties.

Collaborative Traceable Shamir. CTS is a t-out-of-n scheme that extends Shamir secret sharing with trace credentials. Shares remain compact, each containing two field elements. The dealer samples two polynomials over a finite field $\mathbb{F}$: $p(X) = s + \sum_{j=1}^{t-1} a_j X^j$ of degree $t-1$, with secret $s = p(0)$; and $q(X) = d + \sum_{j=1}^{t} b_j X^j$ of degree t, with public $d = q(0)$. For each party $i \in [n]$, the dealer: chooses a random $x_i \in \mathbb{F}$ and computes the share $y_i = p(x_i)$, computes $u_i = \mathcal{F}(x_i)$ using a one-way function $\mathcal{F}$ and the trace credential $\varphi_i = (u_i, d_i)$, where $d_i = q(u_i)$; and publishes a public identifier $\mathsf{C}_i = \mathcal{H}(u_i \parallel d_i)$. The private share for party i is $\mathsf{sh}_i = (x_i, y_i)$. The public value is $\mathsf{pub} = (\mathsf{C}_1, \ldots, \mathsf{C}_n; d)$.

Any coalition $\mathcal{C} \subseteq [n]$ with t trace credentials may run tracing. If a corrupt party provides a false credential, it is caught by checking $\mathsf{C}_k \stackrel{?}{=} \mathcal{H}(u_k \parallel d_k)$ which holds only for valid φ_k due to collision resistance of $\mathcal{H}$. With $\mathcal{C}$ and black-box access to a pirate reconstruction box $\mathcal{R}$ (holding $f < t$ leaked shares), tracing reduces to a Reed–Solomon list decoding problem. The tracer queries $\mathcal{R}$ to obtain noisy points, uses the Guruswami–Sudan algorithm to list all polynomials of degree $\leq f$, and for each root w computes $u' = \mathcal{F}(w)$ and $d' = q(u')$, checking whether $\mathcal{H}(u' \parallel d')$ matches some C_i in pub. These matches identify the corrupt set $\mathcal{I}$. The proof is $\pi = \{(\mathsf{C}_i, x_i, d_i)\}_{i \in [f]}$. Verification is fully public: anyone recomputes $u_i = \mathcal{F}(x_i)$ and checks $\mathsf{C}_i \stackrel{?}{=} \mathcal{H}(u_i \parallel d_i)$. Secrecy follows from Shamir, traceability from list decoding, and non-imputability from the one-wayness of $\mathcal{F}$, since forging π would invert $\mathcal{F}$. Tracing runs in $\mathrm{poly}(\lambda, 1/\varepsilon)$ time using $N = O(1/\varepsilon^2)$ queries.

Collaborative Traceable Blakley. We recall that the Blakley$^+$ secret-sharing scheme of [BPR24], used as a building block, is deferred to Appendix A.2. Our construction of *Collaborative Traceable Blakley* (CTB) builds on this scheme while adding collaborative, publicly verifiable tracing. As in Blakley$^+$, the dealer first decomposes the secret $\boldsymbol{s} \in \mathbb{F}_p^t$ as $\boldsymbol{s} = \boldsymbol{r}_1 + \cdots + \boldsymbol{r}_t$ with each $\boldsymbol{r}_j \in \mathbb{F}_p^t$, and shares every $\boldsymbol{r}_j$ via a standard (t, n) Blakley scheme so that party i receives hyperplanes $(H_i^{(1)}, \ldots, H_i^{(t)})$ where $H_i^{(j)}$ is a random hyperplane passing through $\boldsymbol{r}_j$. To support tracing, CTB masks the hyperplanes corresponding to $\boldsymbol{r}_1$: the dealer samples $d \leftarrow \mathbb{F}_p$, Shamir-shares d among the parties, and publishes only the homomorphic encodings $(g^{\boldsymbol{a}_{i,1} + \boldsymbol{d}}, g^{b_{i,1}+d})$ for $\boldsymbol{d} = (d, \ldots, d) \in \mathbb{F}_p^t$ in a cyclic group of order p. Under the Discrete-Log Relation (DLR) assumption, these ciphertext-like values hide the affine coefficients and prevent adversaries from testing affine constraints in the exponent, ensuring that no single party can unmask or trace alone. For the reconstruction of the secret, any t parties with any t shares obtain a consistent system of t linear equations for each unknown $\boldsymbol{r}_j$ ($j \in [t]$). Solving these t systems yields all component vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t$ and hence the secret $\boldsymbol{s}$. Thus, CTB preserves correctness and secrecy while enabling threshold-collaborative and publicly verifiable tracing.

The tracing algorithm takes the public data pub, a set of t trace credentials $\varphi_{\mathcal{C}}$, and outputs a set of corrupt parties $\mathcal{I}$ along with a proof π. The tracer first queries the reconstruction box $\mathcal{R}$ using $t - f$ genuine hyperplanes for each $\boldsymbol{r_j}$ ($j \geq 2$) and $t - f$ random hyperplanes, obtains $\boldsymbol{s}^*$, and recovers $\boldsymbol{r_1}^* = \boldsymbol{s}^* - (\boldsymbol{r_2} + \cdots + \boldsymbol{r_1})$.

Using the credentials in $\varphi_{\mathcal{C}}$, it interpolates d and then un-masks pub to compute $(g^{\boldsymbol{a}_{i,1}}, g^{bi,1})$ for all $i \in [n]$ via $g^{\boldsymbol{a}_{i,1}} = g^{\boldsymbol{d_i}} \cdot g^{-\boldsymbol{d}}$ and $g^{b_{i,1}} = g^{e_i} \cdot g^{-d}$, where $\boldsymbol{d} = (d, \ldots, d)$. A party i is identified as corrupt if $\prod_{j=1}^{t} (g^{a^j_{i,1}})^{r^{*,j}_1} = g^{b_{i,1}}$, which amounts to checking whether the hyperplane defined by $\boldsymbol{a}_{i,1}$ passes through $\boldsymbol{r}^*_1$ by verifying the inner product $\langle \boldsymbol{a}_{i,1}, \boldsymbol{r}^*_1 \rangle$ in the exponent.

1.2 Related Work

Chor, Fiat, and Naor [CFNP00] introduced traitor tracing for broadcast encryption, and a large body of subsequent work developed many constructions and techniques [NP98], [BF99], [FT99], [SNW00], [KY01], [NNL01], [BSW06], [BN08], [GKSW10], [BZ17], [GKW18], [CVW+18], [Zha20], [Wee20].

Following the formalization of TSS [GSS21], Boneh et al. [BPR24] established a practical framework, constructing efficient TSS for Shamir and Blakley schemes. This spurred numerous extensions, including TSS for other schemes like Mignotte [Hof24], Asmuth-Bloom [LD25], Bottom-Up Secret Sharing [HKMP25] as well as new primitives like traceable VSS [GKMN25,BEMS25], traceable VRF [BPR25], TSS for general access structures [FG25], traceable threshold encryption [BBF+25,BHO25], 'secret sharing with snitching' [DFLM24,BDF+25]. Recently, Goyal et al. [GJP25] strengthened the TSS notion for monotone access structures using advanced cryptographic techniques like indistinguishability obfuscation.

1.3 Notations

We use $\mathbb{N}$ for the set of positive integers, and for $n \in \mathbb{N}$ we write $[n] := \{1, 2, \ldots, n\}$. Let $\mathbb{F}$ denote a finite field. For any finite set X, the notation $x \xleftarrow{\$} X$ denotes picking an element of X uniformly at random and assigning it to x. For a vector $\boldsymbol{a} = (a_1, \ldots, a_t) \in \mathbb{F}_q^t$ and a cyclic group $\mathbb{G}$ of order q generated by g, we write $g^{\boldsymbol{a}} = (g^{a_1}, \ldots, g^{a_t})$. We use one-way functions in Section 3; see [Gol01] for definitions. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$ be a collision-resistant hash function modeled as a random oracle. We describe the modified Shamir and Blakley secret-sharing schemes from Boneh, Partap, and Rotem [BPR24] in Section A. The Reed-Solomon List Decoding and its solver, the Guruswami-Sudan List Decoding algorithm, are stated in Section A.

Organization of the Paper. In section 2 we explain the CTSS model, syntax and security definitions. Section 3 contains a detailed technical discussion and construction of *Collaborative Traceable Shamir*. In section 4 we discuss *Collaborative Traceable Blakely*. Section 5 contains complexity comparison, and in section 6 we conclude and state future direction.

2 Collaborative Traceable Secret Sharing

In this section, we formalize the notion of *Collaborative Traceable Threshold Secret Sharing* (CTSS). Our model departs from the previous work of Boneh, Partap, and Rotem [BPR24] by eliminating the need for a designated tracing authority.

2.1 Syntax and Correctness

Definition 1. *A Collaborative Traceable Secret Sharing Scheme is a tuple of four algorithms,* $\mathsf{CTSS} = (\mathsf{Share}, \mathsf{Recon}, \mathsf{Trace}, \mathsf{Verify})$ *for an underlying secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$*, where* $\mathcal{S}_\lambda$ *denotes the secret space parameterized by the security parameter* λ*.*

- $\mathsf{Share}(1^\lambda, s, n, t, \rho) \to (\mathsf{pub}, \{\mathsf{sh}_i\}_{i=1}^n, \{\varphi_i\}_{i=1}^n)$*. It is a randomized algorithm run by a trusted dealer. It takes a security parameter* λ*, a secret* $s \in \mathcal{S}_\lambda$*, numbers of parties* n *and threshold* $t \leq n$*, a correlation string* $\rho \in \{0,1\}^\kappa$ *with* $\kappa = \kappa(n, t, \lambda) \in \mathbb{N}$*. It then outputs shares of* n *parties i.e.,* $\{\mathsf{sh}_i\}_{i=1}^n$*, the 'trace credential' for* n *parties* $\{\varphi_i\}_{i=1}^n$ *, and public value* pub*.*
- $\mathsf{Recon}(\mathsf{pub}, \{\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_t}\}) \to s$*. It is a deterministic algorithm and run by a coalition of parties with at least* t *shares to reconstruct the secret* s*.*
- $\mathsf{Trace}^{\mathcal{R}}(\mathsf{pub}, \varphi_{\mathcal{C}}, 1^{1/\delta}) \to (\mathcal{I}, \pi)$*. It is a randomized algorithm run by any* ***collaborative*** *tracer with public information* pub*, valid "trace credentials"* $\varphi_{\mathcal{C}}$ *of a set* $\mathcal{C} \subseteq [n]$ *of parties such that* $|\mathcal{C}| = t$*, and an error parameter* δ*. It outputs a subset* $\mathcal{I} \subseteq [n]$ *of identities of share leaking parties and a proof* π *that the parties in* $\mathcal{I}$ *are corrupt.*
- $\mathsf{Verify}(\mathsf{pub}, \mathcal{I}, \pi) \to \{0,1\}$*. It is a deterministic algorithm and runs by any* ***public*** *verifier. It takes as input the public value* pub*, an alleged corrupt subset* $\mathcal{I} \subseteq [n]$*, and an associated proof* π*, and outputs either* 1 *or* 0*, indicating the success or failure of the verification process, respectively.*

Our syntax extends the standard TSS model of [BPR24] with two key components:

1. **Public Value.** pub contains public identifiers and other required information. It provides the public parameters for collaborative tracing and enables public verification. With pub and their trace credentials, any coalition $\mathcal{C}$ can check others' credentials and discard false ones. Both our CTS (Section 3) and CTB (Section 4) rely on these properties of pub.
2. **Trace Credential.** Trace credentials are the core innovation enabling collaborative tracing. For each party $i \in [n]$, the dealer securely sends both the share sh_i and the trace credential φ_i. These credentials act as a distributed key: tracing works only when a coalition $\mathcal{C}$ of size t combines their credentials $\varphi_{\mathcal{C}}$. Since $|\mathcal{C}| = t$, even if the $f < t$ corrupt parties sell their trace credentials, tracing still cannot be done. Our trace credential design follows three principles:
   - *Efficient Generation.* Each trace credential φ_i must be easy to compute.
   - *Secrecy Independence.* The set $\{\varphi_i\}_{i\in[n]}$ must not reveal anything about the secret s.
   - *Efficient Validity.* Whether φ_i is valid can be checked efficiently.

   Both our CTS and CTB schemes in sections 3 and 4, respectively, follow this design rationale.

Correctness. As usual, any set of t honest parties should always reconstruct the secret. Formally, we define,

Definition 2 (Correctness). *Let $\epsilon = \epsilon(\lambda) \in [0,1]$ be a function of the security parameter λ. We say that a collaborative traceable threshold secret sharing scheme* CTSS *with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ is ϵ-correct if for every $\lambda \in \mathbb{N}$, every secret $s \in \mathcal{S}_\lambda$, every $n \in \mathbb{N}$, every t with $0 < t \leq n$, every correlation string $\rho \in \{0,1\}^\kappa$ with $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$ and every subset $\{i_1, i_2, \ldots, i_t\} \subseteq [n]$ of size t, it holds that*

$$\Pr[s' = s] \geq 1 - \epsilon(\lambda)$$

where $((\mathsf{sh}_i)_{i=1}^n, (\varphi_i)_{i=1}^n, \mathsf{pub}) \leftarrow \mathsf{Share}(1^\lambda, s, n, t, \rho)$ and $s' \leftarrow \mathsf{Recon}(\{\mathsf{sh}_{i_j}\}_{j=1}^t)$.

2.2 Security

The security definition for our collaborative model must be strengthened to account for the fact that there are no private trace keys. An adversary now has access to pub and all trace credentials $\{\varphi_i\}_{i\in[n]}$. Our security notions must ensure that secrecy is maintained even with this public information and trace credentials.

Definition 3 (Secrecy). *Let $\epsilon = \epsilon(\lambda) \in [0,1]$ be a function of the security parameter λ. A collaborative traceable threshold secret sharing scheme* CTSS *with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ satisfies* ***ϵ-secrecy*** *if for every* PPT *algorithm $\mathcal{D}$, every $\lambda \in \mathbb{N}$, any two secrets $s, s' \in \mathcal{S}_\lambda$, every $n \in \mathbb{N}$, every t with $0 < t \leq n$ and every subset $\Lambda \subseteq [n]$ of size at most $t-1$, it holds that*

$$\left|\Pr[\mathcal{D}(\mathsf{pub}, \{\mathsf{sh}_i\}_{i\in\Lambda}, \{\varphi_i\}_{i\in[n]}) = 1] - \Pr[\mathcal{D}(\mathsf{pub}', \{\mathsf{sh}'_i\}_{i\in\Lambda}, \{\varphi'_i\}_{i\in[n]}) = 1]\right| \leq \epsilon(\lambda)$$

where $\rho \xleftarrow{\$} \{0,1\}^\kappa$ with $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$, $((\mathsf{sh}_i)_{i=1}^n, (\varphi_i)_{i=1}^n, \mathsf{pub}) \leftarrow \mathsf{Share}(1^\lambda, s, n, t, \rho)$ and $((\mathsf{sh}'_i)_{i=1}^n, (\varphi'_i)_{i=1}^n, \mathsf{pub}') \leftarrow \mathsf{Share}(1^\lambda, s', n, t, \rho)$.

On Tracer Secrecy in CTSS**.** Boneh, Partap, and Rotem [BPR24] define tracer secrecy: even with the private trace keys and any $t-1$ shares, no distinguisher can learn the secret. Thus, the tracer has only a negligible advantage, even with trace keys and $t-1$ shares. We do not need this notion because tracing in our model is collaborative: any t parties can already reconstruct the secret. During tracing, parties never use their shares; only their trace credentials are used. Tracer secrecy would let a distinguisher see all credentials $\{\varphi_i\}_{i\in[n]}$, any $t-1$ shares, pub, yet still the secret remains hidden. It exactly matches our definition of secrecy (Definition 3). Thus, tracer secrecy is redundant in our CTSS setting.

2.3 Traceability and Non-Imputability

The main security goal of our work is traceability. An adversary corrupts a set $\mathcal{I}$ of $f < t$ parties and uses their shares and trace credentials to build a reconstruction box $\mathcal{R}$. This $\mathcal{R}$ is an algorithm that outputs the secret when given the missing $t-f$ shares. The tracer has only black-box access: it can give inputs and see outputs, but learns nothing about the internal code. In CTSS we even allow $\mathcal{R}$ to hard-code the corrupt parties' credentials $\varphi_\mathcal{I}$.

Definition 4 (Good Reconstruction Box). *Let $\lambda \in \mathbb{N}$, $n, t, f \in \mathbb{N}$ such that $0 < f < t \leq n$ and $\kappa = \kappa(n, t, \lambda) \in \mathbb{N}$. For $\epsilon \in [0, 1]$, a secret s, correlation string $\rho \in \{0,1\}^\kappa$, we say that a reconstruction box $\mathcal{R}$ in a collaborative traceable threshold secret sharing scheme* CTSS *with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ is $(n, t, f, s, \rho, \epsilon)$-good if*

$$\Pr\left[\mathcal{R}\left(\{\mathsf{sh}'_i\}_{i=1}^{t-f}\right) = s\right] \geq \epsilon$$

where the probability is taken over $((\mathsf{sh}'_i)_{i=1}^n, (\varphi_i)_{i=1}^n, \mathsf{pub}) \leftarrow \mathsf{Share}(1^\lambda, s, n, t, \rho)$ and random coins of $\mathcal{R}$.

With the notion of a 'good' reconstruction box, we can now define traceability. The security experiment ExpTrace is given in Figure 1.

Definition 5 (Traceability). *A collaborative traceable threshold secret sharing scheme* $\mathsf{CTSS} = (\mathsf{Share}, \mathsf{Recon}, \mathsf{Trace}, \mathsf{Verify})$ *with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ satisfies traceability if for every* PPT *adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\mathsf{ExpTrace}_{\mathcal{A},\mathsf{CTSS}}(\lambda) = 1\right]$$

is negligible in λ where ϵ, δ are functions of the security parameter λ and the experiment $\mathsf{ExpTrace}_{\mathcal{A},\mathsf{CTSS}}(\lambda, \epsilon, \delta)$ between a challenger and the adversary $\mathcal{A}$ is presented in Fig. 1.

Remark 1. In the experiment ExpTrace (Fig. 1, line 9), the challenger gives Trace a set of credentials $\varphi_\mathcal{C}$ with $|\mathcal{C}| = t$. In the experiment ExpTrace (Fig. 1, line 9), the challenger gives Trace a set of credentials $\varphi_\mathcal{C}$ with $|\mathcal{C}| = t$. Even if $\mathcal{C}$ includes corrupt parties, our efficient validity property for trace credentials ensures each credential can be checked easily. In both our Shamir- and Blakley-based CTSS schemes, any coalition $\mathcal{C} \subseteq [n]$ can verify its own credentials, allowing false or forged credentials to be detected easily.

$\mathsf{ExpTrace}_{\mathcal{A},\mathsf{CTSS},\epsilon}(\lambda)$/$\mathsf{ExpUniTrace}_{\mathcal{A},\mathsf{CTSS},\Gamma,\epsilon}(\lambda)$

1. $(n, t, \mathcal{I}) \leftarrow \mathcal{A}(1^\lambda)$
2. $s^* \xleftarrow{\$} \mathcal{S}_\lambda$ and $\rho \xleftarrow{\$} \{0,1\}^\kappa$
3. $(\mathsf{pub}, (\mathsf{sh}_i)_{i=1}^n, (\varphi_i)_{i=1}^n) \leftarrow \mathsf{Share}(1^\lambda, s^*, n, t, \rho)$
4. $\mathcal{R} \leftarrow \mathcal{A}(\mathsf{pub}, \{\mathsf{sh}_j\}_{j\in\mathcal{I}}, \{\varphi_j\}_{j\in\mathcal{I}})$
5. $\mathbf{sh} = \{\mathsf{sh}_j\}_{j\in\mathcal{I}}$
6. **if** ($\mathcal{R}$ is not $(n, t, f, s^*, \rho, \epsilon)$-good) **then return** 0. (**if** ($\mathcal{R}$ is not $(n, t, f, \mathbf{sh}, \Gamma(\rho), \epsilon)$-good) **then return** 0.)
7. $(\mathcal{I}', \pi) \leftarrow \mathsf{Trace}^\mathcal{R}(\mathsf{pub}, \varphi_\mathcal{C}, 1^{1/\delta})$
8. **if** $(\mathcal{I} = \mathcal{I}' \wedge \mathsf{Verify}(\mathsf{pub}, \mathcal{I}', \pi) \to 1)$ **then return** 0, **else return** 1

Fig. 1: Traceability and Universal traceability experiment for CTSS. The blue lines in the code are extra addition (line 5) and replacement (line 6) to Universal Traceability.

In our framework, the *correlation string* $\rho \in \{0,1\}^\kappa$ determines how shares are jointly sampled. To organize possible values of ρ, we consider a *partition*

$\Gamma = \{\mathcal{P}_1, \mathcal{P}_2, \ldots\}$ of $\{0,1\}^\kappa$; for a given ρ, let $\Gamma(\rho)$ denote the unique class containing it. For a subset $\mathcal{P} \subseteq \{0,1\}^\kappa$, let $\mathsf{sh} \xleftarrow{\$} SH_{\lambda,n,t}(\mathcal{P})$ denote the process of sampling a random share of a random secret, conditioned on $\rho \in \mathcal{P}$.

Definition 6 (Universally Good Reconstruction Box). *Let $\lambda \in \mathbb{N}$, $n, t, f \in \mathbb{N}$ such that $0 < f < t \leq n$ and $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$. For $\epsilon \in [0,1]$, f shares $\mathbf{sh} = \{\mathsf{sh}_{i_k}\}_{k=1}^{f}$ and a subset $\Gamma \subseteq \{0,1\}^\kappa$, we say that a reconstruction box $\mathcal{R}$ in a collaborative traceable threshold secret sharing scheme* CTSS *with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ is $(n,t,f,\mathbf{sh},\mathcal{P},\epsilon)$-good if*

$$\Pr\left[\mathcal{R}(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \mathsf{Recon}(\mathsf{sh}_1, \ldots, \mathsf{sh}_f, \mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f})\right] \geq \epsilon.$$

The randomness depends on $\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f} \xleftarrow{\$} SH_{\lambda,n,t}(\mathcal{P})$, and internal coin tosses of $\mathcal{R}$

Thus, as a natural follow-up to the universal good reconstruction box, we define universal traceability, given in figure 1, with the added and modified blue.

Definition 7 (Universal Traceability). *A collaborative traceable threshold secret sharing scheme* CTSS *with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ satisfies universal traceability if for every* PPT *adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{uni.trace}}_{\mathcal{A},\mathsf{CTSS},\Gamma,\epsilon}(\lambda) = \Pr\left[\mathsf{ExpUniTrace}_{\mathcal{A},\mathsf{CTSS},\Gamma,\epsilon}(\lambda) = 1\right]$$

is negligible in λ where $\Gamma = \{\mathcal{P}_1, \mathcal{P}_2, \ldots\}$ is a partition of the space $\{0,1\}^\kappa$ of correlation strings with $\Gamma(\rho) \subset \{0,1\}^\kappa$ is a unique subset $\mathcal{P}_i$ that contains $\rho \in \{0,1\}^\kappa$ with $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$ and the experiment $\mathsf{ExpUniTrace}_{\mathcal{A},\mathsf{CTSS},\Gamma,\epsilon}(\lambda)$ between a challenger and the adversary $\mathcal{A}$ is presented in Fig. 1.

This definition generalizes standard traceability by quantifying success over correlation-string classes. Looking ahead, in section 3 we will use Γ as the *trivial partition* $\Gamma = \{\{0,1\}^\kappa\}$, and in section 4 we will use Γ as the *complete partition* into singletons $\Gamma = \{\{\rho\} : \rho \in \{0,1\}^\kappa\}$. Finally, we introduce Non-imputability in CTSS. It ensures that an honest party $i^* \in [n]$ cannot be framed, even by a malicious adversary who has full access to the public parameters, all trace credentials, and all other secret shares except i^*'s.

Definition 8 (Non-Imputability). *A collaborative traceable threshold secret sharing scheme* CTSS *with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ satisfies non-imputability if for every* PPT *adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{NI}}_{\mathcal{A},\mathsf{CTSS}}(\lambda) = \Pr[\mathsf{ExpNI}_{\mathcal{A},\mathsf{CTSS}}(\lambda) = 1]$$

is negligible in λ where the experiment $\mathsf{ExpNI}_{\mathcal{A},\mathsf{CTSS}}(\lambda)$ between a challenger and the adversary $\mathcal{A}$ is described in Fig. 2.

In simpler terms, an honest party cannot be framed by crafting a bogus proof: any valid proof must have come from an actual leak.

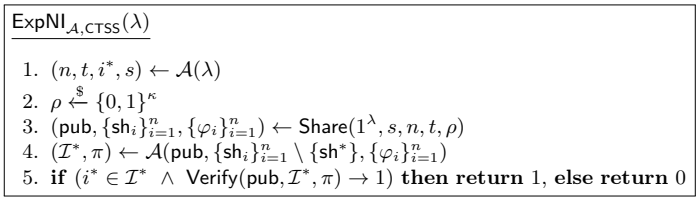

Fig. 2: Non-Imputability experiment for CTSS

If a scheme satisfies universal traceability, then it also satisfies traceability. The proof follows the same reduction as in Lemma 1 of [BPR24], ensuring that our stronger definition covers the standard one.

Theorem 1. *A collaborative traceable threshold secret sharing scheme* CTSS *which satisfies universal traceability according to the definition 7 also satisfies standard traceability according to the definition 5.*

3 Collaborative Traceable Shamir

In this section, we show how to trace leakage in a modified Shamir scheme collaboratively. We introduce a Shamir variant that fits our CTSS model (Definition 1), called "*Collaborative Traceable Shamir*" (CTS), defined by $\mathsf{CTS} = (\mathsf{CTS.Share}, \mathsf{CTS.Recon}, \mathsf{CTS.Trace}, \mathsf{CTS.Verify})$.

Use of Random Evaluation Point. Textbook Shamir often uses deterministic evaluation points (e.g., $x_i = i$). This breaks black-box tracing: a reconstruction box $\mathcal{R}$ can check whether the queried x-coordinates correspond to honest parties and output $\perp$ otherwise. A tracer, having only black-box access, would be forced to guess the correct $t - f$ honest shares to feed $\mathcal{R}$, which makes tracing infeasible. To prevent this, following [BPR24], we choose each x_i uniformly at random from $\mathbb{F}$. Over a large field, correctness is barely affected. A share (x_i, y_i) then looks like two field elements and cannot be linked to party i without extra information.

3.1 Share and Reconstruction Algorithm for CTS

CTS.Share and CTS.Recon The share algorithm CTS.Share is given in figure 3. To share a secret s, we parse the correlation string ρ to obtain random coefficients $a_1, \ldots, a_{t-1} \in \mathbb{F}$. We set the secret-sharing polynomial $p(X) = s + \sum_{j=1}^{t-1} a_i X^j$, so that $p(0) = s$. We pick a random $d \neq s$ from $\mathbb{F}$ and random coefficients $b_1, \ldots, b_t$ from $\mathbb{F}$ to define a 'trace credential generator polynomial' $q(X) = d + \sum_{j=1}^{t} b_j X^j$. Next, for each party $i \in [n]$, we pick a random field element $x_i \in \mathbb{F}$. Thinking

ahead, we use a one-way function $\mathcal{F}$ to make our scheme non-imputable. For a one-way function $\mathcal{F}$ we compute, $\mathcal{F}(x_i) = u_i$ for $i \in [n]$. Then compute the shares of the secret as $y_i = p(x_i)$, and also compute $d_i = q(u_i)$. We include $\mathsf{C}_i = \mathcal{H}(u_i \parallel d_i)$ and d in the public value pub, where $\mathcal{H}$ is a collision-resistant hash function. Finally, we form the share and trace credentials delivered to party i, respectively, as $\mathsf{sh}_i = (x_i, y_i), \varphi_i = (u_i, d_i)$. Note that knowledge of φ_i is required to link a party to its public identifier C_i. The reconstruction algorithm is the standard Shamir's reconstruction algorithm, taking into account the random choices of the x_i's. For simplicity of presentation, we present with shares numbered $1, \ldots, t$, and we assume that $x_i \neq x_j$ for every $1 \leq i < j \leq n$.

On Trace Credential in CTS. In CTS, trace credentials come from a polynomial $q(X)$ chosen to satisfy the design principles in Section 2. They are easy to generate by evaluating $q(X)$, reveal nothing about the secret since $q(X)$ is independent of the secret polynomial $p(X)$, and enable collaborative tracing because recovering $q(X)$ requires t parties. This keeps shares compact and allows for polynomial-time tracing. Similar to the additive layer used in [BPR24] for Blakley, we use a secondary Shamir-like layer to support collaborative tracing and public verification. Validity or authenticity of a trace credential can be checked via: $\mathsf{C}_i \stackrel{?}{=} \mathcal{H}(u_i \parallel d_i)$.

Why q must have degree $> t$ in CTS? The degree of the polynomial $q(X)$ should not be less than t, as if the degree is less that t then the corrupt parties can reconstruct this polynomial $q(x)$ and check the equation $\mathsf{C}_i \stackrel{?}{=} \mathcal{H}(u_i \parallel d_i)$ inside the reconstruction box and output $\perp$ for a random share (x_i, y_i), where $\mathcal{F}(x_i) = u_i, d_i = q(u_i)$.

$\underline{\mathsf{CTS.Share}(1^\lambda, s, n, t, \rho)}$

1. Parse ρ as $(a_1, \ldots, a_{t-1}) \in \mathbb{F}^{t-1}$.
2. Set $p(X) = s + a_1 X + \cdots + a_{t-1} X^{t-1}$, and thus $p(0) = s$.
3. Sample $d \xleftarrow{\$} \mathbb{F} \setminus \{s\}$ and $(b_1, \ldots, b_t) \xleftarrow{\$} \mathbb{F}^t$.
4. Set $q(X) = d + b_1 X + \cdots + b_t X^t$.
5. For $i = 1$ to n,
   - Sample $x_i \xleftarrow{\$} \mathbb{F}$ and set $\mathcal{F}(x_i) = u_i$
   - Set $y_i \leftarrow p(x_i)$ and $d_i \leftarrow q(u_i)$
   - Set $\mathsf{C}_i \leftarrow \mathcal{H}(u_i \parallel d_i)$
   - Set $\mathsf{sh}_i \leftarrow (x_i, y_i), \varphi_i \leftarrow (u_i, d_i)$
6. Set $\mathsf{pub} \leftarrow (\mathsf{C}_1, \ldots, \mathsf{C}_n; d)$
7. **return** $(\mathsf{pub}, (\mathsf{sh}_i)_{i=1}^n, (\varphi_i)_{i=1}^n)$

$\underline{\mathsf{CTS.Recon}(\mathsf{pub}, \mathsf{sh}_1, \ldots, \mathsf{sh}_t)}$

1. Parse sh_i as (x_i, y_i) for every $i = 1$ to t.
2. Compute $s \leftarrow \sum_{i=1}^t \left(\prod_{k \in [t] \setminus \{i\}} \frac{x_k}{x_k - x_i} \right) \cdot y_i$
3. **return** s.

Fig. 3: Share and Reconstruction algorithm of CTS.

Correctness of CTS. Correctness follows directly from the correctness of standard Shamir secret sharing. Since $p(X)$ is a degree $t-1$ polynomial, any t shares (x_i, y_i) with distinct x_i values can perfectly reconstruct $p(X)$ via interpolation and recover $s = p(0)$. The probability of two x_i values colliding is negligible for a sufficiently large field $\mathbb{F}$.

3.2 Tracing in CTS

We now show how to trace a reconstruction box $\mathcal{R}$ that is "good"—i.e., it outputs the secret on t-share inputs with probability ϵ. A perfect box is unrealistic; an adversary instead builds an ϵ-good box with f hard-coded shares. We will assume $\mathcal{R}$ has been fixed after seeing all shares sh_i and trace credentials φ_i for $i \in [f]$, it can not adapt during tracing, since it is a black box. When Trace queries $\mathcal{R}$, many queries fail, producing erroneous evaluation points, so standard interpolation does not work. To handle this, we rely on the *list decoding problem for Reed-Solomon codes* and use the *Guruswami-Sudan List Decoding* algorithm [Gur98], stated in A.

CTS.Trace and CTS.Verify The tracing algorithm CTS.Trace (Fig. 4) works in phases. It first reads the public value pub and the coalition's credentials $\varphi_{\mathcal{C}}$. Each credential is checked using $\mathsf{C}_i \stackrel{?}{=} \mathcal{H}(u_i \parallel d_i)$ for all $i \in \mathcal{C}$. Using d and $\varphi_{\mathcal{C}}$, it reconstructs the trace credential generator polynomial $q(X)$. It then initializes empty lists for candidate polynomials and corrupt parties. We make N random queries to $\mathcal{R}$ to cancel the influence of the secret polynomial $p(X)$. For each query ℓ, pick a random point $x'_\ell \in \mathbb{F}$ and a difference $\delta_\ell \neq 0$. We form two sets of t shares: one uses $\mathsf{sh}'_\ell = (x'_\ell, y'_\ell)$ as the last share, and the other uses $\mathsf{sh}''_\ell = (x'_\ell, y'_\ell + \delta_\ell)$. Crucially, all other $t-1$ shares (including the f shares hard-coded in $\mathcal{R}$) are held fixed. Let $s_\ell = \mathcal{R}(\mathsf{sh}_{f+1}, \ldots, \mathsf{sh}_{t-1}, \mathsf{sh}'_\ell)$ and $s'_\ell = \mathcal{R}(\mathsf{sh}_{f+1}, \ldots, \mathsf{sh}_{t-1}, \mathsf{sh}''_\ell)$. Standard interpolation shows that $s'_\ell - s_\ell = \delta_\ell \cdot h(x'_\ell)$ for some degree-f polynomial h. We compute, $z_\ell := \left(\frac{\delta_\ell}{s'_\ell - s_\ell}\right) \cdot \prod_{i=f+1}^{t-1} \frac{x_i - x'_\ell}{x_i}$, which ideally equals $h(x'_\ell)$. Collecting all pairs gives $\mathcal{L} = (x'_\ell, z_\ell)_{\ell \in [N]}$, many of which may be erroneous. We run Guruswami–Sudan list decoding (Theorem 5) to recover all degree$\leq f$ polynomials matching at least $C \geq \sqrt{fN}$ points, obtaining a list H. For each $h_j \in H$, factor h_j to find roots $w_1, \ldots, w_f$. For each root, compute $u'_i = \mathcal{F}(w_i)$ and $\mathsf{C}'_i = \mathcal{H}(u'_i \parallel q(u'_i))$. If every C'_i apears in the public list $(\mathsf{C}_1, \ldots, \mathsf{C}_n)$, then h_j is valid, ; otherwise discard it. Let h^* be the first valid polynomial, with roots $x_{i_1}, \ldots, x_{i_f}$. The trace result is $\mathcal{I} = \{i_1, \ldots, i_f\}$ and the proof $\pi = \{(\mathsf{C}_i, x_i, d_i)\}_{i \in [f]}$. Verification is public and simple: recompute $u_{i_j} = \mathcal{F}(x_{i_j})$, and check $\mathcal{H}(u_{i_j} \parallel d_{i_j}) \stackrel{?}{=} \mathsf{C}_{i_j}$ and ensure each C_{i_j} appears in pub.

3.3 Tracing and Non-imputability Advantage of CTS

CTS is Non-imputable. We give an intuition to the non-imputability of CTS. An adversary's goal is to frame an honest party $i^* \in [n]$ with access to the

$\underline{\mathsf{CTS.Trace}^{\mathcal{R}}(\mathsf{pub}, \varphi_{\mathcal{C}}, 1^{1/\epsilon})}$

1. Parse pub as $(\mathsf{C}_1, \ldots, \mathsf{C}_n; d)$, $\varphi_{\mathcal{C}} = ((u_{c_1}, d_{c_1}), \ldots, (u_{c_t}, d_{c_t}))$ for $\mathcal{C} = \{\mathsf{C}_{c_1}, \ldots, \mathsf{C}_{c_t}\}$.
2. Using $\varphi_{\mathcal{C}} \cup \{(0, d)\}$ reconstruct the polynomial $q(X)$.
3. Initialize $\mathcal{L}, \mathcal{I} \leftarrow \emptyset$.
4. For $\ell = 1$ to N
   - For $i = f+1, \ldots, t-1$: sample $x_{\ell,i}, y_{\ell,i} \xleftarrow{\$} \mathbb{F}$ and set $\mathsf{sh}_{\ell,i} = (x_{\ell,i}, y_{\ell,i})$
   - Sample $x'_\ell, y'_\ell, \delta_\ell \xleftarrow{\$} \mathbb{F}$. Set $\mathsf{sh}'_\ell \leftarrow (x'_\ell, y'_\ell)$ and $\mathsf{sh}''_\ell = (x'_\ell, y'_\ell + \delta_\ell)$.
   - Query and obtain: $s_\ell \leftarrow \mathcal{R}(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}'_\ell)$ and $s'_\ell \leftarrow \mathcal{R}(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}''_\ell)$.
   - If $s'_\ell = s_\ell$ or $\delta_\ell = 0$ or $x_{\ell,i} = x'_\ell$ for some $i \in \{f+1, \ldots, t-1\}$, or if $\ell > 1$ and $x'_\ell = x'_i$ for some $i \in [\ell - 1]$, then terminate and output $\perp$. Otherwise, output $z_\ell \leftarrow \frac{\delta_\ell}{s'_\ell - s_\ell} \prod_{i=f+1}^{t-1} \frac{x_{\ell,i}}{x_{\ell,i} - x'_\ell}$.
   - Update $\mathcal{L} \leftarrow \mathcal{L} \cup (x'_\ell, z_\ell)$.
5. Obtain a list of all candidate polynomials $H = \{h_i\}$ by running $\mathcal{D}_{\mathsf{GS}}$, where each $h_i \in H$ is of degree at most f that agree with at least C evaluation points out of $\mathcal{L}$.
6. For every $h_i \in H$: find all the roots $w_{i_1}, \ldots, w_{i_f}$ of h_i. For each root w_{i_j} where $j \in [f]$ compute $\mathcal{F}(w_{i_j}) = u'_{i_j}$, $d_{i_j} = q(u'_{i_j})$, and $\mathsf{C}_{i_j} = \mathcal{H}(u'_{i_j} \parallel d_{i_j})$. If $\mathsf{C}_{i_j} \notin \{\mathsf{C}_1, \ldots, \mathsf{C}_n\}$, then remove h_i from H.
7. If H is empty, terminate and output $\perp$. Else, h_{i^*} be the first polynomial in H with roots $x_{i_1}, \ldots, x_{i_f}$ such that the above check passes for all the roots. Compute $\mathcal{F}(x_{i_j}) = u_{i_j}$ for all $j \in [f]$.
8. Set $\mathcal{I} = \{i_1, \ldots, i_f\}$ where $\mathsf{C}_{i_j} = \mathcal{H}(u_{i_j} \parallel d_{i_j})$ for every $j \in [f]$.
9. Publish $\mathcal{I}$ and a proof $\pi = \{(\mathsf{C}_{i_j}, x_{i_j}, d_{i_j})\}_{j \in [f]}$
10. **return** $(\mathcal{I}, \pi)$

$\underline{\mathsf{CTS.Verify}(\mathsf{pub}, \mathcal{I}, \pi)}$

1. Parse pub as $(\mathsf{C}_1, \ldots, \mathsf{C}_n; d)$.
2. Parse $\mathcal{I}$ as $\{i_1, \ldots, i_f\}$ and π as $\{(\mathsf{C}_{i_j}, x_{i_j}, d_{i_j})\}_{j \in [f]}$
3. For $j = 1$ to f
   - Compute $\mathcal{F}(x_{i_j}) = u_{i_j}$
   - Check: $\mathsf{C}_{i_j} \stackrel{?}{=} \mathcal{H}(u_{i_j} \parallel d_{i_j}) \ \wedge \ \mathsf{C}_{i_j} \stackrel{?}{\in} \{\mathsf{C}_1, \ldots, \mathsf{C}_n\}$
4. If Check pass for all $j \in [f]$, **return** 1, else **return** 0.

Fig. 4: Trace and Verification Algorithm for CTS

public value pub, all the shares except i^*'s, i.e., $\{\mathsf{sh}_i\}_{i \in [n] \setminus \{i^*\}}$, and all the trace credentials $\{\varphi_i\}_{i \in [n]}$. Note that, $\varphi_i = (u_i, d_i)$ where $u_i = \mathcal{F}(x_i)$ and $d_i = q(u_i)$. The public identifiers will be $\mathsf{C}_i = \mathcal{H}(u_i \parallel d_i)$ for $i \in [n]$. To frame party i^*, it has to include $(\mathsf{C}_{i^*}, x^*, d_{i^*})$ in the proof π so that $\mathcal{F}(x^*) = u_{i^*}$. Finding such x^* is computationally infeasible due to the oneway-ness of $\mathcal{F}$. We formally state the tracing advantage and non-imputability advantage theorem; the proof is postponed in section B.

Theorem 2. *Let* CTS *be a t-out-of-n collaborative traceable Shamir secret sharing scheme defined in the figures 3, 4. Let $\mathcal{H}$ be a collision-resistant hash function modeled as a random oracle and $\mathcal{F}$ be a one-way function. Then,*

1. *For every adversary $\mathcal{A}$ playing the universal tracing game for* CTS*, for every* $\lambda \in \mathbb{N}$*, for every* $N, C \in \mathbb{N}$ *and* $\epsilon \in \left[\sqrt{\frac{2(N+f+t^2)}{p}}, 1\right]$ *satisfying* $\epsilon \geq \sqrt{\frac{2C}{N}}$ *and* $\sqrt{fN} \leq C \leq N < p$*, it holds that*

$$\mathsf{Adv}^{\mathsf{uni.trace}}_{\mathcal{A},\mathsf{CTS},\epsilon}(\lambda) \leq e^{\frac{\epsilon^2 N}{2}\cdot(1-1/r)^2} + \frac{f \cdot n \cdot \tau}{p} + \mathsf{negl}(\lambda)$$

where $p = |\mathbb{F}|$*,* $r = \frac{\epsilon^2 N}{2C}$*,* n *be the total number of parties,* f *be the number of corruptions,* $\tau = \tau(N, f, \log p)$ *be polynomial bound on Guruswami-Sudan list size, and* negl *be a negligible function.*
2. *For every adversary $\mathcal{A}$ attacking non-imputability of* CTS*, there exists an adversary $\mathcal{B}$ such that for every* $\lambda \in \mathbb{N}$*, the following holds true,*

$$\mathsf{Adv}^{\mathsf{NI}}_{\mathcal{A},\mathsf{CTS}}(\lambda) \leq \Pr[\mathcal{F}(\mathcal{B}(\mathcal{F}(x^*))) = \mathcal{F}(x^*)]$$

where $x \xleftarrow{\$} \mathbb{F}$ *and randomness depends on the internal coin tosses of* $\mathcal{B}$

Learning f. When the tracing algorithm does not know the number of corrupt parties f^*, it tests f from $t-1$ downward. If $f > f^*$, any traced set of size f includes an honest party with probability at most $n^2\tau/p$. For $f = f^*$, by part 1 of Theorem 2. Failure probability is at most $e^{-\lambda} + n^2\tau/p$. By a union bound, the correct set is identified with probability at least $1 - (e^{-\lambda} + 2n^3\tau/p)$.

4 Collaborative Traceable Blakley

We recall the Blakley$^+$ construction from Section A.2, on which we build collaborative tracing. Our main focus is to make this extended Blakely Secret Sharing collaboratively traceable. In order to do that, we propose "*collaborative traceable Blakley*" (CTB), which is equipped with another Shamir Secret Sharing that distributes some randomness among the parties, which is used to hide the information of the hyperplanes passing through $\boldsymbol{r_1}$. In the private tracing proposed by [BPR24] the private trace-key is of the form $\{\boldsymbol{r_2}, \ldots, \boldsymbol{r_t}; (\boldsymbol{a_{1,1}}, b_{1,1}), \ldots, (\boldsymbol{a_{n,1}}, b_{n,1})\}$. Note that if this trace-key is public, then the reconstruction box $\mathcal{R}$ can output $\perp$ any time the tracer queries the box with a random tuple $(\boldsymbol{a'}, b')$ corresponding to $\boldsymbol{r_1}$ while tracing. So the goal is hide $\{(\boldsymbol{a_{i,1}}, b_{i,1})\}_{i\in[n]}$ and prevent the box to abort. To achieve this, we assume a cyclic group $\mathbb{G}$ with $|\mathbb{G}| = p$ and $g \in \mathbb{G}$ is the generator. Then the set $\{(g^{\boldsymbol{a_{i,1}}}, g^{b_{i,1}})\}_{i\in[n]}$ hides $\{(\boldsymbol{a_{i,1}}, b_{i,1})\}_{i\in[n]}$ perfectly due to the discrete logarithm assumption. We run a setup algorithm $\mathcal{G}(1^\lambda)$ at the beginning of the share algorithm to generate the public parameters $(\mathbb{G}, g, p)$.

4.1 Share and Reconstruction algorithm for CTB

CTB.Share and CTB.Recon The share algorithm takes the security parameter λ, a secret $\boldsymbol{s} \in \mathbb{F}_p^t$, n, t and a correlation string ρ and outputs shares $\{\mathsf{sh}_i\}_{i\in[n]}$, trace credentials $\{\varphi_i\}_{i\in[n]}$, a public information pub. The sh_i and φ_i are given

to party i and pub is public. We choose a secret random scalar $d \xleftarrow{\$} \mathbb{F}_p$ and use Shamir secret sharing to generate the shares φ_i. Then d is used to mask the information $\{(\boldsymbol{a_{i,1}}, b_{i,1})\}_{i\in[n]}$ and then these are raised to the power of $g \in \mathbb{G}$. We assume a collision-resistant hash function $\mathcal{H}$ to generate a public identifier C_i of the trace credential φ_i for all $i \in [n]$. The reconstruction algorithm is the standard Blakeley's reconstruction. For simplicity of presentation, we present with shares numbered $1, \ldots, t$. With t shares, one gets a consistent system of t linear equations for the indeterminate $\boldsymbol{r_j}$, for each $j \in [t]$. Solving these t systems one can get the vectors $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t}$. A detailed algorithm is described in 5. The following theorem establishes the secrecy of CTB. The proof is deferred to Section C.

$\underline{\mathsf{CTB.Share}(1^\lambda, \boldsymbol{s}, n, t, \rho)}$

1. Run $(\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}$.
2. Parse ρ as $(\boldsymbol{r_2}, \ldots, \boldsymbol{r_t}) \in \mathbb{F}_p^t$.
3. Set $\boldsymbol{r_1} \leftarrow \boldsymbol{s} - (\boldsymbol{r_2} + \ldots + \boldsymbol{r_t})$
4. For $i = 1$ to n,
5. For $j = 1$ to t
   - Sample $\boldsymbol{a_{i,j}} \xleftarrow{\$} \mathbb{F}_p^t$
   - Set $b_{i,j} \leftarrow \langle \boldsymbol{a_{i,j}}, \boldsymbol{r_j} \rangle$
6. Set $\mathsf{sh}_i \leftarrow \{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j=1}^t$ for $i \in [n]$.
7. Sample $d \xleftarrow{\$} \mathbb{F}_p$ and $q_1, \ldots, q_{t-1} \xleftarrow{\$} \mathbb{F}_p^{t-1}$.
8. Set $q(X) = d + q_1 X + \ldots + q_{t-1} X^{t-1}$
9. Sample $x_i \xleftarrow{\$} \mathbb{F}_p$ without replacement for all $i \in [n]$.
10. Set $y_i \leftarrow q(x_i)$, $\varphi_i \leftarrow (x_i, y_i)$ and $\mathsf{C}_i \leftarrow \mathcal{H}(\varphi_i)$ for all $i \in [n]$.
11. For $i = 1$ to n
    - Set $\boldsymbol{d_i} \leftarrow \boldsymbol{a_{i,1}} + \boldsymbol{d}$, where $\boldsymbol{d} = (d, d, \ldots, d) \in \mathbb{F}_p^t$
    - Set $e_i \leftarrow b_{i,1} + d$
12. Set $g^{\boldsymbol{d_i}} = (g^{d_i^1}, \ldots, g^{d_i^t})$ and $\boldsymbol{d_i} = (d_i^1, \ldots, d_i^t)$ for all $i \in [n]$.
13. Set $\mathsf{pub} \leftarrow \big((\mathbb{G}, g, p), \boldsymbol{r}_2, \ldots, \boldsymbol{r}_t, (g^{\boldsymbol{d_1}}, g^{e_1}), \ldots, (g^{\boldsymbol{d_n}}, g^{e_n}), \mathsf{C}_1, \ldots, \mathsf{C}_n\big)$.
14. **return** $(\mathsf{pub}, (\mathsf{sh}_i)_{i=1}^n, (\varphi_i)_{i=1}^n)$

$\underline{\mathsf{CTB.Recon}(\mathsf{sh}_1, \ldots, \mathsf{sh}_t)}$

1. Parse sh_i as $\{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j=1}^t$ for every $i = 1$ to t.
2. For $j = 1, \ldots, t$: Consider the system of affine equations $\{\langle \boldsymbol{a_{i,j}}, \boldsymbol{R_j} \rangle = b_{i,j}\}_{i\in[t]}$ in the indeterminates $\boldsymbol{R}_j = (R_{j,1}, \ldots, R_{j,t})$. This is a system of t equations in t variables. If the equations are linearly independent, compute the unique solution $\boldsymbol{R_j} = \boldsymbol{r_j}$.
   Otherwise, if there is more than one solution to the system, terminate and output $\perp$.
3. Compute $\boldsymbol{s} \leftarrow \boldsymbol{r_1} + \boldsymbol{r_2} + \ldots + \boldsymbol{r_t}$
4. **return** $\boldsymbol{s}$.

Fig. 5: Share and Reconstruction Algorithm for CTB

Theorem 3 (Secrecy of CTB). *Let λ be the security parameter, and let n, t be parameters such that $0 < t < n$. Under the Discrete Log Relation (DLR)*

assumption in group $\mathbb{G}$ of order p, the CTB *scheme described in Algorithms 5 satisfies ϵ-secrecy (Definition 3) for $\epsilon = \frac{t+1}{p-1} + \mathsf{negl}(\lambda)$.*

4.2 Tracing in CTB

The collaborative tracing of CTB requires a collaboration of t parties. The reason behind this is to reconstruct the randomness d and mask the public information, thereby recovering the original information relevant for tracing. Let $\mathcal{C} \subseteq [n]$ be the set of parties that collaborate during tracing and $|\mathcal{C}| = t$. The tracer gathers the trace credentials by collecting shares of d from the parties in $\mathcal{C}$, and we define $\varphi_{\mathcal{C}} := \{\varphi_i : i \in \mathcal{C}\}$. Note that no party from the set $\mathcal{C}$ can cheat by sending false shares because the tracer can always check $\mathsf{C}_i \stackrel{?}{=} \mathcal{H}(\varphi_i)$ for all $i \in \mathcal{C}$, as the C_i are public. First we parse the public information pub as $\{\boldsymbol{r_2}, \ldots, \boldsymbol{r_t}; (g^{\boldsymbol{d_1}}, g^{e_1}), \ldots, (g^{\boldsymbol{d_n}}, g^{e_n}); (\mathsf{C}_1, \ldots, \mathsf{C}_n)\}$. The first part of pub is used to probe the box $\mathcal{R}$ with $t - f$ shares and compute the vector $\boldsymbol{r_1^*}$. The second part is relevant to finding the corrupt parties after demasking with d, and the third part is required to detect false credentials in $\varphi_{\mathcal{C}}$.

CTB.Trace and CTB.Verify The trace algorithm takes public value pub, a set of t trace credentials $\varphi_{\mathcal{C}}$ and an error parameter as ϵ input and outputs a set of corrupt parties $\mathcal{I}$ of size f and a proof π. Tracer first queries the reconstruction box $\mathcal{R}$ with $t-f$ hyper planes passing through $\boldsymbol{r_j}$ for each $j = 2, \ldots, t$ and $t-f$ random hyperplanes. Let $\mathcal{R}$ outputs $\boldsymbol{s^*}$ and tracer computes $\boldsymbol{r_1^*} = \boldsymbol{s^*} - (\boldsymbol{r_2} + \ldots + \boldsymbol{r_t})$. With access to the trace credentials in $\varphi_{\mathcal{C}}$, the tracer reconstructs d using Lagrange interpolation. From the pub tracer computes $(g^{\boldsymbol{a_{i1}}}, g^{b_{i1}})$ $\forall i \in [n]$ in following way: $g^{\boldsymbol{a_{i,1}}} = g^{\boldsymbol{d_i}} \cdot g^{-\boldsymbol{d}}$ and $g^{b_{i1}} = g^{e_i} \cdot g^{-d}$ where $\boldsymbol{d} = (d, \ldots, d)$ is a t tuple vector. The tracer detects a party i of corruption if $\prod_{j=1}^{t} (g^{a_{i,1}^j})^{r_1^j} = g^{b_{i,1}}$, where $\boldsymbol{a_{i,1}} = (a_{i,1}^j)_{j=1}^t$ and $\boldsymbol{r_1^*} = (r_1^j)_{j=1}^t$. The computation we have described is just computing the inner product $\langle \boldsymbol{a_{i,1}}, \boldsymbol{r_1^*} \rangle$ in the exponent of g. This is similar to checking which hyperplane passes through $\boldsymbol{r_1^*}$ and determining the corrupt party based on that hyperplane. The verification algorithm takes the public value pub, a set of corrupt parties $\mathcal{I} \subseteq [n]$, and a proof π, and outputs 0 or 1 to reject or accept the proof, respectively. In the figure 6, algorithmic details are provided. The following theorem establishes tracing advantage and non-imputability; proof of the theorem is deferred to section C.

Theorem 4 (Tracing and Non-imputability advantage). *Let* CTB *be a t-out-of-n collaborative traceable Blakley secret sharing scheme defined in the figures 5, 6. Then,*

1. *For every adversary $\mathcal{A}$, for every $\lambda \in \mathbb{N}$ and $\epsilon \in [0, 1]$ it holds that*

$$\mathsf{Adv}^{\mathsf{uni.trace}}_{\mathcal{A},\mathsf{CTB},\Gamma,\epsilon}(\lambda) \leq e^{-\lambda} + \frac{2n\lambda}{\epsilon p} + \mathsf{negl}(\lambda)$$

$\mathsf{CTB.Trace}^{\mathcal{R}}(\mathsf{pub}, \varphi_{\mathcal{C}}, 1^{1/\epsilon})$

1. Parse pub as $\big((\mathbb{G}, g, p), \boldsymbol{r}_2, \ldots, \boldsymbol{r}_t, (g^{\boldsymbol{d_1}}, g^{e_1}), \ldots, (g^{\boldsymbol{d_n}}, g^{e_n}), \mathsf{C}_1, \ldots, \mathsf{C}_n\big)$.
2. For $\ell = 1, \ldots, 2\lambda/\epsilon$
   (a) Sample $t - f$ shares to feed to $\mathcal{R}$. For $i = 1, \ldots, t - f$
      i. For $j = 2, \ldots, t$: sample $\boldsymbol{a'_{i,j}} \xleftarrow{\$} \mathbb{F}_p^t$ and set $b'_{i,j} \leftarrow \langle \boldsymbol{a'_{i,j}}, \boldsymbol{r}_j \rangle$.
      ii. For $j = 1$: $\boldsymbol{a'_{i,1}} \xleftarrow{\$} \mathbb{F}_p^t$ and $b'_{i,1} \xleftarrow{\$} \mathbb{F}_p$
      iii. Set $\mathsf{sh}'_i = \{(\boldsymbol{a'_{i,j}}, b'_{i,j})\}_{j=1}^t$
   (b) Query $\mathcal{R}$ on $(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f})$ and get back some secret $\boldsymbol{s}^* \in \mathbb{F}_p^t$.
   (c) Compute $\boldsymbol{r}_1^* \leftarrow \boldsymbol{s}^* - (\boldsymbol{r}_2 + \ldots + \boldsymbol{r}_t)$
   (d) Use Lagrange interpolation on the elements in $\varphi_{\mathcal{C}}$ and reconstruct the secret d.
   (e) Compute $g^{\boldsymbol{a_{i,1}}} = g^{\boldsymbol{d_i}} \cdot g^{-\boldsymbol{d}}$ and $g^{b_{i,1}} = g^{e_i} \cdot g^{-d}$, for all $i \in [n]$.
   (f) Parse $\boldsymbol{r}_1^*$ as $(r_1^1, \ldots, r_1^t)$ and $g^{\boldsymbol{a_{i,1}}}$ as $(g^{a_{i,1}^1}, \ldots, g^{a_{i,1}^t})$ $\forall i \in [n]$.
   (g) Compute a set $\mathcal{I} \subseteq [n]$ as the set of indices $i \in [n]$ for which the following check holds : $\prod_{j=1}^{t} (g^{a_{i,1}^j})^{r_1^j} \stackrel{?}{=} g^{b_{i,1}}$. Set $\pi \leftarrow (\boldsymbol{r}_1^*, d)$.
   (h) If $|\mathcal{I}| = f$, output $\mathcal{I}, \pi$ and terminate.
3. If reached, output ϕ.

$\mathsf{CTB.Verify}(\mathsf{pub}, \mathcal{I}, \pi)$

1. Parse pub as $\big((\mathbb{G}, g, p), \boldsymbol{r}_2, \ldots, \boldsymbol{r}_t, (g^{\boldsymbol{d_1}}, g^{e_1}), \ldots, (g^{\boldsymbol{d_n}}, g^{e_n}), \mathsf{C}_1, \ldots, \mathsf{C}_n\big)$.
2. Parse $\mathcal{I}$ as $(i_1, \ldots, i_f)$ and π as $(\boldsymbol{r}_1^*, \boldsymbol{d})$.
3. Parse $\boldsymbol{r_1^*}$ as $(r_1^1, r_1^2, \ldots, r_1^t)$.
4. Compute $g^{\boldsymbol{a_{i,1}}} = g^{\boldsymbol{d_i}} \cdot g^{-\boldsymbol{d}}$ and $g^{b_{i,1}} = g^{e_i} \cdot g^{-d}$, for all $i \in [n]$, where $\boldsymbol{d} = (d, \ldots, d) \in \mathbb{F}_p^t$.
5. For each $i \in \mathcal{I}$
   – Check: $\prod_{j=1}^{t} (g^{a_{i,1}^j})^{r_1^j} \stackrel{?}{=} g^{b_{i,1}}$
6. If the check pass for all $i \in \mathcal{I}$, **return** 1.
7. Else **return** 0.

Fig. 6: Trace and verification Algorithm for CTB

where $n = n(\lambda)$ is an upper bound on the number of parties, and assuming $p \geq \frac{2n}{\epsilon}$

2. *For every adversary $\mathcal{A}$ for the non-imputabiltiy of* CTB*, there exists an algorithm $\mathcal{B}$ for the* DLR *problem, such that for every $\lambda \in N$, it holds that*

$$\mathcal{A}^{\mathsf{ni}}_{\mathcal{A},\mathsf{CTB}}(\lambda) = \mathcal{A}^{\mathsf{DLR}}_{\mathcal{A},\mathcal{G}}(\lambda).$$

5 Complexity Comparison

Goyal, Song, and Srinivasan (GSS) [GSS21] provided the first publicly traceable Shamir-based scheme, but its share size increases by a factor of *secpara*. Subsequent works, T-Shamir and T-Blakley by Boneh, Partap, and Rotem [BPR24], Mignotte's MTSS and its semi-public variant SPMTSS [Hof24], and TBUSS [HKMP25] achieve far better efficiency with only a two-fold share-size increase. However, all provide only *private* traceability or verification, and MTSS further suffers from exponential-time tracing.

Our CTS and CTB schemes introduce *collaborative* traceability and support *public* verification, while maintaining a share size that is doubled, matching the best-known overhead. Tracing runs in time polynomial in λ and $1/\epsilon$: CTS.Trace

uses Guruswami–Sudan decoding with $N = O(1/\epsilon^2)$ queries; CTB.Trace performs $\ell = 2\lambda/\epsilon$ polynomial-time group operations. A comparison table with the existing work is provided, where $|\mathsf{sh}|$ stands for share size; 'Any t?' means if the threshold t is unrestricted; 'semi' denotes public tracing and private verification, and 'collab' denotes collaborative tracing.

Traceable SS scheme	$\|\mathsf{sh}\|$ **increase**	Trace **Complexity**	**Public?**	**Any** t**?**
GSS [GSS21]	λ	$\mathrm{poly}(\lambda, \epsilon^{-1})$	yes	yes
T-Shamir [BPR24]	2	$\mathrm{poly}(\lambda, \epsilon^{-1})$	no	yes
T-Blakely [BPR24]	2	$\mathrm{poly}(\lambda, \epsilon^{-1})$	no	yes
MTSS [Hof24]	2	$\mathrm{poly}(2^{\kappa}, \epsilon^{-1})$	no	no
SPMTSS [Hof24]	2	$\mathrm{poly}(\lambda, \epsilon^{-1})$	semi	no
TBUSS [HKMP25]	2	$\mathrm{poly}(\lambda, \epsilon^{-1})$	no	yes
CTS [This Work]	2	$\mathrm{poly}(\lambda, \epsilon^{-1})$	collab	yes
CTB [This Work]	2	$\mathrm{poly}(\lambda, \epsilon^{-1})$	collab	yes

6 Conclusion and Future Direction

This paper introduced Collaborative Traceable Secret Sharing (CTSS), which replaces centralized, private tracing and verification with collaborative tracing and public verification. We formalized this model, providing new syntax and robust security definitions, and built efficient, non-imputable schemes: Collaborative Traceable Shamir (CTS) and Collaborative Traceable Blakley (CTB), that preserve compact share size while removing the need for a designated tracer. By introducing trace credentials and utilizing techniques such as list decoding and discrete-log-based masking, we preserve the compact share size of prior schemes while eliminating the single point of failure and dependency inherent in private and semi-public models.

Future work includes extending collaborative tracing to other secret-sharing schemes, such as Mignotte's scheme [Hof24] and the traceable Asmuth–Bloom construction [LD25], thereby covering Chinese Remainder Theorem-based schemes. It is also promising to explore collaborative variants of traceable threshold encryption and related primitives. Another interesting direction will be to use CTSS as a compiler for Traceable secret sharing schemes. Ultimately, achieving fully public tracing requiring neither private trace keys nor collaboration remains a challenging open problem.

Acknowledgment Rittwik Hajra is supported by MeitY sponsored project titled "*Information Security Education and Awareness (ISEA) Project Phase III*" at the R. C. Bose Centre for Cryptology & Security, Indian Statistical Institute.

References

BBF+25. Rishiraj Bhattacharyya, Jan Bormet, Sebastian Faust, Pratyay Mukherjee, and Hussien Othman. CCA-secure traceable threshold (ID-based) encryption and application. Cryptology ePrint Archive, Paper 2025/341, 2025.

BDF+25. Jan Bormet, Stefan Dziembowski, Sebastian Faust, Tomasz Lizurej, and Marcin Mielniczuk. Strong secret sharing with snitching. Cryptology ePrint Archive, Paper 2025/1119, 2025.

BEMS25. Karim Baghery, Ehsan Ebrahimi, Omid Mirzamohammadi, and Mahdi Sedaghat. Traceable verifiable secret sharing and applications. Cryptology ePrint Archive, Paper 2025/318, 2025.

BF99. Dan Boneh and Matthew Franklin. An efficient public key traitor tracing scheme. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 338–353, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

BHO25. Jan Bormet, Jonas Hofmann, and Hussien Othman. Traceable threshold encryption without trusted dealer. Cryptology ePrint Archive, Paper 2025/342, 2025.

BLA79. G. R. BLAKLEY. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.

BN08. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, page 501–510, New York, NY, USA, 2008. Association for Computing Machinery.

BPR24. Dan Boneh, Aditi Partap, and Lior Rotem. Traceable secret sharing: Strong security and efficient constructions. Cryptology ePrint Archive, Paper 2024/405, 2024.

BPR25. Dan Boneh, Aditi Partap, and Lior Rotem. Traceable verifiable random functions. Cryptology ePrint Archive, Paper 2025/312, 2025.

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 573–592, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

BZ17. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79(4):1233–1285, December 2017.

CFNP00. B. Chor, A. Fiat, M. Naor, and B. Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

CVW+18. Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from lwe made simple and attribute-based. In *Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part II*, page 341–369, Berlin, Heidelberg, 2018. Springer-Verlag.

DFLM24. Stefan Dziembowski, Sebastian Faust, Tomasz Lizurej, and Marcin Mielniczuk. Secret sharing with snitching. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 840–853, New York, NY, USA, 2024. Association for Computing Machinery.

FG25. Oriol Farràs and Miquel Guiot. Traceable secret sharing schemes for general access structures. Cryptology ePrint Archive, Paper 2025/1120, 2025.

FT99. Amos Fiat and Tamir Tassa. Dynamic traitor training. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 354–371. Springer, 1999.

GJP25. Vipul Goyal, Abhishek Jain, and Aditi Partap. Traceable secret sharing revisited. Cryptology ePrint Archive, Paper 2025/1980, 2025.

GKMN25. Tiantian Gong, Aniket Kate, Hemanta K. Maji, and Hai H. Nguyen. Disincentivize collusion in verifiable secret sharing. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025*, pages 34–64, Cham, 2025. Springer Nature Switzerland.

GKSW10. Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, page 121–130, New York, NY, USA, 2010. Association for Computing Machinery.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 660–670, New York, NY, USA, 2018. Association for Computing Machinery.

Gol01. Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

GSS21. Vipul Goyal, Yifan Song, and Akshayaram Srinivasan. Traceable secret sharing and applications. In *Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part III*, page 718–747, Berlin, Heidelberg, 2021. Springer-Verlag.

Gur98. Venkatesan Guruswami. Improved decoding of reed-solomon and algebraic-geometric codes. volume 45, pages 28–37, 12 1998.

HKMP25. Rittwik Hajra, Subha Kar, Pratyay Mukherjee, and Soumit Pal. Traceable bottom-up secret sharing and law & order on community social key recovery (full version). Cryptology ePrint Archive, Paper 2025/2089, 2025.

Hof24. Charlotte Hoffmann. Traceable secret sharing based on the chinese remainder theorem. Cryptology ePrint Archive, Paper 2024/811, 2024.

KY01. Aggelos Kiayias and Moti Yung. Self protecting pirates and black-box traitor tracing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 63–79, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

LD25. Maria Leslie and Ratna Dutta. A traceable threshold asmuth–bloom secret sharing scheme. Cryptology ePrint Archive, Paper 2025/1561, 2025.

NNL01. Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. Cryptology ePrint Archive, Paper 2001/059, 2001.

NP98. Moni Naor and Benny Pinkas. Threshold traitor tracing. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 502–517, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

Sha79. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

SNW00. Reihaneh Safavi-Naini and Yejing Wang. Sequential traitor tracing. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 2000.

Wee20. Hoeteck Wee. Functional encryption for quadratic functions from k-lin, revisited. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I*, page 210–228, Berlin, Heidelberg, 2020. Springer-Verlag.

Zha20. Mark Zhandry. New techniques for traitor tracing: Size $n^{1/3}$ and more from pairings. Cryptology ePrint Archive, Paper 2020/954, 2020.

A Traceable Shamir and Traceable Blakley [BPR24]

A.1 Traceable Shamir (TS) [BPR24]

We discuss Trceable Shamir construction in the figure 7

$\mathsf{TS.Share}(1^\lambda, s, n, t, \rho) \to ((\mathsf{sh}_i)_{i\in[n]}, (\mathsf{tk}_i)_{i\in[n]}, (\mathsf{vk}_i)_{i\in[n]})$

1. Parse ρ as $(a_1, \ldots, a_{t-1}) \in \mathbb{F}^{t-1}$ and set $p(X) = s + a_1X + \cdots + a_{t-1}X^{t-1}$, and thus $p(0) = s$.
2. For $i = 1$ to n,
   - Sample $x_i \xleftarrow{\$} \mathbb{F}$
   - Compute $y_i \leftarrow p(x_i)$
   - $\mathsf{sh}_i \leftarrow (x_i, y_i), \varphi_i \leftarrow (x_i, d_i)$
3. Set $\mathsf{sh}_i \leftarrow (x_i, y_i), \mathsf{tk}_i \leftarrow x_i, \mathsf{vk}_i \leftarrow x_i$
4. **return** $((\mathsf{sh}_i)_{i\in[n]}, (\mathsf{tk}_i)_{i\in[n]}, (\mathsf{vk}_i)_{i\in[n]})$.

$\mathsf{TS.Recon}(\mathsf{sh}_1, \ldots, \mathsf{sh}_t) \to s$

1. Parse sh_i as (x_i, y_i) for every $i = 1$ to t.
2. Compute $s \leftarrow \sum_{i=1}^{t} \left(\prod_{k\in[t]\setminus\{i\}} \frac{x_k}{x_k - x_i} \right) \cdot y_i$
3. **return** s.

Fig. 7: Share and reconstruction algorithm of TS.

A.2 Traceable Blakley (TB) [BPR24]

In this scheme, the work of [BPR24] took a detour from the classical Blakley secret sharing by first encode the secret as a vector in $\mathbb{F}_p^t$ and then additively share the secret $\boldsymbol{s}$ in t vectors in $\mathbb{F}_p^t$, and finally sharing each of these t additive share vectors among n parties. They call it Blakley$^+$. For the notational convenience, we denote the traceable Blakley$^+$ scheme as TB.

A.3 Reed Solomon List Decoding and Guruswami-Sudan Algorithm

Definition 9 (The Reed-Solomon List Decoding Problem). *Let $\mathbb{F}$ be a finite field, and let $k, N, C \in \mathbb{N}$ such that $C \leq N < |\mathbb{F}|$. Given k, C and N pairs of elements $\{(x_i, z_i)\}_{i\in[N]} \in \mathbb{F}^2$, output a list H of univariate polynomials of degree at most k that agree with at least C pairs $\{(x_i, z_i)\}$. In particular,*

$$H = \{h \in \mathbb{F}[X] \mid \deg(h) \leq k \ \wedge \ |\{j \in [N] \mid h(x_j) = z_j\}| \geq C\}$$

The list-decoding algorithms output a list of all polynomials of a certain degree that agree with a pre-determined fraction of the evaluation points. Now, we introduce the Guruswami-Sudan list decoding algorithm.

$\mathsf{TB.Share}(1^\lambda, \boldsymbol{s}, n, t, \rho) \to ((\mathsf{sh}_i)_{i\in[n]}, (\mathsf{tk}_i)_{i\in[n]}, (\mathsf{vk}_i)_{i\in[n]})$

1. Parse ρ as $\boldsymbol{r_2}, \ldots, \boldsymbol{r_t} \in \mathbb{F}_p^t$.
2. Set $\boldsymbol{r_1} \leftarrow \boldsymbol{s} - (\boldsymbol{r_2} + \cdots + \boldsymbol{r_t})$.
3. For $i = 1$ to n,
4. For $j = 1$ to t
   - Sample $\boldsymbol{a_{i,j}} \xleftarrow{\$} \mathbb{F}_p^t$
   - Set $b_{i,j} \leftarrow \langle \boldsymbol{a_{i,j}}, \boldsymbol{r_j} \rangle$
5. Set $\mathsf{sh}_i \leftarrow \{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j=1}^t$ for $i \in [n]$.
6. **return** $(\mathsf{sh}_i)_{i\in[n]}$

$\mathsf{TB.Recon}(\mathsf{sh}_1, \ldots, \mathsf{sh}_t) \to s$

1. Parse sh_i as $\{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j=1}^t$ for every $i = 1$ to t.
2. For $j = 1, \ldots, t$: Consider the system of affine equations $\{\langle \boldsymbol{a_{i,j}}, \boldsymbol{R_j} \rangle = b_{i,j}\}_{i\in[t]}$ in the indeterminates $\boldsymbol{R}_j = (R_{j,1}, \ldots, R_{j,t})$. This is system of t equations in t variables. If the equations are linearly independent, compute the unique solution $\boldsymbol{R_j} = \boldsymbol{r_j}$.
   Otherwise, if there is more than one solution to the system, terminate and output $\perp$.
3. Compute $\boldsymbol{s} \leftarrow \boldsymbol{r_1} + \boldsymbol{r_2} + \ldots + \boldsymbol{r_t}$
4. **return** $\boldsymbol{s}$.

Fig. 8: Share and reconstruction algorithm of TB.

Theorem 5 (Guruswami-Sudan Algorithm [Gur98]). *Let $\mathbb{F}$ be a finite field, and let $k, N, C \in \mathbb{N}$ such that $C \le N < |\mathbb{F}|$ and $C \ge \sqrt{kN}$. Then, there exists an algorithm $\mathcal{D}_{\mathsf{GS}}$ that solves the list decoding problem as defined in Definition 9 in time polynomial in N, k and $\log |\mathbb{F}|$.*

Since the runtime of the algorithm is polynomial, the list H it outputs will be of polynomial length. We denote $\tau(N, k, \log |\mathbb{F}|)$ to be the polynomial bounding the list length, i.e., $|H| \le \tau$, where the input to the τ polynomial is understood from the context.

B Proof of Theorem 2

Proof. 1. Consider an adversary $\mathcal{A}$ in the universal traceability experiment $\mathsf{ExpUniTrace}_{\mathcal{A},\mathsf{CTS},\epsilon}$. Let $\mathcal{I}$ be the corrupted set of size f with shares $\{(x_j^*, y_j^*)\}_{j=1}^f$ and the public value is $\mathsf{pub} = (\mathsf{C}_1, \ldots, \mathsf{C}_n; d)$. Let G be the event that the adversary outputs a reconstruction box $\mathcal{R}$ that is $(n, t, f, \mathbf{sh}, \epsilon)$-good where n and t are chosen by the adversary and $\mathbf{sh}$ is given by the challenger. Conditioned on $\neg\mathsf{G}$, the output of the experiment is 0 with probability 1, so we condition the rest of the analysis on G. Let us denote $\mathsf{sh}_{i_j} = (x_j^*, y_j^*)$ for all $j \in [f]$, and let $\mathcal{I}$ be the set of parties corrupted by $\mathcal{A}$, i.e., $\{\mathsf{C}_j^*\}_{j\in[|\mathcal{I}|]} = \{\mathsf{C}_i\}_{i\in\mathcal{I}}$. Let us consider the following events, for $\ell = 1, \ldots, N$, let us define the following bad events:

(a) $\mathsf{E}_{\delta,\ell} : \delta_\ell = 0$ for $\ell = 1, \ldots, N$
(b) $\mathsf{E}_{x,j,\ell,1} : x_{\ell,j} = x'_\ell$ for some $j \in \{f+1, \ldots, t\}$ and $\ell \in [N]$
(c) $\mathsf{E}_{x,j,\ell,\mathcal{J}} : x_{\ell,j} = x^*_k$ for some $k \in [f]$
(d) $\mathsf{E}_{x,j,\ell,2} : x_{\ell,j} = x_{\ell,i}$ for some $i \in \{f+1, \ldots, j-1\}$
(e) $\mathsf{E}_\ell : x'_\ell = x'_j$ for some $j < \ell$
(f) $\mathsf{E}_{\ell,\mathcal{J}} : x'_\ell = x^*_j$ for some $j \in [f]$
(g) $\mathsf{E}_{1,\ell} : \mathcal{R}(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t}, \mathsf{sh}'_\ell) = \mathsf{Recon}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f}, \mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t}, \mathsf{sh}'_\ell)$, i.e, $\mathcal{R}$ outputs correct s_ℓ for sh'_ℓ
(h) $\mathsf{E}_{2,\ell} : \mathcal{R}(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t}, \mathsf{sh}''_\ell) = \mathsf{Recon}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f}, \mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t}, \mathsf{sh}''_\ell)$, i.e, $\mathcal{R}$ outputs correct s_ℓ for sh''_ℓ

We denote $\mathsf{E}_{x,j,\ell} := \mathsf{E}_{x,j,\ell,1} \vee \mathsf{E}_{x,j,\ell,\mathcal{J}} \vee \mathsf{E}_{x,j,\ell,2}$ for all $j \in \{f+1, \ldots, t\}$ and $\ell \in [N]$. Let $\mathcal{T}$ denote the set $\{x^*_j\}_{j\in[f]}$, let $\mathcal{T}_\ell = \{x_{\ell,j}\}_{j\in\{f+1,\ldots,t\}}$ and let $\mathcal{T}^*_\ell = \mathcal{T}^* \cup \mathcal{T}_\ell \cup \{x'_\ell\}$. Thus, in the event $\neg\mathsf{E}_{\delta,\ell} \wedge_{j\in\{f+1,\ldots,t\}} (\neg\mathsf{E}_{x,j,\ell}) \wedge \neg\mathsf{E}_\ell \wedge \neg\mathsf{E}_{\ell,\mathcal{J}} \wedge \neg\mathsf{E}_{\varphi,\ell} \wedge \mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}$, we have that $|\mathcal{T}^*_\ell| = t$ and

$$s_\ell = \sum_{x^*_j \in \mathcal{T}^*} \left(\prod_{x \in \mathcal{T}^*_\ell \setminus \{x^*_j\}} \frac{x}{x - x^*_j} \right) y^*_j + \sum_{x_{\ell,j} \in \mathcal{T}_\ell} \left(\prod_{x \in \mathcal{T}^*_\ell \setminus \{x_{\ell,j}\}} \frac{x}{x - x_{\ell,j}} \right) y_{\ell,j} + \left(\prod_{x \in \mathcal{T}^*_\ell \setminus \{x'_\ell\}} \frac{x}{x - x'_\ell} \right) y'_\ell$$

and

$$s'_\ell = \sum_{x^*_j \in \mathcal{T}^*} \left(\prod_{x \in \mathcal{T}^*_\ell \setminus \{x^*_j\}} \frac{x}{x - x^*_j} \right) y^*_j + \sum_{x_{\ell,j} \in \mathcal{T}_\ell} \left(\prod_{x \in \mathcal{T}^*_\ell \setminus \{x_{\ell,j}\}} \frac{x}{x - x_{\ell,j}} \right) y_{\ell,j} + \left(\prod_{x \in \mathcal{T}^*_\ell \setminus \{x'_\ell\}} \frac{x}{x - x'_\ell} \right) (y'_\ell + \delta_\ell)$$

Thus we yield,

$$s'_\ell - s_\ell = \delta_\ell \cdot \left(\prod_{x_{\ell,j} \in \mathcal{T}_\ell} \frac{x_{\ell,j}}{x_{\ell,j} - x'_\ell} \right) \cdot \left(\prod_{x^*_j \in \mathcal{T}^*} \frac{x^*_j}{x^*_j - x'_\ell} \right)$$

Let us define a polynomial $h^*(x) = \prod_{x^*_j \in \mathcal{T}^*} \frac{x^*_j - X}{x^*_j}$. Then, the algorithm Trace returns a correct evaluation z_ℓ of $h^*(X)$ at $X = x'_\ell$ in ℓ-th iteration. Thus, for each $\ell \in [N]$ we obtain a correct evaluation of $g^*(X)$ at a unique point x'_ℓ in the event $\neg\mathsf{E}_{\delta,\ell} \wedge_{j\in\{f+1,\ldots,t\}} (\neg\mathsf{E}_{x,j,\ell}) \wedge \neg\mathsf{E}_\ell \wedge \neg\mathsf{E}_{\ell,\mathcal{J}} \wedge \neg\mathsf{E}_{\varphi,\ell} \wedge \mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}$. We bound the probability of this event.

Probability Bound

(a) $\Pr[\mathsf{E}_{\delta,\ell}] = \frac{1}{p}$ for all $\ell \in [N]$ as δ_ℓ is uniformly sampled from $\mathbb{F}$ for each ℓ.

(b) $\Pr[\mathsf{E}_{x,j,\ell,1}] = \frac{1}{p}, \quad \Pr[\mathsf{E}_{x,j,\ell,\mathcal{J}}] \leq \frac{f}{p}, \quad \Pr[\mathsf{E}_{x,j,\ell,2}] \leq \frac{t-f}{p}$ for all $\ell \in [N]$ and all $j \in \{f+1, \ldots, t\}$. Hence, by union over j we obtain $\Pr[\mathsf{E}_{x,j,\ell}] \leq \frac{t}{p}$.

(c) $\Pr[\mathsf{E}_\ell \vee \mathsf{E}_{\ell,\mathcal{J}}] \leq \frac{N+f}{p}$ for all $\ell \in [N]$.

(d) Since $\mathcal{R}$ is $(n, t, f, \mathbf{sh}, \epsilon)$-good we have $\Pr[\mathsf{E}_{1,\ell}]$ and $\Pr[\mathsf{E}_{2,\ell}]$ are both at least ϵ. Let W_ℓ be the random variabls $(\{(x_{\ell,j}, y_{\ell,j})\}_{j\in[f+1,t]}, x'_\ell)$. Then by the total law of probability and independence of $\mathsf{E}_{1,\ell}|W_\ell$ and $\mathsf{E}_{2,\ell}|W_\ell$ and Jensen's inequality we obtain that $\Pr[\mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}] \geq \epsilon^2$.

Thus we obtain the following,

$$
\begin{aligned}
&\Pr\left[\neg\mathsf{E}_{\delta,\ell} \bigwedge_{j\in\{f+1,\ldots,t\}} (\neg\mathsf{E}_{x,j,\ell}) \wedge \neg\mathsf{E}_\ell \wedge \neg\mathsf{E}_{\ell,\mathcal{J}} \wedge \mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}\right] \\
&\geq \Pr[\mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}] - \Pr\left[\mathsf{E}_\ell \vee \mathsf{E}_{\delta,\ell} \vee \mathsf{E}_{\ell,\mathcal{J}} \bigvee_{j\in[t]\setminus[f]} \mathsf{E}_{x,j,\ell}\right] \\
&\geq \epsilon^2 - 1/p - (N+f)/p - (n-t-1)/p \\
&\geq \epsilon^2 - (N+t^2+f)/p \\
&\geq \epsilon^2/2
\end{aligned}
$$

The last inequality follows from the fact $\epsilon^2 \geq 2(N+n+t^2)/p$. Let us define an indicator random variable $Z_\ell = \mathbb{I}[\neg\mathsf{E}_{\delta,\ell} \wedge_{j\in\{f+1,\ldots,t\}} (\neg\mathsf{E}_{x,j,\ell}) \wedge \neg\mathsf{E}_\ell \wedge \neg\mathsf{E}_{\ell,\mathcal{J}} \wedge \mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}]$, clearly $Z_\ell = 1$ iff we get a correct evaluation of $h^*(X)$ at a unique x'_ℓ. Since all the shares are sampled independently for each $\ell \in [N]$ we obtain $\Pr[Z_\ell = 1] = \mathbb{E}[Z_\ell] \geq \epsilon^2/2$. Let $Z = \sum_{\ell\in[N]} Z_\ell$. By the Chernoff bound, we have that for every $\eta > 0$,

$$\Pr[Z \leq (1-\eta)\epsilon^2 N/2] \leq e^{-\frac{\epsilon^2 N \eta^2}{4}}$$

Since $N = (2rC)/\epsilon^2$, we can set $\eta = 1 - 1/r$ and thus we obtain,

$$\Pr[Z > C] \geq 1 - e^{-\frac{\epsilon^2 N(1-1/r)^2}{4}}$$

Thus with the probability $1 - e^{-\frac{\epsilon^2 N(1-1/r)^2}{4}}$ the algorithm $\mathcal{D}_{\mathsf{GS}}$ will output the polynomials with at least C evaluations out of $\{x'_\ell, z_\ell\}_{\ell\in[N]}$, including the polynomial $h^*(X)$.

Also, we claim that with high probability there will be no other polynomial in the list H after Step 5. Let us define an event $\mathsf{E}_{h,i}$ for all honest parties $i \in [n]\setminus\mathcal{J}$, denoting that x_i is a root of some polynomial in the list H. We observe

that shares of the honest parties, i.e., $\{x_i, y_i\}_{i \notin \mathcal{J}}$ are statistically independent from both the view of $\mathcal{A}$ and the shares sampled by the $\mathsf{Shamir.Trace}$ algorithm. Hence, the probability that x_i is a root of any polynomial h_j in H is bounded by f/p by Schwarz-Zippel Lemma. Since H contains τ polynomials we obtain that $\Pr[\mathsf{E}_{h,i}] \leq f \cdot \tau/p$. Then, by union bound over all honest parties we obtain

$$\Pr\left[\bigvee_{i \in [n] \setminus \mathcal{J}} \mathsf{E}_i\right] \leq \frac{f \cdot (n-f) \cdot \tau}{p}$$

There is one additional bad event we need to analyze.

- $\mathsf{E}_{\mathsf{hash}}$: For some $a \in [t]$ there exists some (x', d') such that $\mathsf{C}_{c_a} = \mathcal{H}(x' \parallel d')$. Note that, in this case, the tracer will generate a false proof containing $(\mathsf{C}_{c_a}, x', d')$ and there is no way to verify that. If some party finds such a collision of the hash function and tracer uses this pair (x', d') to trace, then it would yield to a polynomial $q^* \neq q$, eventually generating wrong identities that might not be a component of pub. In this case, the tracer would be able to detect the issue after the tracing algorithm yields false identities. To generate such an input (x', d') to the hash that collides with some valid trace key of a certain party is negligible due to the collision-resistance property of $\mathcal{H}$. Thus, we obtain,

$$\Pr[\mathsf{E}_{\mathsf{hash}}] \leq \mathsf{negl}(\lambda)$$

Finally, observe that $\mathcal{A}$ can win the game only if (a) $Z < C$ so that $\mathcal{D}_{\mathsf{GS}}$ fails to return h^* or (b) if either events $\mathsf{E}_{h,i}$ occur, or (c) there is a collision in the underlying hash function. Thus, we obtain the advantage,

$$\mathsf{Adv}^{\mathsf{uni.trace}}_{\mathcal{A},\mathsf{CTS},\epsilon}(\lambda) \leq e^{\frac{-\epsilon^2 N}{2} \cdot (1-1/r)^2} + \frac{f \cdot n \cdot \tau}{p} + \mathsf{negl}(\lambda)$$

□

2. Let $\mathcal{A}$ be an adversary against the non-imputability of CTS. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to attack the one-way function $\mathcal{F}$. The algorithm $\mathcal{B}$ operates as follows:
   - **Input :** $\mathcal{B}$ receives a challenge $u^* = \mathcal{F}(x^*)$ for a uniformly random $x^* \in \mathbb{F}$.
   - **Simulation Setup:**
     (a) $\mathcal{B}$ runs $\mathcal{A}(\lambda)$ to obtain parameters n, t, a target party index i^*, and a secret s.
     (b) $\mathcal{B}$ samples a correlation string $\rho = (a_1, \ldots, a_{t-1}) \xleftarrow{\$} \mathbb{F}^{t-1}$ and defines the polynomial $p(X) = s + \sum_{j=1}^{t-1} a_j X^j$.
     (c) $\mathcal{B}$ samples $d \xleftarrow{\$} \mathbb{F}$ and $(b_1, \ldots, b_t) \xleftarrow{\$} \mathbb{F}^t$ and defines the polynomial $q(X) = d + \sum_{j=1}^{t} b_j X^j$.
     (d) $\mathcal{B}$ generates secret shares, trace credentials, public information to pass to $\mathcal{A}$:

i. For each $i \in [n] \setminus \{i^*\}$, $\mathcal{B}$ picks $x_i \xleftarrow{\$} \mathbb{F}$ and compute $y_i \leftarrow p(x_i)$ and sets $\mathsf{sh}_i = (x_i, y_i)$.
ii. $\mathcal{B}$ generates trace credentials and public value:
- For $i \in [n] \setminus \{i^*\}$, compute $u_i = \mathcal{F}(x_i)$ and $d_i \leftarrow q(u_i)$ and sets the trace credential $\varphi_i \leftarrow (x_i, d_i)$ and public value $\mathsf{C}_i = \mathcal{H}(u_i \parallel d_i)$,
- For party i^*, $\mathcal{B}$ computes $d_{i^*} = q(u^*)$ and set $\varphi_{i^*} = (u^*, d_{i^*})$, $\mathsf{C}_{i^*} = \mathcal{H}(u^* \parallel d_{i^*})$.

(e) $\mathcal{B}$ then sets $\mathsf{pub} \leftarrow ((\mathsf{C}_i)_{i\in[n]}; d)$ and publishes pub. Then, $\mathcal{B}$ provides $\mathcal{A}$ with secret shares $\{\mathsf{sh}_i\}_{i\neq i^*}$ and all the trace credentials $\{\varphi_i\}_{i\in[n]}$. Note that, as mentioned in the non-imputability game in Figure 2, $\mathcal{A}$ gets all the shares except party i^*'s, but gets all trace credentials.

– **Extraction :** The adversary $\mathcal{A}$ outputs a corrupt set $\mathcal{I}^*$ and a proof π, such that, $\mathsf{C}_{i^*} \in \mathcal{I}^*$ and $(C_{i^*}, x', d_{i^*}) \in \pi$. If the proof π is accepted by the verification algorithm, then $\mathcal{B}$ extracts from π a value x' satisfying $\mathcal{F}(x') = u^*$ and outputs x'. Otherwise, $\mathcal{B}$ aborts.

The simulation provided by $\mathcal{B}$ is indistinguishable from the real non-imputability experiment defined in the Figure 2. Thus, any successful accusation of party i^* by $\mathcal{A}$ enables $\mathcal{B}$ to invert $\mathcal{F}$ at u^* with the same probability. Therefore,

$$\mathsf{Adv}^{\mathsf{NI}}_{\mathcal{A},\mathsf{CTS}}(\lambda) = \Pr[\mathcal{F}(\mathcal{B}(\mathcal{F}(x^*))) = \mathcal{F}(x^*)]$$

completing the proof. □

C Missing Proofs from Section 4

C.1 Proof of Theorem 3

Proof (sketch). We will show the following, for two different secrets $\boldsymbol{s} \neq \boldsymbol{s'} \in \mathbb{F}_p^t$ the following two distributions are indistinguishable:

– $\big((\mathbb{G}, g, p), \boldsymbol{r_2}, \ldots, \boldsymbol{r_t}, (g^{\boldsymbol{d_1}}, g^{e_1}), \ldots, (g^{\boldsymbol{d_n}}, g^{e_n}), \mathsf{C}_1, \ldots, \mathsf{C}_n, (\mathsf{sh}_i)_{i\in\Lambda}, (\varphi_i)_{i\in\Lambda}\big)$
– $\Big((\mathbb{G}, g, p), \boldsymbol{r'_2}, \ldots, \boldsymbol{r'_t}, (g^{\boldsymbol{d'_1}}, g^{e'_1}), \ldots, (g^{\boldsymbol{d'_n}}, g^{e'_n}), \mathsf{C}'_1, \ldots, \mathsf{C}'_n, (\mathsf{sh}'_i)_{i\in\Lambda'}, (\varphi'_i)_{i\in\Lambda'}\Big)$

where $\Lambda, \Lambda' \subseteq [n]$ and $|\Lambda| = |\Lambda'| = t - 1$.

Note that, $(\boldsymbol{r_2}, \ldots, \boldsymbol{r_t}) \sim (\boldsymbol{r'_2}, \ldots, \boldsymbol{r'_t})$ because they are $t-1$ additive shares of $\boldsymbol{s}$ and $\boldsymbol{s'}$, respectively. Similarly, we obtain that $(g^{\boldsymbol{d_1}}, g^{e_1}), \ldots, (g^{\boldsymbol{d_n}}, g^{e_n}) \sim (g^{\boldsymbol{d'_1}}, g^{e'_1}), \ldots, (g^{\boldsymbol{d'_n}}, g^{e'_n})$ due to masking of the random element in the power. We get $(\mathsf{C}_1, \ldots, \mathsf{C}_n) \sim (\mathsf{C}'_1, \ldots, \mathsf{C}'_n)$ assuming n-wise independence of the hash function $\mathcal{H}$. We obtain, $(\mathsf{sh}_i)_{i\in\Lambda} \sim (\mathsf{sh}'_i)_{i\in\Lambda'}$ by the theorem 4 of the secrecy of Blakley$^+$ in the work of [BPR24]. Finally, we obtain $(\varphi_i)_{i\in\Lambda} \sim (\varphi'_i)_{i\in\Lambda}$ as they are Shamir's secret share for $d \neq d'$. Thus, we conclude that the above two distributions are indistinguishable. □

C.2 Proof of Theorem 4

Proof of Part 3 of Theorem 4

Proof. Let $\mathcal{A}$ be an adversary playing the game $\mathsf{ExpUniTrace}_{\mathcal{A},\mathsf{CTB},\Gamma,\epsilon}(\lambda)$ and n and t are chosen by $\mathcal{A}$ at the beginning of the experiment. Let G be the event that the reconstruction box $\mathcal{R}$ output by $\mathcal{A}$ is $(n, t, f, \mathbf{sh}, \Gamma, \epsilon)$-good, where $\mathbf{sh} = \{\mathsf{sh}_{i_k}\}_{k=1}^{f}$ are the shares given to it by the challenger. If $\neg G$ occurs, then the experiment outputs 0 with probability 1. Hence, we condition the remainder of the proof on G.

Consider the tracing procedure. In each iteration of Step-2 in Figure 6, the tracer samples fresh random shares $\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}$ satisfying the correlation string $(\boldsymbol{r_2}, \ldots, \boldsymbol{r_t})$, and queries $\mathcal{R}$ on these inputs. Since $\mathcal{R}$ is assumed to be $(n, t, f, \mathbf{sh}, \Gamma, \epsilon)$-good, so in each iteration it holds that

$$\mathcal{R}(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \mathsf{CTB.Recon}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f}, \mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f})$$

with probability at least ϵ. By the definition of the reconstruction algorithm, whenever the above equation holds, we know that

$$\mathcal{R}(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \boldsymbol{r_1} + \boldsymbol{r_2} + \ldots + \boldsymbol{r_t}$$

where for each $j \in [t]$, r_j is the single solution to the system of equations

$$\left\{\langle \boldsymbol{a}_{i_k,j}, \boldsymbol{R}_j\rangle = b_{i_k,j}\right\}_{k\in[f]} \cup \left\{\langle \boldsymbol{a}'_{i,j}, \boldsymbol{R}_j\rangle = b'_{i,j}\right\}_{i\in[t-f]}. \tag{1}$$

where $\boldsymbol{R}_j = (R_{j1}, \ldots, R_{jt})$ is a variable. Recall that the tracing algorithm computes r_1^* as $\boldsymbol{r}_1^* \leftarrow \boldsymbol{s}^* - (\boldsymbol{r}_2 + \ldots + \boldsymbol{r}_t)$ where $\boldsymbol{s}^* = \mathcal{R}(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f})$ and subsequently determines the set $\mathcal{I}'$ of indices $i \in [n]$ for which the following check holds : $\prod_{j=1}^{t} (g^{a^j_{i1}})^{r^j_1} \stackrel{?}{=} g^{b_{i1}}$ where $\boldsymbol{a_{i1}} = (a^j_{i1})_{j=1}^t$ and $\boldsymbol{r_1^*} = (r^j_1)_{j=1}^t$. From Equation 1 we can see that every corrupted party must satisfy the check equation, so every corrupted party must appear in $\mathcal{I}'$. For an honest party to incorrectly appear in $\mathcal{I}'$, the defining hyperplane for that party would need to be incidentally satisfied by $\boldsymbol{r}_1^*$. For a uniformly random hyperplane, this occurs with probability $1/p$. By a union bound, the probability that any honest party is included in $\mathcal{I}'$ is at most $\frac{t}{p}$.

Overall, we have that in each iteration, the probability that tracer outputs the correct subset $\mathcal{I}' = \mathcal{I}$ is at least $\epsilon - \frac{t}{p} \geq \epsilon/2$. Hence, the probability that it computes the correct subset in at least one iteration is at least

$$1 - (1 - \frac{\epsilon}{2})^{\ell} \;\geq\; 1 - e^{\frac{-\epsilon\ell}{2}} \;=\; 1 - e^{-\lambda}. \tag{2}$$

Moreover, the probability that it outputs the wrong subset $\mathcal{I}' \neq \mathcal{I}$ in a given iteration is at most $\frac{n}{p}$, and hence the probability that it outputs the wrong subset in any of the $\ell = \frac{2\lambda}{\epsilon}$ iterations is at most $\frac{2n\lambda}{\epsilon p}$.There is also the possibility that among the collaborating parties p_i, a corrupted party submits a forged share

$\varphi_i^* = (x_i^*, y_i^*)$ during the reconstruction of d, where the hash $\mathcal{H}(\varphi_i^*)$ happens to match one of the valid commitments C_i. However, since $\mathcal{H}$ is collision-resistant, the probability of such an event occurring is only negligible, i.e., $\mathsf{negl}(\lambda)$. Overall, we get that the advantage of $\mathcal{A}$ is bounded by $e^{-\lambda} + \frac{2n\lambda}{\epsilon p} + \mathsf{negl}(\lambda)$. □

Proof of Part 2 of Theorem 4. We first build necessary tools to prove non-imputability of CTB. For our construction of CTB non-imputability property guarantees that having knowledge of the public value pub and trace credentials $\varphi_{\mathcal{C}}$, where $|\mathcal{C}| = t$, the tracer can not accuse an honest party of contributing its share to a pirate reconstruction box $\mathcal{R}$. More concretely, if a challenger gives away the share and trace credential of all parties except for the ones of a party i^*, then no PPT adversary can falsely accuse party i^* of contributing in $\mathcal{R}$.

For our construction of CTB, the public value pub contains information of the hyperplanes $\{(\boldsymbol{a_{i,1}}, b_{i,1})\}_{i\in[n]}$ passing through $\boldsymbol{r_1}$ under a *homomorphic one way function*. The direct knowledge of these hyperplanes shall not be public, otherwise there would be imputability issue. Thus we have used a *homomorphic one way function* over these hyperplanes to hide them. We consider a cyclic group $\mathbb{G}$ of order p and $g \in \mathbb{G}$ be the generator. The map $\mathcal{F} : \mathbb{F}_p^t \times \mathbb{F}_p \to \mathbb{G}^t \times \mathbb{G}$ is defined by

$$\mathcal{F}(\boldsymbol{a}, b) := (g^{\boldsymbol{a}}, g^b)$$

where $g^{\boldsymbol{a}} = (g^{a_1}, \ldots, g^{a_t})$ and $\boldsymbol{a} = (a_1, \ldots, a_t)$. This $\mathcal{F}$ hides the hyperplanes $\{(\boldsymbol{a_{i,1}}, b_{i,1})\}_{i\in[n]}$ and also allows to check whether $\langle \boldsymbol{a_{i,1}}, \boldsymbol{r_1} \rangle = b_{i,1}$. Now if an adversary wants to generate a false proof $(\boldsymbol{r_1^*, d})$, accusing an honest party i^* of being corrupt, it needs to compute $\boldsymbol{r_1^*}$ over the exponents of g. More concretely, the adversary needs to find a $\boldsymbol{r_1^*} = (r^{*1}, \ldots, r^{*t}) \in \mathbb{F}_p^t$ such that $\prod_{j=1}^{t} (g^{(d_{i^*}^j - d)})^{r^{*j}} = g^{(e_{i^*} - d)}$ holds. Finding such $\boldsymbol{r_1^*}$ requires computing r^{*j} from $(g^{(d_{i^*}^j - d)})^{r^{*j}}$, which contradicts the DLR assumption. Thus our CTB is non-imputable by construction. So in this section, we will formally prove that our CTB construction is non-imputable if the Discrete-Log Relation (DLR) problem is hard.

Definition 10. *Let $\mathcal{G}$ be a group generation algorithm and let $\ell = \ell(\lambda)$ be a function of the security parameter $\lambda \in \mathbb{N}$. We say that the* Discrete-Log Relation (DLR) *problem is hard relative to $\mathcal{G}$ if for every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that*

$$\mathcal{A}^{\mathsf{DLR}}_{\mathcal{A},\mathcal{G}}(\lambda) := \Pr\left[(a, r) = 0 \ \wedge \ r \neq 0 : \begin{array}{l}(\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}(1^\lambda) \\ a \leftarrow \mathbb{Z}_p^\ell \\ r \leftarrow \mathcal{A}((\mathbb{G}, g, p), g^a)\end{array}\right] < \nu(\lambda)$$

for all $\lambda \in \mathbb{N}$.A simple reduction shows that the DLR problem in $\mathbb{G}$ is tightly equivalent to the discrete-log problem in $\mathbb{G}$.

Definition 11. *For every adversary $\mathcal{A}$ for the non-imputabiltiy of* CTB*, there exists an algorithm $\mathcal{B}$ for the* DLR *problem, such that for every $\lambda \in N$, it holds that*

$$\mathcal{A}^{\mathsf{ni}}_{\mathcal{A},\mathsf{CTB}}(\lambda) = \mathcal{A}^{\mathsf{DLR}}_{\mathcal{A},\mathcal{G}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an adversary participating in the non-imputability experiment for CTB. We construct an algorithm $\mathcal{B}$ that breaks the discrete log relation problem with respect to $\mathbb{G}$. B gets as input a vector $g^{\boldsymbol{x}} = (g^{x_1}, \ldots, g^{x_t})$ of group elements sampled uniformly at random from G. It invokes $\mathcal{A}$ on 1^λ to obtain $n \in \mathbb{N}$, $t \leq n$, a party $i^* \in [n]$, and a secret $\boldsymbol{s} \in \mathbb{F}_p^t$. It then generates the public values, trace credentials, and shares for $\mathcal{A}$ as follows:

1. It samples an additive secret sharing $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t}$ for $\boldsymbol{s}$.
2. For every $i \in [n] \setminus \{i^*\}$, $\mathcal{B}$ does the following
   - Samples a secret share $\mathsf{sh}_i = \{(\boldsymbol{a}_{i,j}, b_{i,j})\}_{j\in[t]}$ for $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t}$ honestly. That is, for every $j \in [t]$, $\boldsymbol{a}_{i,j}$ is uniformly-random and $b_{i,j} = \langle \boldsymbol{a}_{i,j}, \boldsymbol{r}_j \rangle$.
   - Sample $d \xleftarrow{\$} \mathbb{F}_p$ and $q_1, \ldots, q_{t-1} \xleftarrow{\$} \mathbb{F}_p^{t-1}$.
   - Set $q(X) = d + q_1 X + \ldots + q_{t-1} X^{t-1}$
   - Sample $x_i \xleftarrow{\$} \mathbb{F}_p$ without replacement for all $i \in [n] \setminus i^*$.
   - Set $y_i \leftarrow q(x_i)$, $\varphi_i \leftarrow (x_i, y_i)$ and $\mathsf{C}_i \leftarrow \mathcal{H}(\varphi_i)$ for all $i \in [n] \setminus i^*$.
   - For $i \in [n] \setminus i^*$
     - Set $\boldsymbol{d_i} \leftarrow \boldsymbol{a_{i,1}} + \boldsymbol{d}$, where $\boldsymbol{d} = (d, d, \ldots, d) \in \mathbb{F}_p^t$
     - Set $e_i \leftarrow b_{i,1} + d$
   - Set $g^{\boldsymbol{d_i}} = (g^{d_i^1}, \ldots, g^{d_i^t})$ and $\boldsymbol{d_i} = (d_i^1, \ldots, d_i^t)$ for all $i \in [n] \setminus i^*$.
3. For $i = i^*$, $\mathcal{B}$ sets $g^{\boldsymbol{d}_{i^*}} \leftarrow g^{\boldsymbol{x}}.g^{\boldsymbol{d}}$ and $g^{e_{i^*}} \leftarrow g^{\langle \boldsymbol{x}, \boldsymbol{r}_1 \rangle}$.
4. Set $\mathsf{pub} \leftarrow \big((\mathbb{G}, g, p), \boldsymbol{r}_2, \ldots, \boldsymbol{r}_t, (g^{\boldsymbol{d_1}}, g^{e_1}), \ldots, (g^{\boldsymbol{d_n}}, g^{e_n}), \mathsf{C}_1, \ldots, \mathsf{C}_n\big)$.
5. $\mathcal{B}$ gives $\mathcal{A}$ the shares $\{\mathsf{sh}_i\}_{i\in[n]\setminus\{i^*\}}$, as well as the public value pub.

$\mathcal{A}$ then outputs a subset $\mathcal{I}^*$ and a proof π. If $i^* \in \mathcal{I}^*$ and π is accepting, then it must include a vector $\boldsymbol{r}^*$ and d such that

$$\prod_{j=1}^{t} (g^{(d_{i^*}^j - d)})^{r^{*j}} = g^{(e_{i^*} - d)}$$

where $g^{\boldsymbol{d}_{i^*}} = (g^{d_{i^*}^1}, \ldots, g^{d_{i^*}^t})$ and $\boldsymbol{r}^* = (r^{*1}, \ldots, r^{*t})$ Moreover, for π to be accepting it must be that $\boldsymbol{r}^* \neq \boldsymbol{r}_1$, since $\langle \boldsymbol{a}_{i',1}, \boldsymbol{r}_1 \rangle = b_{i',1}$ for some $i' \in [n]$. By construction, we know that

$$\prod_{j=1}^{t} (g^{(d_{i^*}^j - d)})^{r^j} = g^{(e_{i^*} - d)}.$$

Where $\boldsymbol{r} = (r^1, \ldots, r^t)$. Dividing the two equations, we obtain

$$\prod_{j=1}^{t} (g^{(d_{i^*}^j - d)})^{(r^* - r^j)} = 1_{\mathbb{G}}.$$

Recalling that $g^{\boldsymbol{d}_{i^*}} = g^{\boldsymbol{x}}.g^{\boldsymbol{d}}$, this implies that $\langle \boldsymbol{x}, \boldsymbol{r}^* - \boldsymbol{r}_1 \rangle = 0$. Hence, if not aborted before, $\mathcal{B}$ outputs the vector $\boldsymbol{r}^* - \boldsymbol{r}_1$.

Observe that $\mathcal{B}$ perfectly simulates the non-imputability experiment to $\mathcal{A}$. Moreover, whenever $\mathcal{A}$ wins in the experiment, $\mathcal{B}$ succeeds in solving the discrete log relation problem. Hence,

$$\mathcal{A}^{\mathsf{ni}}_{\mathcal{A},\mathsf{CTB}}(\lambda) = \mathcal{A}^{\mathsf{DLR}}_{\mathcal{A},\mathcal{G}}(\lambda).$$

completing the proof. □