A Traceable Threshold Asmuth–Bloom Secret Sharing Scheme

Maria Leslie and Ratna Dutta

Department of Mathematics, Indian Institute of Technology Kharagpur, India
leslieedberg98@kgpian.iitkgp.ac.in, ratna@maths.iitkgp.ac.in

Abstract. In a t-out-of-n threshold secret sharing scheme, accountability is crucial when a subset of $f < t$ servers collude to leak secret shares. *Traceable Threshold Secret Sharing* (TTSS) ensures that leaked shares can be traced back to the compromised servers while preventing false accusations through non-imputability. In Crypto'24, Boneh et al. proposed new definitions and more practical constructions for TTSS based on Shamir's and Blakley's secret sharing schemes, removing the practical limitation of existing TTSS.

Our work presents a traceable secret sharing scheme built upon an additive variant of the Asmuth-Bloom scheme, relying only on black-box access to the reconstruction box $\mathcal{R}$. In our model, a subset of $f < t$ colluding servers can construct a reconstruction box $\mathcal{R}$ that recovers the secret with the assistance of an additional $t - f$ random shares. We note that integrating tracing in the standard (t, n)-Asmuth-Bloom Secret Sharing (ABSS) scheme exhibits a tracing leakage issue. We fix this limitation by introducing additive variants of ABSS, ABSS-I and ABSS-II that retain the security of the original scheme ABSS while splitting the secret s into t additive components and generating all shares from the additive components of s. Based on ABSS-I, we construct a TTSS scheme, TTSS-I, that introduces traceability into the framework and is proven to be universally traceable in the random oracle model, assuming $\mathcal{R}$ is a universally good reconstruction box. We integrate a tracing mechanism in ABSS-II and propose a second scheme, TTSS-II, which extends TTSS-I by additionally concealing partial information about the additive component of the secret s to introduce non-imputability to prevent the tracer from falsely accusing any honest party by fabricating evidence of its corruption. The security of TTSS-II is also in the random oracle model and relies on the hardness of the discrete logarithm problem.

Keywords: Asmuth-Bloom · Traceability · Non-Imputability

1 Introduction

Threshold secret sharing (TSS) is a fundamental cryptographic technique, where the secret is distributed among n servers and only groups consisting of at least $t \leq n$ servers are capable of reconstructing the secret while smaller groups

learn nothing. TSS was introduced independently by Shamir [Sha79] in 1979 and Blakley [Bla79] in the same year. These schemes are the foundation for modern secret sharing, influencing cryptographic protocols and secure multi-party computation. Shamir's approach utilizes polynomial interpolation within a finite field, while Blakley's method is founded on the principles of geometric hyperplanes. These schemes are the foundation for modern secret sharing, influencing cryptographic protocols and secure multi-party computation. Later in 1983, Asmuth-Bloom [AB83] and Mignotte [Mig82] extended TSS using the *Chinese Remainder Theorem* (CRT) and introduced additional security properties and flexibility in key management. A core application of threshold secret sharing is the protection of sensitive data stored on untrusted servers. Consider the scenario in which a party shares confidential information (e.g., a cryptographic key) among n servers using a TSS. If $f < t$ servers collude to leak or sell their shares, potentially in a concealed manner, tracing them back to the original servers is challenging.

Traceable TSS (TTSS). Goyal, Song and Srinivasan [GSS21] proposed the notion of TTSS that enables the identification of parties responsible for leaking secret shares. The TTSS gives the following two guarantees:

- If a set of secret shares is leaked, it is possible to identify the specific servers responsible for the leakage. Furthermore, the scheme allows for the generation of a verifiable proof that can be used to hold them accountable. This property is called *traceability*.
- No one can fabricate a false yet convincing proof that implicates an innocent server who was uninvolved in the leakage. This property is called *non-imputability*.

The TTSS scheme put forth by Goyal et al. satisfies both the properties. However, their construction faces significant efficiency challenges. In particular, the size of each share increases quadratically with respect to the secret, whereas classical schemes like Shamir maintain a share size equivalent to that of the secret. Additionally, the tracing procedure is computationally intensive, relying on the Goldreich-Levin decoding algorithm.

These inefficiencies motivated Boneh, Partap and Rotem [BPR24] to propose a more refined definition of TTSS. They designed practical constructions based on both Shamir's and Blakley's classic TSS schemes. Their TTSS schemes achieve both traceability and non-imputability, with a significant improvement over prior work. A natural open question that arises is whether other well-known secret sharing schemes can support similar tracing mechanisms. Hoffmann [Hof24] introduced a tracing method for a modified version of Mignotte's TSS scheme. However, that scheme offers only a weaker security. In any case, the issue of identifying leaks in Asmuth-Bloom secret sharing continues to be an important unsolved challenge. This scheme is especially appealing because it can be efficiently extended to support weighted threshold access structures, where a weight is assigned to each party and reconstruction of the secret is possible only if the total weight of the participants meets the required threshold.[[Gar+23], [Zou+11]].

1.1 Our contribution

In this paper, we present a TTSS construction based on the Asmuth-Bloom scheme [AB83]. Our contributions can be summarized as follows:

1. We propose a novel additive variant of the classical ABSS scheme.
2. We extend this additive variant to incorporate the traceability property.
3. We further enhance the scheme to achieve the non-imputability property.

It is important to observe that when the count of colluding servers $|\mathcal{I}|$ reaches or exceeds t, the process of tracing becomes unfeasible. This is due to the capability of these servers to independently reconstruct the secret and subsequently leak it, resulting in the absence of any identifiable evidence. Consequently, attention is directed towards the scenario in which $f < t$. In our model, a subset $\mathcal{I}$ of $f < t$ compromising servers generates a reconstruction box $\mathcal{R}$ that depends upon the shares held by the servers in $\mathcal{I}$. This component takes in extra $t - f$ shares that are not part of $\mathcal{I}$ and produces the secret by combining the input shares with those held by the servers in $\mathcal{I}$. In the outlined approach, the dealer is responsible for both distributing the shares and generating a tracing key (tk) along with a verification key (vk). Subsequently, the definition necessitates the existence of a tracing algorithm that, when provided with black-box access to $\mathcal{R}$, can identify the f colluding servers in $\mathcal{I}$ using the tk and produce a proof implicating them.. This proof must be publicly verifiable using the vk. Importantly, the tracing algorithm operates in a black-box manner, meaning that it does not require access to the internal code of $\mathcal{R}$, since malicious servers may obfuscate or cloak their implementation before releasing it. Regardless, the tracing algorithm is designed to function with only black-box access to $\mathcal{R}$.

Tracing information leakage in ABSS. In Section 3 we present an algorithm for tracing in t-out-of-n ABSS scheme. We denote the secret by s, where $s \in \mathbb{Z}$. To incorporate the threshold property, we select pairwise co-prime integers $m_0 < m_1 < \cdots < m_n$, with m_0 being prime, such that the product $M = \prod_{i=1}^{t} m_i$ is strictly greater than $M' = m_0 \cdot \prod_{i=1}^{t-1} m_{n-i+1}$. To uniquely reconstruct the given secret $s < m_0$, the dealer randomly encodes the secret as $y = s + A \cdot m_0$, ensuring $y < M$ by adding a random $A \in \mathbb{N}$. Each server $i \in [n]$ receives as its share the modular congruence $Y_i = y \bmod m_i$. Since the moduli m_i are pairwise relatively prime and $y < M$, it is sufficient to have any set of t shares to reconstruct y uniquely and consequently, the original secret s. Meanwhile, any collection of fewer than t shares (specifically, $t - 1$) reveals negligible information about s, as the value y may exceed M', thereby providing statistical secrecy.

Consider a perfect reconstruction box $\mathcal{R}$ that contains f modular congruences $Y_{i_1}, Y_{i_2}, \ldots, Y_{i_f}$ hardcoded within it. Let $\mathcal{R}$ output the secret $s' = y' \bmod m_0$, where y' is the solution to a system of modular congruences comprising the hardcoded congruences $Y_{i_1}, Y_{i_2}, \ldots, Y_{i_f}$ and $t - f$ random congruences $Y'_1, Y'_2, \ldots, Y'_{t-f}$ provided as input. Assume that we can compute the encoded value y' from the reconstructed secret s'. In such a case, the process of tracing becomes much simpler. It is sufficient to execute $\mathcal{R}$ on a set of $t - f$ randomly

generated modular congruences $Y'_1, Y'_2, \ldots, Y'_{t-f}$ in order to track $\mathcal{R}$ back to the corrupted servers $i_1, i_2, \ldots, i_f$. It is easy to observe that the encoded value y' obtained from $\mathcal{R}$'s output is a random point in $\mathbb{Z}$, conditioned on satisfying the f hardcoded modular congruences. If the moduli m_i are large, then with overwhelming probability, the resulting y' will not satisfy any of the honest servers' congruences. This discrepancy reveals the identities of compromised servers. Thus, a single inquiry to $\mathcal{R}$ allows us to identify the malicious parties.

However, a challenge arises: to compute y' from s', we require knowledge of both s' and a value A' such that $y' = s' + A' \cdot m_0$. Since we only have the reconstructed secret s', this information is insufficient to uniquely determine y' and thus the above tracing strategy fails. To overcome this, in Section 3.1, we propose a variant of Blakley's scheme wherein the secret is divided as elements in $\mathbb{Z}$ and ensuring that this modification preserves the security of the underlying scheme. Then, in Section 3.2, we propose a tracing technique based on the idea that was presented before for our variation scheme. This tracing algorithm is optimal as it only needs one query to the reconstruction box $\mathcal{R}$.

Our variant of the ABSS scheme offers the following advantages:

1. It enables the tracing of all f leaked shares.
2. The scheme remains effective even with imperfect reconstruction boxes.
3. It satisfies both traceability and non-imputability properties.

However, when the tracing key is revealed, our scheme achieves a weaker form of secret hiding. It functions as a private tracing scheme.

1.2 Related works

Recent progress in cryptographic research has introduced several primitives designed to identify and trace unauthorized access or leaks within cryptographic systems.

Traitor Tracing. A similar notion to TTSS is traitor tracing, first formalized by Chor et al. [CFN94], where the central party samples a public parameter which includes an encryption key, n secret decryption keys and sends these n secret decryption keys to a set of n parties through a secure communication channel. Suppose any set $\mathcal{J}$ of colluding parties produces a pirate decryption box $\mathcal{D}$ that depends on the secret decryption keys of the parties in $\mathcal{J}$ and even an unauthorized party can decrypt the well-formed ciphertexts using $\mathcal{D}$. Traitor tracing ensures that the leaked secret keys can be traced back to at least one of the parties responsible for the leak. Over the years, numerous efficient traitor tracing schemes have been developed in both the classical and quantum settings [[BS95], [KD98], [NP98], [BF99], [FT99], [KY01], [KY02], [DF03], [CPP05], [BSW06], [BZ14], [NWZ16], [GKW18], [Zha20], [Wee20], [GLW23]].

TTSS. Goyal et al. [GSS21] introduced the primitive of TTSS, where the secret is distributed among n servers and at least $t \leq n$ secret shares can reconstruct the secret. Suppose a set of $f < t$ colluding servers produces a reconstruction box $\mathcal{R}$ and their scheme allows for identifying at least one of the leaked secret shares

by tracing them back to the servers who leaked them. However, their approach has practical limitations due to its computationally intensive tracing algorithm which requires quadratic-sized shares and relies on Goldreich-Levin decoding. These constraints make it challenging to implement the scheme efficiently in real-world applications. Building on this foundational work, Boneh, Partap and Rotem [BPR24] presented a practical construction for a TTSS scheme based on Shamir's secret sharing scheme and an extended variant of Blakley's secret sharing scheme. Their schemes significantly reduce overhead by constructing shares that are only twice the size of the secret. In 2024, Hoffmann [Hof24] presented the first CRT based TTSS scheme. This innovative scheme extends Mignotte's secret sharing into a traceable variant, supporting weighted threshold access structures and proven secure in the random oracle model. Compared to the original Mignotte scheme, the share size grows only by a factor of two.

2 Preliminary

Notations. This section presents the basic notations and cryptographic primitives that are used in this work. Let $\mathbb{Z}$ represent the set of all integers, $\mathbb{N}$ be the set of all positive integers, $\mathbb{R}$ be the set of real numbers and $\mathbb{Z}_p$ be the finite field of integer modulo a prime number p. For a positive integer n, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$ and $\lfloor n \rfloor$ the floor function, the greatest integer less than or equal to x. For the real numbers a and b, we denote the set $[a, b] = \{x \in \mathbb{R} | a \leq x \leq b\}$. We represent by $x \leftarrow S$ used for denoting x is the output of algorithm S and by $y \xleftarrow{\$} S$ the process of sampling a value y randomly from the set S. A function $f(\lambda)$ is said to be negligible in the security parameter λ if it decreases faster than the inverse of any polynomial in λ. All other notations and existing algorithms or cryptographic primitives that we use are provided in the corresponding sections.

Definition 1 (Chinese Remainder Theorem [Wan98]). *Let $p_1, p_2, \ldots, p_n$ be pairwise co-prime integers and $x_1, x_2, \ldots, x_n \in \mathbb{Z}$. Then the system*

$$\begin{cases} X = x_1 \bmod p_1 \\ X = x_2 \bmod p_2 \\ \quad \vdots \\ X = x_n \bmod p_n \end{cases}$$

has a unique solution modulo $N = p_1 p_2 \ldots p_n$ given by $X = \sum_{i=1}^{n} x_i N_i M_i$ where for $i \in [n]$, $N_i = \frac{N}{p_i}$ and $M_i = N_i^{-1} \bmod p_i$.

Definition 2 (Group Generation Algorithm [OBr90]). *A group generation algorithm $\mathcal{G}$ takes (1^λ) as input where λ is a security parameter and outputs a triple $(\mathbb{G}, g, p)$ where $\mathbb{G}$ is the description of a group of order p generated by g.*

Definition 3 (Discrete Logarithm Problem [McC90]). *Let $\mathcal{G}$ be a group generation algorithm and $\lambda \in \mathbb{N}$ be a security parameter. The Discrete Logarithm*

(DL) problem is hard relative to $\mathcal{G}$ if for every probabilistic polynomial time (PPT) adversary $\mathcal{B}$, it holds that

$$Adv^{DL}_{\mathcal{B},\mathcal{G}}(\lambda) = \Pr[Exp^{DL}_{\mathcal{B},\mathcal{G}}(\lambda) = 1]$$

is negligible in λ where the experiment $Exp^{DL}_{\mathcal{B},\mathcal{G}}(\lambda)$ between a challenger and the adversary $\mathcal{A}$ is presented in Fig.1

$\mathsf{Exp}^{\mathsf{DL}}_{\mathcal{B},\mathcal{G}}(\lambda)$

$(\mathbb{G}, g, p) \leftarrow \mathcal{G}(1^\lambda)$;
$x \leftarrow \mathbb{Z}_p$;
$h = g^x \bmod p$;
$x' \leftarrow \mathcal{B}(\mathbb{G}, g, h)$;
if $(g^{x'} = h \bmod p)$ **then**
  return 1;
else
  return 0;
end if

Fig. 1: The hardness assumption for the discrete logarithm problem

Definition 4 (Threshold Secret Sharing). *A threshold secret sharing scheme $TSS = (Share, Rec)$ consists of two polynomial time algorithms satisfying the following requirements with 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.*

- $Share(1^\lambda, s, n, t) \to (pp, \{sh_i\}_{i=1}^n)$. *This PPT algorithm is run by a dealer who takes as input a security parameter λ, a secret $s \in \mathcal{S}_\lambda$, the number of parties n and a threshold $t \leq n$. It outputs a public parameter pp which is made publicly available and a share sh_i of the i-th server which is sent through a secure communication channel between the dealer and the i-th server for $i = 1, 2, \ldots, n$.*
- $Rec(pp, \{sh_{i_k}\}_{k=1}^t) \to s$. *This deterministic algorithm is run by a combiner who possesses the public parameter pp and at least t distinct valid shares $sh_{i_1}, sh_{i_2}, \ldots, sh_{i_t}$ and outputs a secret $s \in \mathcal{S}_\lambda$.*

Correctness. Any subset of at least t shares can correctly reconstruct the secret in $\mathsf{TSS} = (\mathsf{Share}, \mathsf{Rec})$ presented above.

Security. A coalition of fewer than t shares cannot reconstruct the secret in $\mathsf{TSS} = (\mathsf{Share}, \mathsf{Rec})$ presented above.

Definition 5 (Traceable Threshold Secret Sharing [BPR24]). *A Traceable Threshold Secret Sharing scheme $TTSS = (Share, Rec, Trace, Verify)$ consists of four polynomial time algorithms satisfying the following requirements with 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.*

- $\mathsf{Share}(1^\lambda, s, n, t, \rho) \to (\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$. *This PPT sharing algorithm is run by a dealer who takes as input a security parameter λ, a secret $s \in \mathcal{S}_\lambda$, the number of parties n, a threshold $t \leq n$ and a correlation string $\rho \in \{0,1\}^\kappa$ with $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$. It generates a public parameter pp which is made publicly available, computes a share sh_i for the i-th server which is sent to the i-th server through a secure communication channel between the dealer and the i-th server for $i = 1, 2, \ldots, n$, sets a tracing key tk which is sent to the tracer through a secure communication channel between the dealer and the tracer and publishes a verification key vk.*
- $\mathsf{Rec}(\mathsf{pp}, \mathsf{sh}_{i_1}, \mathsf{sh}_{i_2}, \ldots, \mathsf{sh}_{i_t}) \to s$. *This deterministic algorithm is run by a combiner who possesses the public parameter pp and at least t distinct valid shares $\{\mathsf{sh}_{i_k}\}_{k=1}^t$ and outputs a secret $s \in \mathcal{S}_\lambda$.*
- $\mathsf{Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon}) \to (\mathcal{I}', \pi)$. *This randomized tracing algorithm run by a tracer holding the tracing key tk takes as input the public parameter pp, an error parameter ϵ and has oracle (black-box) access to a reconstruction box $\mathcal{R}$ containing $f < t$ hardcoded shares. The algorithm outputs a subset $\mathcal{I}' \subseteq [n]$ of identities of share leaking parties and an associated proof π that the parties in $\mathcal{I}'$ are guilty of revealing share.*
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathcal{I}', \pi) \to 0/1$. *This deterministic algorithm is run by any public verifier. It takes as input the public parameter pp, the verification key vk, an alleged traitor subset $\mathcal{I}' \subseteq [n]$ and an associated proof π and outputs either 1, implying acceptance of the proof π that the parties in $\mathcal{I}'$ consisting of identities of share leaking parties or 0, implying rejection.*

Correctness. The correctness requirement for a TTSS scheme is the standard correctness property of the (t,n)-TSS scheme which states that any t shares should suffice to correctly reconstruct the secret in TTSS = (Share, Rec, Trace, Verify) presented above.

Definition 6 (ϵ-correct [BPR24]). *Let $\epsilon = \epsilon(\lambda) \in [0,1]$ be a function of the security parameter λ. We say that a traceable threshold secret sharing scheme $\mathit{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ 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, 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 $\Lambda = \{i_1, i_2, \ldots, i_t\} \subseteq [n]$ of size t, it holds that*

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

where $(\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk}) \leftarrow \mathit{TTSS}.\mathsf{Share}(1^\lambda, s, n, t, \rho)$ and $s' \leftarrow \mathit{TTSS}.\mathsf{Rec}(\mathsf{pp}, \{\mathsf{sh}_{i_k}\}_{k=1}^t)$.

Security. A traceable secret-sharing scheme TTSS = (Share, Rec, Trace, Verify) presented above should satisfy three security properties: secrecy, traceability and non-imputability.

Definition 7 (ϵ-secrecy [BPR24]). *Let $\epsilon = \epsilon(\lambda) \in [0,1]$ be a function of the security parameter λ. We say that a traceable threshold secret sharing scheme*

TTSS = (*Share*, *Rec*, *Trace*, *Verify*) *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'*, 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*

$$\Pr[\mathcal{D}(\{sh_i\}_{i\in\Lambda'} = 1) - \mathcal{D}(\{sh'_i\}_{i\in\Lambda'} = 1)] \leq \epsilon(\lambda)$$

where $\rho \in \{0,1\}^\kappa$ *with* $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$, $(pp, \{sh_i\}_{i=1}^n, tk, vk) \leftarrow$ *TTSS.Share*$(1^\lambda, s, n, t, \rho)$ *and* $(pp', \{sh'_i\}_{i=1}^n, tk', vk') \leftarrow$ *TTSS.Share*$(1^\lambda, s', n, t, \rho)$.

Definition 8 (Good Reconstruction Box [BPR24]). *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 traceable threshold secret sharing scheme* *TTSS* = (*Share*, *Rec*, *Trace*, *Verify*) *with secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* (n,t,f,s,ρ,ϵ)*-good if*

$$\Pr[\mathcal{R}(\{sh'_{i_k}\}_{k=1}^{t-f}) = s] \geq \epsilon$$

where $(pp, \{sh'_i\}_{i=1}^n, tk', vk') \leftarrow$ *TTSS.Share*$(1^\lambda, s, n, t, \rho)$

Definition 9 (Universally-Good Reconstruction Box [BPR24]). *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* $sh = \{sh_{i_k}\}_{k=1}^f$ *and a subset* $\mathcal{C}' \subseteq \{0,1\}^\kappa$*, we say that a reconstruction box* $\mathcal{R}$ *in a traceable threshold secret sharing scheme* *TTSS* = (*Share*, *Rec*, *Trace*, *Verify*) *with secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* $(n,t,f,sh,\mathcal{C}',\epsilon)$*-good if*

$$\Pr[s' = s'' \mid s' \leftarrow \mathcal{R}(\{sh'_{i_k}\}_{k=1}^{t-f}), s'' \leftarrow \mathsf{TTSS.Rec}(pp,sh, \{sh'_{i_k}\}_{k=1}^{t-f})] \geq \epsilon$$

where $(pp, \{sh'_i\}_{i=1}^n, tk', vk') \leftarrow$ *TTSS.Share*$(1^\lambda, s, n, t, \rho)$ *and probability is on the random coins of* $\mathcal{R}$

$\mathsf{Exp}^{\mathsf{Trace}}_{\mathcal{A},\mathsf{TTSS}}(\lambda, \epsilon, \delta)$

1: $(n, t, \mathcal{I}) \leftarrow \mathcal{A}(1^\lambda)$ where $\mathcal{I} \subset [n]$ is the set of corrupted parties and $|\mathcal{I}| = f < t$;
2: $s^* \xleftarrow{\$} \mathcal{S}_\lambda$;
3: $\rho \xleftarrow{\$} \{0,1\}^\kappa$;
4: $(\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk}) \leftarrow \mathsf{TTSS.Share}(1^\lambda, s^*, n, t, \rho)$;
5: $\mathcal{R} \leftarrow \mathcal{A}(\{\mathsf{sh}_j\}_{j\in\mathcal{I}})$;
6: **if** ($\mathcal{R}$ is not $(n,t,f,s^*,\rho,\epsilon)$-good) **then**
7: **return** 0;
8: **else**
9: $(\mathcal{I}', \pi) \leftarrow \mathsf{TTSS.Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\delta})$;
10: **if** ($\mathcal{I} = \mathcal{I}' \wedge \mathsf{TTSS.Verify}(\mathsf{pp},\mathsf{vk}, \mathcal{I}', \pi) \to 1$) **then**
11: **return** 0;
12: **else**
13: **return** 1;
14: **end if**
15: **end if**

Fig. 2: Traceability experiment for TTSS

Definition 10 (Traceability [BPR24]). *A traceable threshold secret sharing scheme* $\mathit{TTSS} = (\mathit{Share}, \mathit{Rec}, \mathit{Trace}, \mathit{Verify})$ *with secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ *satisfies traceability if for every PPT adversary* $\mathcal{A}$*, it holds that*

$$Pr[\mathit{Exp}^{\mathit{Trace}}_{\mathcal{A},\mathit{TTSS},}(\lambda,\epsilon,\delta)=1]$$

is negligible in λ *where* ϵ*,* δ *are functions of the security parameter* λ *and the experiment* $\mathit{Exp}^{\mathit{Trace}}_{\mathcal{A},\mathit{TTSS},}(\lambda,\epsilon,\delta)$ *between a challenger and the adversary* $\mathcal{A}$ *is presented in Fig.2*

$\mathsf{Exp}^{\mathsf{UniTrace}}_{\mathcal{A},\mathsf{TTSS}}(\lambda,\epsilon,\Gamma)$

1: $(n,t,\mathcal{I}) \leftarrow \mathcal{A}(1^\lambda)$ where $\mathcal{I}\in[n]$ is the set of corrupted parties and $|\mathcal{I}|=f<t$;
2: $s^* \xleftarrow{\$} \mathcal{S}_\lambda$;
3: $\rho \xleftarrow{\$} \{0,1\}^\kappa$;
4: $(\mathsf{pp},\{\mathsf{sh}_i\}_{i=1}^n,\mathsf{tk},\mathsf{vk}) \leftarrow \mathsf{TTSS.Share}(1^\lambda,s^*,n,t,\rho)$;
5: $\mathcal{R} \xleftarrow{\$} \mathcal{A}(\{\mathsf{sh}_j\}_{j\in\mathcal{I}})$;
6: $\mathsf{sh}=\{\mathsf{sh}_j\}_{j\in\mathcal{I}}$;
7: **if** ($\mathcal{R}$ is not $(n,t,f,\mathsf{sh},\Gamma(\rho),\epsilon)$-good) **then**
8: **return** 0;
9: **else**
10: $(\mathcal{I}',\pi) \leftarrow \mathsf{TTSS.Trace}^{\mathcal{R}}(\mathsf{pp},\mathsf{tk},f,1^{1/\epsilon})$;
11: **if** ($\mathcal{I}=\mathcal{I}' \wedge$ TTSS.Verify(pp,vk, $\mathcal{I}'$, π) $\rightarrow 1$) **then**
12: **return** 0;
13: **else**
14: **return** 1;
15: **end if**
16: **end if**

Fig. 3: Universal Traceability experiment for TTSS

Definition 11 (Universal Traceability [BPR24]). *A traceable threshold secret sharing scheme* $\mathit{TTSS} = (\mathit{Share}, \mathit{Rec}, \mathit{Trace}, \mathit{Verify})$ *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*

$$\mathit{Adv}^{\mathit{UniTrace}}_{\mathcal{A},\mathit{TTSS1}}(\lambda,\epsilon,\Gamma(\rho)) = Pr[\mathit{Exp}^{\mathit{UniTrace}}_{\mathcal{A},\mathit{TTSS}}(\lambda,\epsilon,\Gamma)=1]$$

is negligible in λ *where* $\Gamma=\{\mathcal{C}_1,\mathcal{C}_2,\dots\}$ *is a partition of the space* $\{0,1\}^\kappa$ *of correlation strings with* $\Gamma(\rho)\subset\{0,1\}^\kappa$ *is a unique subset* $\mathcal{C}_i$ *that contains* $\rho\in\{0,1\}^\kappa$ *with* $\kappa=\kappa(n,t,\lambda)\in\mathbb{N}$ *and the experiment* $\mathit{Exp}^{\mathit{UniTrace}}_{\mathcal{A},\mathit{TTSS}}(\lambda,\epsilon,\Gamma)$ *between a challenger and the adversary* $\mathcal{A}$ *is presented in Fig.3*

Theorem 1. *[BPR24] A traceable threshold secret sharing scheme* $\mathit{TTSS} = (\mathit{Share}, \mathit{Rec}, \mathit{Trace}, \mathit{Verify})$ *which satisfies universal traceability also satisfies standard traceability as per Definition 10.*

Definition 12 (Non-Imputability [BPR24]). *A traceable threshold secret sharing scheme* $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ *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*

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

is negligible in λ *where the experiment* $\mathsf{Exp}^{\mathsf{NI}}_{\mathcal{A},\mathsf{TTSS}}(\lambda)$ *between a challenger and the adversary* $\mathcal{A}$ *is described in Fig.4*

$\mathsf{Exp}^{\mathsf{NI}}_{\mathcal{A},\mathsf{TTSS}}(\lambda)$

$(n, t, i^*, s) \leftarrow \mathcal{A}(1^\lambda)$;
$\rho \xleftarrow{\$} \{0,1\}^\kappa$;
$(\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk}) \leftarrow \mathsf{TTSS.Share}(1^\lambda, s, n, t, \rho)$;
$(\mathcal{I}^*, \pi) \leftarrow \mathcal{A}(\{\mathsf{sh}_i\}_{i=1}^n \backslash \{\mathsf{sh}_{i^*}\}, \mathsf{tk}, \mathsf{vk})$;
if $(i^* \in \mathcal{I}^* \wedge \mathsf{TTSS.Verify}(\mathsf{pp},\mathsf{vk}, \mathcal{I}^*, \pi) \to 1)$ **then**
    return 1;
else
    return 0;
end if

Fig. 4: Non-Imputability experiment for TTSS

3 Traceable Threshold secret sharing

In this section, we show how to trace leakage in a modified version of the classical ABSS scheme. We first introduce a variant of ABSS scheme. We then present a very efficient tracing algorithm for this variant scheme.

3.1 ABSS-I

We begin by recalling the original ABSS scheme [AB83] and then present our variant scheme. In the original ABSS scheme, the secret s is an element of $\mathbb{Z}$. To share a secret $s < m_0$, the dealer first selects a sequence of pairwise relative prime integers $m_0 < m_1 < \cdots < m_n$ such that $\prod_{i=1}^{t} m_i > m_0 \cdot \prod_{i=1}^{t-1} m_{n-i+1}$. The dealer then chooses a random integer A and computes the encoded value $y = s + A \cdot m_0$, ensuring $y < \prod_{i=1}^{t} m_i$. Each participant $i \in [n]$ receives the share $Y_i = y \bmod m_i$. Since the m_i are pairwise co-prime and $y < \prod_{i=1}^{t} m_i$, any t shares can be combined using the CRT to reconstruct y and the secret is recovered as $s = y \bmod m_0$. Any set of fewer than t shares yields almost no information about s, since many possible y values satisfy the given congruences.

However, in our tracing procedure, we adapt the secret representation by expressing the secret as a sum of elements $s_1, s_2, \ldots, s_t$ in $\mathbb{Z}$. Suppose we share each s_i using the naive ABSS approach described above; that is, we encode s_i as $y_i = s_i + A_i \cdot m_0$ using independent randomness $A_1, A_2, \ldots, A_t$ and the

i-th secret share contains t congruences $\{Y_{j,i} = y_j \bmod m_i\}_{j=1}^{t}$. In this case, the same tracing leakage present in the original ABSS scheme reappears. To address this, we make the encoded randomness A_i corresponding to exactly one additive component s_i public and represent its encoded value simply as $y_i = s_i + A_i$. We then apply the naive tracing procedure only to the remaining additive components. This enables effective tracing of leaked shares. Since the remaining randomness values are kept private, the security of the original ABSS scheme is preserved. More precisely, the presence of at least one privately encoded component ensures that the secrecy guarantees of the classical scheme remain intact. We now present the scheme in detail.

Our first construction ABSS-I = (Share, Rec) for a threshold secret sharing scheme based on an additive variant of Asmuth-Bloom secret sharing with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{S}_\lambda \subset \mathbb{Z}$ and works as follows.

- $\mathsf{Share}(1^\lambda, s, n, t, \rho) \to (\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$. The procedure is described in Algorithm 1 which is run by a dealer that takes as input a security parameter λ, a secret $s \in \mathbb{Z}$, the number of parties n, a threshold $t \le n$ and a correlation string $\rho \in \{0,1\}^\kappa$ with $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$ that is interpreted as the randomness for the additive secret sharing of s and outputs a public parameter pp, a share sh_i for the i-th server for $i = 1, 2, \ldots, n$, a tracing key tk and a verification key vk. The public parameter pp and the verification key vk are made public, the tracing key tk is sent to the tracer through a secure communication channel between the dealer and the tracer and the share sh_i is sent to the i-th server through a secure communication channel between the dealer and the i-th server for $i = 1, 2, \ldots, n$.

Algorithm 1: $\mathsf{Share}(1^\lambda, s, n, t, \rho) \to (\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$

Choose a set of pairwise relatively prime integers $m_0 < m_1 < m_2 < \cdots < m_n$ such that $m_0 > s$ is a prime and $M = \prod_{i=1}^{t} m_i > m_0 . \prod_{i=1}^{t-1} m_{n-i+1}$;
Parse $\rho = (s_2, s_3, \ldots, s_t) \in \mathbb{Z}^{t-1}$;
Compute $s_1 = s - (s_2 + s_3 + \cdots + s_t)$;
Sample $A_1 \xleftarrow{\$} \{1, 2, \ldots, M - s_1 - 1\}$;
Compute $y_1 = s_1 + A_1$;
for ($i = 1$ to n) **do**
    Compute $Y_{1,i} = y_1 \bmod m_i$;
end for
for ($j = 2$ to t) **do**
    Sample $A_j \xleftarrow{\$} \{1, 2, \ldots, \left\lfloor \frac{M - s_j}{m_0 - 1} \right\rfloor\}$;
    Compute $y_j = s_j + A_j . m_0$;
    for ($i = 1$ to n) **do**
        Compute $Y_{j,i} = y_j \bmod m_i$;
    end for
end for
for ($i = 1$ to n) **do**

17: Set $\mathsf{sh}_i = (m_i, \{Y_{j,i}\}_{j=1}^t)$;
18: **end for**
19: Set $\mathsf{pp} = \{\{m_i\}_{i=0}^n, A_1\}$, $\mathsf{tk} = \{\{y_i\}_{i=2}^t, \{Y_{1,i}\}_{i=1}^n\}$ and $\mathsf{vk} = \{\{Y_{1,i}\}_{i=1}^n\}$;
20: **return** $(\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$;

– $\mathsf{Rec}(\mathsf{pp}, \{\mathsf{sh}_{i_k}\}_{k=1}^t) \to s$. The procedure is presented in Algorithm 2 which is run by a combiner who outputs a secret $s \in \mathbb{Z}$ on input the public parameter pp and t shares $\{\mathsf{sh}_{i_k}\}_{k=1}^t$.

Algorithm 2: $\mathsf{Rec}(\mathsf{pp}, \{\mathsf{sh}_{i_k}\}_{k=1}^t) \to s$

1: Parse $\mathsf{pp} = \{\{m_i\}_{i=0}^n, A_1\}$; $\mathsf{sh}_i = (m_i, \{Y_{j,i}\}_{j=1}^t)$ $\forall i \in \Lambda = \{i_1, i_2, \ldots, i_t\}$;
2: Compute $M_\Lambda = \prod_{i \in \Lambda} m_i$;
3: **for** $(j = 1$ to $t)$ **do**
4: Find the CRT solution y_j mod M_Λ of the following system of congruences:
$$y_j = Y_{j,i} \bmod m_i \; \forall i \in \Lambda;$$
5: **end for**
6: Compute $s_1 = y_1 - A_1$;
7: **for** $(j = 2$ to $t)$ **do**
8: Compute $s_j = y_j \bmod m_0$;
9: **end for**
10: Compute $s = s_1 + s_2 + \cdots + s_t$;
11: **return** s;

Correctness. Note that,

- $y_1 = s_1 + A_1 < M$ as $A_1 < M - s_1$ (see Step 4 of Algorithm 1)
- $y_j = s_j + A_j.m_0 < M$ as $A_j < \frac{M - s_j}{m_0}$ for $j = 2, 3, \ldots, t$ (see Step 10 of Algorithm 1)
- $M < M_\Lambda$ as $|\Lambda| = t$ (see Step 1 of Algorithm 1)

By CRT, y_j can be uniquely reconstructed modulo M_Λ in Step 4 of Algorithm 2 satisfying $0 \leq y_j < M < M_\Lambda$ for all $j \in [t]$ and $m_0 < m_1 < \cdots < m_n$. Consequently, s_1 can be derived as

$$s_1 = y_1 - A_1,$$

s_j can be derived as

$$s_j = y_j \bmod m_0 \; \forall \, j = 2, 3, \ldots, t$$

and hence s can be uniquely constructed as

$$s = s_1 + s_2 + \cdots + s_t$$

Thus the following theorem holds.

Theorem 1. *Our traceable threshold secret sharing scheme* TTSS-I $= (\mathit{Share}, \mathit{Rec}, \mathit{Trace}, \mathit{Verify})$ *with secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{S}_\lambda \subset \mathbb{Z}$ *is* ϵ*-correct as per Definition 6 in Section 2. More precisely, for every* $\epsilon = \epsilon(\lambda) \in [0,1]$, *every* $\lambda \in \mathbb{N}$, *every secret* s, *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* $\Lambda = \{i_1, i_2, \ldots, i_t\} \subseteq [n]$ *of size* t, *it holds that*

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

where $(\mathit{pp}, \{\mathit{sh}_i\}_{i=1}^n, \mathit{tk}, \mathit{vk}) \leftarrow \mathit{TTSS}\text{-I.}\mathit{Share}(1^\lambda, s, n, t, \rho)$ *and* $s' \leftarrow \mathit{TTSS}\text{-I.}\mathit{Rec}(\mathit{pp}, \{\mathit{sh}_{i_k}\}_{k=1}^t)$ *with* $\epsilon(\lambda) = 0$.

Theorem 2. *Our traceable threshold secret sharing scheme* TTSS-I $= (\mathit{Share}, \mathit{Rec}, \mathit{Trace}, \mathit{Verify})$ *with secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{S}_\lambda \subset \mathbb{Z}$ *satisfies* ϵ*-secrecy as per Definition 7 in Section 2. In other word, for every PPT algorithm* $\mathcal{D}$, *every* $\epsilon = \epsilon(\lambda) \in [0,1]$, *every* $\lambda \in \mathbb{N}$, *any two secrets* s,$\bar{s}$, *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*

$$\Pr[\mathcal{D}(\{\mathit{sh}_i\}_{i\in\Lambda'} = 1) - \mathcal{D}(\{\mathit{sh}'_i\}_{i\in\Lambda'} = 1)] \leq \epsilon(\lambda)$$

where $\rho \in \{0,1\}^\kappa$ *with* $\kappa = \kappa(n,t,\lambda) \in \mathbb{N}$, $(\mathit{pp}, \{\mathit{sh}_i\}_{i=1}^n, \mathit{tk}, \mathit{vk}) \leftarrow \mathit{TTSS\text{-}I.Share}(1^\lambda, s, n, t, \rho)$, $(\mathit{pp}', \{\mathit{sh}'_i\}_{i=1}^n, \mathit{tk}', \mathit{vk}') \leftarrow \mathit{TTSS\text{-}I.Share}(1^\lambda, \bar{s}, n, t, \rho)$ *and* $\epsilon(\lambda) = \frac{1}{m_0}$.

Proof. Assume that a coalition Λ' of size fewer than t servers has gathered about the secret $s \in \mathbb{Z}$ where $(\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk}) \leftarrow \mathsf{TTSS\text{-}I.Share}(1^\lambda, s, n, t, \rho)$ and for every $j \in [t]$, let y'_j be the unique solution modulo $M_{\Lambda'} = \prod_{i\in\Lambda'} m_i$ for the following system of congruences

$$y_j = Y_{j,i} \bmod m_i \; \forall i \in \Lambda'$$

Now,

$$\begin{aligned}
& m_0 \cdot \prod_{i=1}^{t-1} m_{n-i+1} < M \quad \text{(see Step 1 of Algorithm 1)} \\
\Rightarrow \quad & m_0 \cdot M_{\Lambda'} < M \quad (\text{since } m_0 < m_1 < \cdots < m_n) \\
\Rightarrow \quad & y'_j + w \cdot M_{\Lambda'} < M \; \forall w < m_0, \; j \in [t] \\
\Rightarrow \quad & y_j = (y'_j + w \cdot M_{\Lambda'}) \text{ for some } w \in \mathbb{Z}_{m_0} \\
\Rightarrow \quad & s_j = (y'_j + w \cdot M_{\Lambda'}) \bmod m_0 \text{ for some } w \in \mathbb{Z}_{m_0}
\end{aligned}$$

For $j \in [t]$, consider the mappings $F_j : \mathbb{Z}_{m_0} \to \mathbb{Z}_{m_0}$ such that $F_j(w) = (y'_j + w \cdot M_{\Lambda'}) \bmod m_0$. Since m_0 and M are relatively prime, then F_j is a bijection.

$$\begin{aligned}
\Rightarrow \quad & \forall\, j \in [t], s_j = y_j \bmod m_0 \text{ can be any element of } \mathbb{Z}_{m_0} \\
\Rightarrow \quad & \forall\, j \in [t], s_j \text{ is nearly uniformly distributed over } \mathbb{Z}_{m_0}
\end{aligned}$$

Suppose the algorithm $\mathcal{D}$ guess s'_j for s_j, then

$$\Pr[s'_j = s_j] = \frac{1}{m_0} \; \forall j \in [t]$$

which implies s is nearly uniformly distributed over $\mathbb{Z}_{m_0}$ since $s < m_o$.

$$\Pr[s' = s] = \frac{1}{m_0}$$

where $s' = \sum_{j=1}^{t} s'_j$.
Since this holds for every secret $s \in \mathbb{Z}$, the distribution over the $t-1$ shares is identical for any two shares $s, \bar{s}$. Hence the proof.

3.2 TTSS-I

We now describe our tracing algorithm for the ABSS-I scheme introduced above. In ABSS-I, the secret s lies in $\mathbb{Z}$. To generate the n shares, the ABSS-I.Share algorithm samples $s_1, s_2, \ldots, s_t$ such that their sum equals s and encodes them as $y_1 = s_1 + A_1$ and $y_j = s_j + A_j \cdot m_0$ for $j = 2, 3, \ldots, t$, where $A_i \in \mathbb{Z}$ are chosen uniformly at random for $i \in [n]$. The i-th share then consists of t modular congruences $\{Y_{j,i} = y_j \bmod m_i\}_{j=1}^{t}$ corresponding to $y_1, y_2, \ldots, y_t$, respectively. The tracing key tk contains the values $y_2, y_3, \ldots, y_t$ together with the first congruence of each party, $Y_{1,1}, Y_{1,2}, \ldots, Y_{1,n}$. The verification key vk consists of $Y_{1,1}, Y_{1,2}, \ldots, Y_{1,n}$. For simplicity in this informal overview, we assume that $t-1$ secret shares are embedded in the reconstruction box $\mathcal{R}$, which then requires one additional share as input. To perform tracing, the algorithm generates a random share consistent with $y_1, y_2, \ldots, y_t$:

1. for $j = 2, 3, \ldots, t$, it samples a congruence $Y'_j = y_j \bmod m'$ for a randomly chosen modulus m' where $y_2, y_3, \ldots, y_t$ extracted from tk.
2. for $j = 1$, it samples Y'_1 uniformly at random.

The constructed share $\mathsf{sh} = (Y'_1, Y'_2, \ldots, Y'_t)$ is then feed into $\mathcal{R}$, which outputs a reconstructed secret $s' \in \mathbb{Z}$. Assuming $\mathcal{R}$ is perfectly correct, it follows by construction that for each $j \geq 2$, Y'_j matches the corresponding congruence of the shares embedded in $\mathcal{R}$ with respect to y_j. This implies that y''_1 must satisfy the first congruence of the embedded shares where $y''_1 = s'_1 + A_1$ and $s'_1 = s' - (s_2 + s_3 + \cdots + s_t)$. The tracing algorithm identifies all parties $i \in [n]$ for which y''_1 satisfies the first congruence $Y_{1,i}$. Because $\mathcal{R}$ is assumed correct, all corrupted parties will be detected, as y''_1 will necessarily satisfy their first congruence. Conversely, for any honest party i, y''_1 is statistically independent of $Y_{1,i}$ and will satisfy it only with negligible probability, thereby avoiding false accusations.

Our first construction for traceable threshold secret sharing scheme TTSS-I = (Share, Rec, Trace, Verify) with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{S}_\lambda \subset \mathbb{Z}$ described below is based on the secret sharing scheme of additive variant ABSS-I=(Share,

Rec) of Asmuth-Bloom where TTSS-I.Share is same as ABSS-I.Share presented in Algorithm 1 and TTSS-I.Rec is same as ABSS-I.Rec presented in Algorithm 2. For simplicity of presentation, we assume that the number of corruptions $f < t$ hardcoded into the reconstruction box $\mathcal{R}$ is given as input to TTSS-I.Trace.

– $\mathsf{Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon}) \rightarrow (\mathcal{I}', \pi)$. The procedure is provided in Algorithm 3 which is run by a tracer on input the public parameter pp, the tracing key tk, an error parameter ϵ and oracle (black-box) access to a reconstruction box $\mathcal{R}$ containing $f < t$ hardcoded shares of a secret $s \in \mathcal{S}_\lambda$ and outputs a subset $\mathcal{I}' \subset [n]$ identifying the parties responsible for share leakage along with an associated proof π that the parties in $\mathcal{I}'$ are guilty of revealing shares.

Algorithm 3: $\mathsf{Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon}) \rightarrow (\mathcal{I}', \pi)$

Parse $\mathsf{pp} = \{\{m_i\}_{i=0}^n, A_1\}$; $\mathsf{tk} = \{\{y_i\}_{i=2}^t, \{Y_{1,i}\}_{i=1}^n\}$;
for $(l = 1$ to $\frac{2\lambda}{\epsilon})$ **do**
  Sample $y_1' \xleftarrow{\$} \mathbb{N}$;
  for $(i = 1$ to $t - f)$ **do**
    Sample $m_i' \xleftarrow{\$} \mathbb{N}$ relatively prime to $\{m_k\}_{k=1}^n$ and $\{m_k'\}_{k=1}^{i-1}$;
    Compute $Y_{1,i}' = y_1' \bmod m_i'$;
    for $(j = 2$ to $t)$ **do**
      Compute $Y_{j,i}' = y_j \bmod m_i'$;
    end for
    Set $\mathsf{sh}_i' = (m_i', \{Y_{j,i}'\}_{j=1}^t)$;
  end for
  Query $\mathcal{R}$ on $\{\mathsf{sh}_k'\}_{k=1}^{t-f}$ which in turn returns a secret $s' \in \mathbb{Z}$;
  for $(i = 2$ to $t)$ **do**
    Compute $s_i = y_i \bmod m_0$;
  end for
  Compute $s_1' = s' - (s_2 + s_3 + \cdots + s_t) \in \mathbb{Z}$;
  Set $y_1'' = s_1' + A_1 \in \mathbb{Z}$; $\mathcal{I}' = \emptyset$; $\pi = y_1''$;
  for $(i = 1$ to $n)$ **do**
    Compute $Y_{1,i}'' = y_1'' \bmod m_i$;
    if $(Y_{1,i} = Y_{1,i}'')$ **then**
      Set $\mathcal{I}' = \mathcal{I}' \cup \{i\} \subset [n]$;
    end if
  end for
  if $(|\mathcal{I}'| = f)$ **then**
    return $(\mathcal{I}', \pi)$;
  end if
end for
return $\emptyset$;

– $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathcal{I}', \pi) \rightarrow 0/1$. The procedure is given in Algorithm 4 which is run by any public verifier who takes as input the public parameter pp, the

verification key vk, a subset $\mathcal{I}'$ and a proof π and outputs 1 if $\mathcal{I}'$ and π are consistent with vk.

Algorithm 4: $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathcal{I}', \pi) \rightarrow 0/1$

Parse $\mathsf{pp} = \{\{m_i\}_{i=0}^n, A_1\}$; $\mathsf{vk} = \{\{Y_{1,i}\}_{i=1}^n\}$; $\pi = y_1'' \in \mathbb{Z}$;
for $(i = 1$ to $n)$ **do**
    Compute $Y_{1,i}'' = y_1'' \bmod m_i$;
end for
if $(Y_{1,i}'' = Y_{1,i}\ \forall i \in \mathcal{I}') \wedge (Y_{1,i}'' \neq Y_{1,i}\ \forall i \notin \mathcal{I}')$ **then**
    return 1;
else
    return 0;
end if

3.3 Theorem for Universal Traceability

The following theorem establishes the universal traceability of TTSS-I. It proves that TTSS-I is universally traceable with respect to the partition of the correlation strings space into $\Gamma(\rho) = \{\{(y_2, y_3, \ldots, y_t)\} \mid y_2, y_3, \ldots, y_t \in \mathbb{Z}\}$. Standard traceability then follows by Theorem 1 in Section 2.

Theorem 3. *Our traceable threshold secret sharing scheme TTSS-*I $=$ *(Share, Rec, Trace, Verify) with secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{S}_\lambda \subset \mathbb{Z}$ *satisfies universal traceability as per Definition 11 in Section 2. More formally, for every PPT adversary* $\mathcal{A}$*, for every* $\lambda \in \mathbb{N}$ *and* $\epsilon \in [0, 1]$*, it holds that*

$$Adv^{UniTrace}_{\mathcal{A}, TTSS\text{-}I}(\lambda, \epsilon, \Gamma(\rho)) \leq e^{-\lambda} + \frac{2n\lambda}{\epsilon m_0}$$

where $\Gamma(\rho) = \{\rho\} \subset \{0,1\}^\kappa$ *with* $\kappa = \kappa(n, t, \lambda) \in \mathbb{N}$, $n = n(\lambda)$ *is an upper bound on the number of parties and* $m_0 \geq \frac{2n}{\epsilon}$.

Proof. Let $\lambda \in \mathbb{N}$ and $\epsilon \in [0, 1]$ and $\mathcal{A}$ be an adversary that breaks the universal traceability of TTSS-I. The adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ in the security experiment $\mathsf{Exp}^{\mathsf{UniTrace}}_{\mathcal{A}, \mathsf{TTSS\text{-}I}}(\lambda, \epsilon, \Gamma(\rho))$ (see Fig.3) for universal traceability described below.

1. The adversary $\mathcal{A}$ sends $(n, t, \mathcal{I})$ to the challenger $\mathcal{C}$ at the beginning of the experiment taking the security parameter λ as input where n is the number of parties, $t \leq n$ is a threshold and $\mathcal{I} = \{i_1, i_2, \ldots, i_f\} \in [n]$ is a set of f corrupted parties with $0 < f < t \leq n$.
2. The challenger $\mathcal{C}$ selects $s^* \xleftarrow{\$} \mathbb{Z}$ and $\rho \xleftarrow{\$} \{0,1\}^\kappa$ where $\kappa = \kappa(n, t, \lambda)$, runs the algorithm $\mathsf{TTSS\text{-}I.Share}(1^\lambda, s^*, n, t, \rho) \rightarrow (\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$ and sends the shares of the corrupted parties $\mathsf{sh} = \{\mathsf{sh}_j\}_{j \in \mathcal{I}}$ to the adversary $\mathcal{A}$ where,

$$\mathsf{sh}_j = (m_j, \{Y_{k,j} = y_k \bmod m_j\}_{k=1}^t)\ \forall\, j \in \mathcal{I}$$

3. On receiving $\mathsf{sh} = \{\mathsf{sh}_j\}_{j\in\mathcal{I}}$ from $\mathcal{C}$, the adversary $\mathcal{A}$ forms a reconstruction box $\mathcal{R}$. Assume that $\mathcal{R}$ is $(n, t, f, \mathsf{sh}, \mathcal{C}' = \Gamma(\rho), \epsilon)$-good with $\Gamma(\rho) = \{\rho\}$ as per Definition 9 i.e.,

$$\Pr[s' = s'' \mid s' \leftarrow \mathcal{R}(\{\mathsf{sh}'_i\}_{i=1}^{t-f}), s'' \leftarrow \mathsf{TTSS\text{-}I.Rec}(\mathsf{pp}, \{\mathsf{sh}_{i_k}\}_{k=1}^{f}, \{\mathsf{sh}'_i\}_{i=1}^{t-f})] \geq \epsilon$$

with any $t - f$ random shares $\{\mathsf{sh}'_i\}_{i=1}^{t-f}$.

4. The challenger $\mathcal{C}$ runs the algorithm $\mathsf{TTSS\text{-}I.Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon}) \rightarrow (\mathcal{I}', \pi)$

Note that in each iteration of Step 2 of tracing algorithm (Algorithm 3), the challenger $\mathcal{C}$ feeds $\mathcal{R}$ with uniformly random shares $sh'_1, sh'_2, \ldots, sh'_{t-f}$ in Step 10 of $\mathsf{TTSS\text{-}I.Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon})$ conditioned on the fact that $\rho = (s_2, s_3, \ldots, s_t) \in \mathbb{Z}^{t-1}$ and

$$\mathsf{sh}'_i = (m'_i, \{Y'_{1,i} = y'_1 \bmod m'_i\} \cup \{Y'_{j,i} = y_j \bmod m'_i\}_{j=2}^{t}) \; \forall \, i \in [t-f]$$

- Using these $t - f$ shares $\{sh'_i\}_{i=1}^{t-f}$ and the f many shares $sh = \{sh_j\}_{j\in\mathcal{I}}$ corresponding to the corrupted parties obtained by $\mathcal{C}$, let the reconstruction by $\mathcal{R}$ outputs $s' \leftarrow \mathcal{R}(\{\mathsf{sh}'_i\}_{i=1^{t-f}})$ and let $s'' \leftarrow \mathsf{TTSS\text{-}I.Rec}(\mathsf{pp}, \mathsf{sh} = \{\mathsf{sh}_j\}_{j\in\mathcal{I}} \cup \{\mathsf{sh}'_i\}_{i=1}^{t-f})$. Note the following,
- Since $\mathcal{R}$ is $(n, t, f, sh, \Gamma(\rho) = \{\rho\}, \epsilon)$ - good, in each iteration it holds that $s' = s''$ with probability at least ϵ.
- By the correctness of the reconstruction algorithm $\mathsf{TTSS\text{-}I.Rec}(\mathsf{pp}, \{\mathsf{sh}_i\}_{i\in\mathcal{I}} \cup \{\mathsf{sh}'_i\}_{i=1}^{t-f})$ we get,

$$s'' = \hat{s}_1 + s_2 + \cdots + s_t$$

for some $\hat{s}_1 \in \mathbb{Z}$.

- Note that $s_j = y_j \bmod m_0$ for $j = 2, 3, \ldots, t$ where y_j is the unique CRT solution modulo $\Lambda{=}m_{i_1} m_{i_2} \ldots m_{i_f} m'_1 m'_2 \ldots m'_{t-f}$ to the system of congruences

$$\begin{cases} y_j = Y_{j,i_1} \bmod m_{i_1} \\ y_j = Y_{j,i_2} \bmod m_{i_2} \\ \quad\vdots \\ y_j = Y_{j,i_f} \bmod m_{i_f} \\ y_j = Y'_{j,1} \bmod m'_1 \\ y_j = Y'_{j,2} \bmod m'_2 \\ \quad\vdots \\ y_j = Y'_{j,t-f} \bmod m'_{t-f} \end{cases}$$

and $\hat{s}_1 = \hat{y}_1 - A_1$ where $\hat{y}_1$ is the unique CRT solution modulo $\Lambda = m_{i_1} m_{i_2} \ldots m_{i_f} m'_1 m'_2 \ldots m'_{t-f}$ to the system of congruences

$$\begin{cases} \hat{y}_1 = y_1 = Y_{1,i_1} \bmod m_{i_1} \\ \hat{y}_1 = y_1 = Y_{1,i_2} \bmod m_{i_2} \\ \quad \vdots \\ \hat{y}_1 = y_1 = Y_{1,i_f} \bmod m_{i_f} \\ \hat{y}_1 = y'_1 = Y'_{1,1} \bmod m'_1 \\ \hat{y}_1 = y'_1 = Y'_{1,2} \bmod m'_2 \\ \quad \vdots \\ \hat{y}_1 = y'_1 = Y'_{1,t-f} \bmod m'_{t-f} \end{cases}$$

- As $s' = s''$ with probability $\geq \epsilon$, we get

  $$s' = \hat{s}_1 + s_2 + \cdots + s_t$$

  with probability $\geq \epsilon$. Which in turn implies

  $$\hat{s}_1 = s' - (s_2 + s_3 + \cdots + s_t)$$

- Also the algorithm $\mathsf{TTSS\text{-}I.Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon})$ computes s'_1 as

  $$s'_1 = s' - (s_2 + s_3 + \cdots + s_t) \qquad \text{(Step 16 of Algorithm 3)}$$

  and therefore,

  $$s'_1 = \hat{s}_1 \qquad (\text{as } \hat{s}_1 = s' - (s_2 + s_3 + \cdots + s_t)\ )$$

Hence, for $k = 1, 2, \ldots, f$

$$\begin{aligned} Y''_{1,i_k} &= y''_1 \bmod m_{i_k} && \text{(by Step 19 of Algorithm 3)} \\ &= (s'_1 + A_1) \bmod m_{i_k} && \text{(by Step 17 of Algorithm 3)} \\ &= (\hat{s}_1 + A_1) \bmod m_{i_k} && (\text{as } s'_1 = \hat{s}_1) \\ &= \hat{y}_1 \bmod m_{i_k} && (\text{as } \hat{s}_1 = \hat{y}_1 - A_1) \\ &= y_1 \bmod m_{i_k} && (\text{as } \hat{y}_1 \text{ is a unique } \mathsf{CRT} \text{ solution modulo } \Lambda \text{ of the} \\ & && \text{system of equations}) \\ &= Y_{1,i_k} && (\text{extracted from } tk = \{\{y_i\}_{i=2}^{t}, \{Y_{1,i}\}_{i=1}^{n}\}) \end{aligned}$$

Thus the verification Step 20-22 in the algorithm $\mathsf{TTSS\text{-}I.Trace}$ succeeds for all $\{Y_{1,j}\}_{j \in \mathcal{I} = \{i_1, i_2, \ldots, i_f\}}$. This means that all corrupted parties $i_1, i_2, \ldots, i_f \in \mathcal{I}$ are included in the set $\mathcal{I}'$ computed by $\mathsf{TTSS\text{-}I.Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon})$.

On the other hand, for every party $i \in [n] \setminus \mathcal{I}$, it holds that the set $\{Y_{j,i}\}_{j=1}^{t}$ is statistically independent from the view of $\mathcal{A}$ as well as from the shares $\mathsf{sh}'_1, \mathsf{sh}'_2, \ldots, \mathsf{sh}'_{t-f}$ sampled by $\mathsf{TTSS\text{-}I.Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon})$ and hence, is independent of

y_1, resulting $\Pr[Y'_{1,i} = Y_{1,i}] = \frac{1}{m_i}$. Therefore, by a union bound, the probability that any honest party is included in $\mathcal{I}'$ is at most $\frac{t}{m_0}$.

In each iteration, the probability that $\mathsf{TTSS\text{-}I.Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon})$ outputs the correct subset $\mathcal{I}' = \mathcal{I}$ is at least $\epsilon - \frac{t}{m_0} \geq \frac{\epsilon}{2}$. Hence, the probability that it computes the correct subset in at least one iteration out of $l = \frac{2\lambda}{\epsilon}$ iterations is at least

$$1 - \left(1 - \frac{\epsilon}{2}\right)^l \geq 1 - e^{-\frac{\epsilon l}{2}} = 1 - e^{-\lambda}.$$

Moreover, the probability that it outputs the wrong subset $\mathcal{I}' \neq \mathcal{I}$ in a given iteration is at most $\frac{n}{m_0}$ and hence the probability that it outputs the wrong subset in any of the $l = \frac{2\lambda}{\epsilon}$ iterations is at most $\frac{2n\lambda}{\epsilon m_0}$. Overall, the advantage of $\mathcal{A}$ is bounded by $e^{-\lambda} + \frac{2n\lambda}{\epsilon m_0}$, concluding the proof.

4 Traceable Threshold secret sharing with Non-imputability

4.1 ABSS-II

In the scheme described above, possession of the tracing key tk would allow an adversary to falsely accuse an honest party in contributing its share to a pirate reconstruction box $\mathcal{R}$, thereby violating the non-imputability property (Definition 12). This occurs because tk contains the first modular congruence $Y_{1,i}$ of every party $i \in [n]$. However, full knowledge of $Y_{1,i}$ is not necessary for tracing. It suffices to determine whether a particular point-specifically, y''_1 reconstructed by the Trace algorithm, satisfies the congruence $Y_{1,i}$. To achieve this, we store in the tracing key only $f(Y_{1,1}), f(Y_{1,2}), \ldots, f(Y_{1,n})$, where f is a homomorphic one-way function. The tracing key tk and verification key vk both contain these function evaluations with tk additionally including $y_2, y_3, \ldots, y_t$ as before. The homomorphic property enables Trace and Verify to check whether $y''_1 \bmod m_i = Y_{1,i}$ indirectly, without revealing $Y_{1,i}$. Intuitively, if party i is honest, the reconstruction box $\mathcal{R}$ is statistically independent of $Y_{1,i}$. Thus, falsely accusing such a party would require discovering a non-trivial linear relation among $f(Y_{1,1}), f(Y_{1,2}), \ldots, f(Y_{1,n})$, which is equivalent to inverting the one-way function f. In our concrete instantiation, f is the exponentiation function in a cyclic group, whose one-wayness relies on the hardness of the discrete logarithm problem. For asymptotic analysis, we assume a distribution ensemble over discrete log hard groups. We now describe the scheme in detail.

Our second construction $\mathsf{ABSS\text{-}II} = (\mathsf{Share}, \mathsf{Rec})$ is a threshold secret sharing scheme based on an additive variant of the Asmuth-Bloom secret sharing scheme with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda \in \mathbb{N}}, \mathcal{S}_\lambda \subset \mathbb{Z}$, that additionally incorporates non-imputability. The scheme $\mathsf{ABSS\text{-}II}$ extends $\mathsf{ABSS\text{-}I}$ presented in Algorithm 1 and 2 by additionally generating a group $\mathbb{G}$ generated by g of prime order p and outputs a tracing key tk and a verification key vk, leveraging the hardness assumption on g to address leakage issues in $\mathsf{TTSS\text{-}I}$ presented in Algorithm 3

and 4. In the description of the ABSS-II algorithm, we use boxes to highlight the differences from ABSS-I. The share generation algorithm Share and the secret reconstruction algorithm Rec operate as follows.

– $\mathsf{Share}(1^\lambda, s, n, t, \rho) \to (\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$. The procedure run by a dealer is described in Algorithm 5 which takes as input a security parameter λ, a secret $s \in \mathbb{Z}$, the number of parties n, a threshold $t \leq n$ and a correlation string $\rho \in \{0,1\}^\kappa$ with $\kappa = \kappa(n, t, \lambda) \in \mathbb{N}$ that is interpreted as the randomness for the additive secret sharing of s and outputs a public parameter $\mathsf{pp} = \{(\mathbb{G}, g, p), \{m_i\}_{i=0}^n, A_1\}$, a share $\mathsf{sh}_i = (m_i, \{Y_{j,i}\}_{j=1}^t)$ for the i-th server for $i = 1, 2, \ldots, n$, a tracing key $\mathsf{tk} = \{\{y_i\}_{i=2}^t, \{Z_i\}_{i=1}^n\}$ and a verification key $\mathsf{vk} = \{\{Z_i\}_{i=1}^n\}$. The public parameter pp and the verification key vk are made public, the tracing key tk is sent to the tracer through a secure communication channel between the dealer and the tracer and the share sh_i is sent to the i-th server through a secure communication channel between the dealer and the i-th server for $i = 1, 2, \ldots, n$.

Algorithm 5: $\mathsf{Share}(1^\lambda, s, n, t, \rho) \to (\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$

Choose a set of pairwise relatively prime integers $m_0 < m_1 < m_2 < \cdots < m_n$ such that $m_0 > s$ is a prime and $M = \prod_{i=1}^t m_i > m_0 . \prod_{i=1}^{t-1} m_{n-i+1}$;
Sample $\boxed{(\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}(1^\lambda)}$ where $\mathbb{G}$ is a cyclic group of p elements generated by $g > m_n$;
Parse $\rho = (s_2, s_3, \ldots, s_t) \in \mathbb{Z}^{t-1}$;
Compute $s_1 = s - (s_2 + s_3 + \cdots + s_t)$;
Sample $A_1 \xleftarrow{\$} \{1, 2, \ldots, M - s_1 - 1\}$;
Compute $y_1 = s_1 + A_1$;
for $(i = 1$ to $n)$ **do**
    Compute $Y_{1,i} = y_1 \bmod m_i$;
    Compute $\boxed{Z_i = g^{Y_{1,i}}}$;
end for
for $(j = 2$ to $t)$ **do**
    Sample $A_j \xleftarrow{\$} \{1, 2, \ldots \left\lfloor \frac{M - s_j}{m_0 - 1} \right\rfloor\}$;
    Compute $y_j = s_j + A_j . m_0$;
    for $(i = 1$ to $n)$ **do**
        Compute $Y_{j,i} = y_j \bmod m_i$;
    end for
end for
Set $\mathsf{pp} = \{\boxed{(\mathbb{G}, g, p)}, \{m_i\}_{i=0}^n, A_1\}$;
for $(i = 1$ to $n)$ **do**
    Set $\mathsf{sh}_i = (m_i, \{Y_{j,i}\}_{j=1}^t)$;
end for

22: Set $\mathsf{tk} = \{\{y_i\}_{i=2}^{t}, \boxed{\{Z_i\}_{i=1}^{n}}\}$ and $\mathsf{vk} = \{\boxed{\{Z_i\}_{i=1}^{n}}\}$;
23: **return** $(\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^{n}, \mathsf{tk}, \mathsf{vk})$;

– $\mathsf{Rec}(\mathsf{pp}, \{\mathsf{sh}_{i_k}\}_{k=1}^{t}) \rightarrow s$. The procedure run by a combiner is presented in Algorithm 6 which outputs a secret $s \in \mathbb{Z}$ on input the public parameter $\mathsf{pp} = \{(\mathbb{G}, g, p), \{m_i\}_{i=0}^{n}, A_1\}$ and t shares $\{\mathsf{sh}_{i_k} = (m_{i_k}, \{Y_{j,i_k}\}_{j=1}^{t})\}_{k=1}^{t}$.

Algorithm 6: $\mathsf{Rec}(\mathsf{pp}, \mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_t}) \rightarrow s$

1: Parse $\mathsf{pp} = \{\boxed{(\mathbb{G}, g, p)}, \{m_i\}_{i=0}^{n}, A_1\}$; $\mathsf{sh}_i = (m_i, \{Y_{j,i}\}_{j=1}^{t})$ $\forall i \in \Lambda = \{i_1, i_2 \ldots, i_t\}$;
2: Compute $M_\Lambda = \prod_{i \in \Lambda} m_i$;
3: **for** $(j = 1$ to $t)$ **do**
4: Find the CRT solution y_j mod M_Λ of the following system of congruences:

$$y_j = Y_{j,i} \bmod m_i \; \forall i \in \Lambda;$$

5: **end for**
6: Compute $s_1 = y_1 - A_1$;
7: **for** $(j = 2$ to $t)$ **do**
8: Compute $s_j = y_j \bmod m_0$;
9: **end for**
10: Compute $s = s_1 + s_2 + \cdots + s_t$;
11: **return** s;

We note that ABSS-II achieves the same correctness and security guarantees as ABSS-I.

4.2 TTSS-II

Our second construction for traceable threshold secret sharing scheme TTSS-II = (Share, Rec, Trace, Verify) with secret space $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{S}_\lambda \subset \mathbb{Z}$ described below is based on the secret sharing scheme of additive variant ABSS-II = (Share, Rec) of Asmuth-Bloom and additionally exhibits non-imputability where TTSS-II.Share is same as ABSS-II.Share presented in Algorithm 5 and TTSS-II.Rec is same as ABSS-II.Rec presented in Algorithm 6. We use box in the algorithm of TTSS-II to highlight the difference of TTSS-II from TTSS-I.

– $\mathsf{Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon}) \rightarrow (\mathcal{I}', \pi)$. The procedure run by a tracer is provided in Algorithm 7 which takes on input the public parameter $\mathsf{pp} = \{(\mathbb{G}, g, p), \{m_i\}_{i=0}^{n}, A_1\}$, the tracing key $\mathsf{tk} = \{\{y_i\}_{i=2}^{t}, \{Z_i\}_{i=1}^{n}\}$, an error parameter ϵ and oracle (black-box) access to a reconstruction box $\mathcal{R}$ containing $f < t$ hardcoded shares of a secret $s \in \mathcal{S}_\lambda$ and outputs a subset $\mathcal{I}' \subset [n]$ identifying the parties responsible for share leakage along with an associated proof π that the parties in $\mathcal{I}'$ are guilty of revealing shares.

Algorithm 7: $\mathsf{Trace}^{\mathcal{R}}(\mathsf{pp}, \mathsf{tk}, f, 1^{1/\epsilon}) \rightarrow (\mathcal{I}', \pi)$

Parse $\mathsf{pp} = \{\boxed{(\mathbb{G}, g, p)}, \{m_i\}_{i=0}^{n}, A_1\}$; $\mathsf{tk} = \{\{y_i\}_{i=2}^{t}, \boxed{\{Z_i\}_{i=1}^{n}}\}$;
for $(l = 1$ to $\frac{2\lambda}{\epsilon})$ **do**
  Sample $y_1' \xleftarrow{\$} \mathbb{N}$;
  for $(i = 1$ to $t - f)$ **do**
    Sample $m_i' \xleftarrow{\$} \mathbb{N}$ relatively prime to $\{m_k\}_{k=1}^{n}$ and $\{m_k'\}_{k=1}^{i-1}$;
    Compute $Y_{1,i}' = y_1' \bmod m_i'$;
    for $(j = 2$ to $t)$ **do**
      Compute $Y_{j,i}' = y_j \bmod m_i'$;
    end for
    Set $\mathsf{sh}_i' = (m_i', \{Y_{j,i}'\}_{j=1}^{t})$;
  end for
  Query $\mathcal{R}$ on $\{\mathsf{sh}_k'\}_{k=1}^{t-f}$ which in turn returns a secret $s' \in \mathbb{Z}$;
  for $(i = 2$ to $t)$ **do**
    Compute $s_i = y_i \bmod m_0$;
  end for
  Compute $s_1' = s' - (s_2 + s_3 + \cdots + s_t) \in \mathbb{Z}$;
  Set $y_1'' = s_1' + A_1 \in \mathbb{Z}$; $\mathcal{I}' = \emptyset$; $\pi = y_1''$;
  for $(i = 1$ to $n)$ **do**
    Compute $\boxed{Z_i' = g^{y_1'' \bmod m_i}}$;
    if $(Z_i = Z_i')$ **then**
      Set $\mathcal{I}' = \mathcal{I}' \cup \{i\} \subset [n]$;
    end if
  end for
  if $(|\mathcal{I}'| = f)$ **then**
    return $(\mathcal{I}', \pi)$;
  end if
end for
return $\emptyset$;

– $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathcal{I}', \pi) \rightarrow 0/1$. The procedure run by any public verifier is given in Algorithm 8 which takes as input the public parameter $\mathsf{pp} = \{(\mathbb{G}, g, p), \{m_i\}_{i=0}^{n}, A_1\}$, the verification key $\mathsf{vk} = \{\{Z_i\}_{i=1}^{n}\}$, a subset $\mathcal{I}' \subset [n]$ and a proof π and outputs 1 if $\mathcal{I}'$ and π are consistent with vk.

Algorithm 8: $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathcal{I}', \pi) \rightarrow 0/1$

Parse $\mathsf{pp} = \{\boxed{(\mathbb{G}, g, p)}, \{m_i\}_{i=0}^{n}, A_1\}$; $\mathsf{vk} = \{\boxed{\{Z_i\}_{i=1}^{n}}\}$; $\pi = y_1'' \in \mathbb{Z}$;
for $(i = 1$ to $n)$ **do**
  Compute $\boxed{Z_i' = g^{y_1'' \bmod m_i}}$;
end for

5: **if** $(Z'_i = Z_i \, \forall i \in \mathcal{I}') \wedge (Z'_i \neq Z_i \, \forall i \notin \mathcal{I}')$ **then**
6: return 1;
7: **else**
8: return 0;
9: **end if**

4.3 Theorem for Non-Imputability

The following theorem establishes the non-imputability of TTSS-II.

Theorem 4. *Our traceable threshold secret sharing scheme* TTSS-II $= (\mathit{Share}, \mathit{Rec}, \mathit{Trace}, \mathit{Verify})$ *with secret space* $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{S}_\lambda \subset \mathbb{Z}$ *presented above satisfies non-imputability as per Definition 11 in Section 2. In other words, for every* $\lambda \in \mathbb{N}$, *for every* PPT *adversary* $\mathcal{A}$ *involved in the non-imputability experiment presented in Fig.4, there exists an* PPT *algorithm* $\mathcal{B}$ *involved in the* DL *experiment presented in Fig.1 such that*

$$\mathit{Adv}^{\mathit{NI}}_{\mathcal{A},\mathit{TTSS}\text{-II}}(\lambda) = \mathit{Adv}^{\mathit{DL}}_{\mathcal{B},\mathcal{G}}(\lambda).$$

Proof. Let $\lambda \in \mathbb{N}$ and $\mathcal{A}$ be an adversary that breaks the non-imputability security of TTSS-II. We construct an algorithm $\mathcal{B}$ that solves the DL experiment $\mathsf{Exp}^{\mathsf{DL}}_{\mathcal{B},\mathcal{G}}(\lambda)$ with respect to $\mathcal{G}$. The algorithm $\mathcal{B}$ gets as input an element g^x sampled uniformly at random from the group $\mathbb{G} = \langle g \rangle$ of order p where p is a large prime and plays the role of challenger in the non-imputability experiment.

1. The adversary $\mathcal{A}$ on input 1^λ and outputs the number of parties $n \in \mathbb{N}$, a threshold $t \leq n$, a party $i^* \in [n]$ and a secret $s \in \mathbb{Z}_{m_0}$ and sends (n, t, i^*, s) to the algorithm $\mathcal{B}$.
2. The algorithm $\mathcal{B}$ selects $\rho \xleftarrow{\$} \{0,1\}^\kappa$ where $\kappa = \kappa(n, t, \lambda) \in \mathbb{N}$.
3. The algorithm $\mathcal{B}$ then generates a tracing key tk^*, a verification key vk^* and the shares $\{\mathsf{sh}_i\}_{i \in [n] \setminus \{i^*\}}$ for $\mathcal{A}$
   - Run the algorithm $\mathsf{TTSS\text{-}II.Share}(1^\lambda, s, n, t, \rho) \to (\mathsf{pp}, \{\mathsf{sh}_i\}_{i=1}^n, \mathsf{tk}, \mathsf{vk})$ where

$$\begin{aligned} \mathsf{pp} &= \{(\mathbb{G}, g, p), m_0, m_1, \ldots, m_n, A_1\}, \\ \mathsf{sh}_i &= (m_i, \{Y_{j,i}\}_{j=1}^t) \quad \forall i \in [n], \\ \mathsf{tk} &= \left\{\{y_i\}_{i=2}^t, \{Z_i\}_{i=1}^n\right\}, \\ \mathsf{vk} &= \{\{Z_i\}_{i=1}^n\} \end{aligned}$$

   where $\{m_i\}_{i=0}^n$ are relatively prime integers such that $M = \prod_{i=1}^t m_i > m_0 . \prod_{i=1}^{t-1} m_{n-i+1}$ with m_0 is prime,

$$Z_i = g^{Y_{1,i}},\ Y_{j,i} = y_j \bmod m_i\ , i \in [n],\ \forall\, j = 2, 3, \ldots, t$$

$$y_1 = s_1 + A_1,\ y_j = s_j + A_j \cdot m_0 \,\forall\, j = 2, 3, \ldots, t$$

   where $\{A_j\}_{j=1}^t$ are random integers, the algorithm selects a public function $\mathcal{F} : \{0,1\}^\kappa \times \mathbf{R} \to \mathbb{Z}^{t-1}$ on a randomness $r \in \mathbf{R}$ to compute $\mathcal{F}(\rho, r) = (s_2, s_3, \ldots, s_t) \in \mathbb{Z}^{t-1}$ and $s_1 = s - (s_2 + s_3 + \cdots + s_t)$.

- The algorithm $\mathcal{B}$ sets $Z_{i^*} = g^x$ for $i = i^*$, $\mathsf{tk}^* = \{\{y_i\}_{i=2}^t, \{Z_i\}_{i=1}^n\}$ and $\mathsf{vk}^* = (\{Z_i\}_{i=1}^n)$ and gives $\mathcal{A}$ the shares $\{\mathsf{sh}_i\}_{i\in[n]\setminus\{i^*\}}$, the modified tracing key tk^* and the modified verification key vk^*.

4. The adversary $\mathcal{A}$ then outputs a proof π implicating a subset $\mathcal{I}^*$ that includes party i^*.

If π is not an accepting proof for implicating party i^*, $\mathcal{B}$ aborts and outputs $\perp$. For $\mathcal{A}$ to win the security experiment, it must be that π is y_1^* such that:

$$g^{y_1^* \bmod m_i^*} = Z_{i^*}$$

Recalling that $Z_{i^*} = g^x$, this implies that $y_1^* \bmod m_i^* = x \bmod p$. Hence, if not aborted before, $\mathcal{B}$ outputs $y_1^* \bmod m_i^*$.

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

$$\mathsf{Adv}^{\mathsf{NI}}_{\mathcal{A},\mathsf{TTSS\text{-}II}}(\lambda) = \mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathcal{G}}(\lambda),$$

completing the proof of the theorem.

5 Conclusion

This paper advances the construction of traceable secret sharing schemes by introducing an additive variant of the Asmuth-Bloom secret sharing scheme. We address the leakage present in the original Asmuth-Bloom scheme and remedy it by proposing a variant of the Asmuth-Bloom secret sharing scheme that preserves its security properties. Building on this variant, we present a traceable secret sharing scheme TTSS-I using only black-box access. Next, we address the violation of the non-imputability property in TTSS-I. To overcome this, we introduce an enhanced traceable scheme TTSS-II which satisfies the non-imputability requirement. This scheme performs all verification operations within the scope of exponentiation and its security is based on the hardness of the discrete logarithm problem.

CRediT authorship contribution statement

Maria Leslie: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing. **Ratna Dutta:** Formal analysis, Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

The first author was partially supported by the Council of University Grants Commission (UGC) fellowship (211610187990) of the Government of India.

References

[AB83] Charles Asmuth and John Bloom. "A Modular Approach to Key Safeguarding". In: *IEEE Transactions on Information Theory* 29.2 (1983), pp. 208–210. DOI: 10.1109/TIT.1983.1056650.

[BF99] Dan Boneh and Matthew K. Franklin. "An Efficient Public Key Traitor Tracing Scheme". In: *Advances in Cryptology - CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science (LNCS). Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1999, pp. 338–353.

[Bla79] G. R. Blakley. "Safeguarding Cryptographic Keys". In: *Proceedings of the National Computer Conference*. Vol. 48. 1979, pp. 313–317.

[BPR24] Dan Boneh, Aditi Partap, and Lior Rotem. "Traceable Secret Sharing: Strong Security and Efficient Constructions". In: *Cryptology ePrint Archive* Paper 2024/405 (2024). To appear at CRYPTO'24. URL: https://eprint.iacr.org/2024/405.

[BS95] Dan Boneh and James Shaw. "Collusion-Secure Fingerprinting for Digital Data (Extended Abstract)". In: *Advances in Cryptology - CRYPTO'95*. Ed. by D. Coppersmith. Vol. 963. Lecture Notes in Computer Science (LNCS). Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1995, pp. 452–465.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. "Fully collusion resistant traitor tracing with short ciphertexts and private keys". In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. Heidelberg: Springer, May 2006, pp. 573–592.

[BZ14] Dan Boneh and Mark Zhandry. "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation". In: *Advances in Cryptology – CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Heidelberg: Springer, Aug. 2014, pp. 480–499.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. "Tracing Traitors". In: *Advances in Cryptology – CRYPTO '94*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Heidelberg, Germany: Springer, Aug. 21–25, 1994, pp. 257–270.

[CPP05] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. "Public traceability in traitor tracing schemes". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Heidelberg: Springer, May 2005, pp. 542–558.

[DF03] Yevgeniy Dodis and Nelly Fazio. "Public key trace and revoke scheme secure against adaptive chosen ciphertext attack". In: *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Yvo Desmedt. Vol. 2567. Lecture Notes in Computer Science. Heidelberg: Springer, Jan. 2003, pp. 100–115.

[FT99] Amos Fiat and Tamir Tassa. "Dynamic Traitor Tracing". In: *Advances in Cryptology - CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science (LNCS). Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1999, pp. 354–371.

[Gar+23] Sanjam Garg et al. "Cryptography with weights: MPC, encryption and signatures". In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. Lecture Notes in Computer Science. Heidelberg, Germany: Springer, Aug. 2023, pp. 295–327.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. "Collusion resistant traitor tracing from learning with errors". In: *50th Annual ACM Symposium on Theory of Computing*. Ed. by Ilias Diakonikolas, David Kempe, and Monika Henzinger. ACM Press, June 2018, pp. 660–670.

[GLW23] Junqing Gong, Ji Luo, and Hoeteck Wee. "Traitor Tracing with $N^{1/3}$-Size Ciphertexts and $O(1)$-Size Keys from k-Lin". In: *Advances in Cryptology - EUROCRYPT 2023, Part III*. Lecture Notes in Computer Science (LNCS). Springer, Heidelberg, Germany, June 2023, pp. 637–668.

[GSS21] V. Goyal, Y. Song, and A. Srinivasan. "Traceable Secret Sharing and Applications". In: *Advances in Cryptology - CRYPTO 2021, Part III*. Ed. by T. Malkin and C. Peikert. Vol. 12827. Lecture Notes in Computer Science (LNCS). Virtual Event: Springer, Heidelberg, Germany, Aug. 2021, pp. 718–747.

[Hof24] Charlotte Hoffmann. *Traceable Secret Sharing Based on the Chinese Remainder Theorem*. Cryptology ePrint Archive, Paper 2024/811. 2024. URL: https://eprint.iacr.org/2024/811.

[KD98] Kaoru Kurosawa and Yvo Desmedt. "Optimum Traitor Tracing and Asymmetric Schemes". In: *Advances in Cryptology - EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science (LNCS). Espoo, Finland: Springer, Heidelberg, Germany, May 1998, pp. 145–157.

[KY01] Aggelos Kiayias and Moti Yung. "Self Protecting Pirates and Black-Box Traitor Tracing". In: *Advances in Cryptology - CRYPTO 2001*.

Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science (LNCS). Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 63–79.

[KY02] Aggelos Kiayias and Moti Yung. "Traitor Tracing with Constant Transmission Rate". In: *Advances in Cryptology - EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science (LNCS). Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 2002, pp. 450–465.

[McC90] Kevin S. McCurley. "The Discrete Logarithm Problem". In: *Cryptology and Computational Number Theory*. Vol. 42. Proceedings of Symposia in Applied Mathematics. American Mathematical Society, 1990, pp. 49–74. DOI: 10.1090/psapm/042/1069793.

[Mig82] M. Mignotte. "How to Share a Secret". In: (1982), pp. 371–375.

[NP98] Moni Naor and Benny Pinkas. "Threshold Traitor Tracing". In: *Advances in Cryptology - CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science (LNCS). Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1998, pp. 502–517.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. "Anonymous traitor tracing: How to embed arbitrary information in a key". In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Heidelberg: Springer, May 2016, pp. 388–419.

[OBr90] E.A. O'Brien. "The p-group generation algorithm". In: *Journal of Symbolic Computation* 9.5 (1990), pp. 677–698. ISSN: 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(08)80082-X.

[Sha79] A. Shamir. "How to Share a Secret". In: *Communications of the ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176.

[Wan98] Yuke Wang. "New Chinese remainder theorems". In: *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers (Cat. No.98CH36284)*. Vol. 1. 1998, 165–171 vol.1. DOI: 10.1109/ACSSC.1998.750847.

[Wee20] Hoeteck Wee. "Functional Encryption for Quadratic Functions from k-Lin, Revisited". In: *TCC 2020: 18th Theory of Cryptography Conference, Part I*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12550. Lecture Notes in Computer Science (LNCS). Durham, NC, USA: Springer, Heidelberg, Germany, Nov. 2020, pp. 210–228.

[Zha20] Mark Zhandry. "New Techniques for Traitor Tracing: Size $N^{1/3}$ and More from Pairings". In: *Advances in Cryptology - CRYPTO 2020, Part I*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12170. Lecture Notes in Computer Science (LNCS). Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2020, pp. 652–682.

[Zou+11] X. Zou et al. "A new approach to weighted multi-secret sharing". In: *Proceedings of the 20th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2011, pp. 1–6.