Traceable Secret Sharing: Strong Security and Efficient Constructions

Dan Boneh, Aditi Partap, and Lior Rotem

Stanford University

{dabo,aditi712,lrotem}@cs.stanford.edu

Abstract. Suppose Alice uses a t-out-of-n secret sharing to store her secret key on n servers. Her secret key is protected as long as t of them do not collude. However, what if a less-than-t subset of the servers decides to offer the shares they have for sale? In this case, Alice should be able to hold them accountable, or else nothing prevents them from selling her shares. With this motivation in mind, Goyal, Song, and Srinivasan (CRYPTO 21) introduced the concept of *traceable secret sharing*. In such schemes, it is possible to provably trace the leaked secret shares back to the servers who leaked them. Goyal et al. presented the first construction of a traceable secret sharing scheme. However, secret shares in their construction are quadratic in the secret size, and their tracing algorithm is quite involved as it relies on Goldreich-Levin decoding.

In this work, we put forth new definitions and practical constructions for traceable secret sharing. In our model, some $f < t$ servers output a reconstruction box R that may arbitrarily depend on their shares. Given additional $t - f$ shares, R reconstructs and outputs the secret. The task is to trace R back to the corrupted servers given black-box access to R. Unlike Goyal et al., we do not assume that the tracing algorithm has any information on how the corrupted servers constructed R from the shares in their possession.

We then present two very efficient constructions of traceable secret sharing based on two classic secret sharing schemes. In both of our schemes, shares are only twice as large as the secret, improving over the quadratic overhead of Goyal et al. Our first scheme is obtained by presenting a new practical tracing algorithm for the widely-used Shamir secret sharing scheme. Our second construction is based on an extension of Blakley's secret sharing scheme. Tracing in this scheme is optimally efficient, and requires just one successful query to R. We believe that our constructions are an important step towards bringing traceable secret-sharing schemes to practice. This work also raises several interesting open problems that we describe in the paper.

1 Introduction

Secret sharing [65, 5] is one of the most foundational concepts in cryptography. Of particular use are threshold secret-sharing schemes in which a secret s is split into n shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ so that any t shares suffice to reconstruct the secret, but any $t-1$ of them reveal nothing about it. A fundamental use case of threshold secret sharing is storing confidential data on untrusted servers. Concretely, say that Alice stores the key to her crypto wallet (or any other piece of sensitive information) on n servers or cloud storage providers using a t-out-of-n secret sharing scheme.

Now suppose that a subset of f servers collude and sell the key shares in their possession to whomever is willing to meet their price. In this worrisome scenario, we would like some mechanism to enable Alice to hold these servers accountable. Otherwise, they would have a risk-free incentive to sell Alice's secret information with no way to hold them accountable for their action.

Traceable secret sharing. This question was recently considered by Goyal, Song, and Srinivasan [41], who introduced the concept of *traceable secret sharing*. Roughly speaking, these are secret sharing schemes where leaked secret shares can be traced back to the servers who leaked them. We will make this requirement precise shortly.

In some secret sharing schemes, well-formed shares can be easily linked to the server to which they were issued. For example, if Shamir secret sharing is implemented such that server i gets the secret share $(i, f(i))$ for some polynomial f, then this share, at least as is, trivially identifies the ith server. However, the corrupted servers could potentially pull their shares together and obfuscate them in a manner that prevents their trivial identification. Hence, what we want is a secret-sharing scheme with the following two guarantees. First, if Alice obtains the leaked information, she should be able to trace it back to the corrupted servers. Moreover, to hold them accountable she should be able to produce a proof that these servers indeed leaked her secret information. This property is called *traceability*. The scenario to consider here is this: Alice, or some law enforcement agency, obtains the leaked information (for example, by buying it from the corrupted servers who offer it for sale) and traces it back to the corrupted servers. They produce a proof implicating the corrupted servers, which Alice may use in court. In this sense, traceability does not aim to, and cannot, prevent the leaking of secret shares with absolute certainty. Rather, it aims to serve as a deterrent, discouraging parties from leaking secret shares. This is similar to the logic underpinning traitor tracing schemes [21].

This leads us to the second requirement of traceable secret sharing, called *non-imputability*. This property stipulates that Alice should not be able to generate a false proof, implicating an honest server who was not involved in any leak. In our example, this prevents Alice from collecting unlawful damages in court.

In their work, Goyal et al. [41] constructed a new secret-sharing scheme, that satisfies both traceability and non-imputability. In their scheme, however, traceability comes at the cost of efficiency. First, the shares in their schemes are very large; they are λ^2-bit long where λ is the security parameter and secrets lie in $\{0,1\}^\lambda$. That is, shares are quadratic in the secret size. This should be contrasted with other secret sharing schemes (e.g., Shamir) in which shares are the same size as the secret. Second, their tracing algorithm is quite involved and relies on Goldreich-Levin decoding [34]. These sources of inefficiency leave open the problem of constructing a *practical* traceable secret-sharing scheme, in which shares are small and tracing is efficient. Additionally, the scheme of Goyal et al. is a custom scheme, and is quite different from standard secret-sharing schemes, such as Shamir's [65] or Blakley's [5] schemes. It is thus an interesting question to trace misbehaving servers in widely-used secret sharing schemes, such as Shamir secret sharing.

1.1 Our results

In this work, we present very efficient constructions of traceable secret-sharing schemes based on the classic schemes of Shamir [65] and Blakley [5]. We believe that our constructions are an important step towards bringing traceable secret-sharing schemes to practice. In more detail, our contributions are threefold:

1. We give new security definitions for traceable secret sharing, strengthening the definitions of Goyal el al. [41], with one caveat on which we elaborate below.
2. We present a new and practical tracing procedure for standard Shamir secret sharing.
3. We put forth a variant of Blakley secret sharing, and present an optimally-efficient tracing procedure for it.

We now elaborate on each of these contributions.

A stronger security notion. We revisit the definition of traceable secret sharing put forth by Goyal, Song, and Srinivasan [41]. In their model, a subset $\mathcal{I} \subseteq [n]$ of servers may leak $f_{\mathcal{I}}(\mathsf{sh}_{\mathcal{I}})$

where $f_{\mathcal{I}}$ is some adversarially chosen function and $\mathsf{sh}_{\mathcal{I}}$ is the set of all shares held by servers in $\mathcal{I}$. There is also a reconstruction box R that takes in the leaked information $f_{\mathcal{I}_1}(\mathsf{sh}_{\mathcal{I}_1}), \dots, f_{\mathcal{I}_k}(\mathsf{sh}_{\mathcal{I}_k})$ from several disjoint subsets and, if $\left|\bigcup_{i\in[k]} \mathcal{I}_k\right| \geq t$, outputs the underlying secret s. For tracing, they require the existence of a tracing algorithm Trace that solves the following problem: given $f_{\mathcal{I}_1}(\mathsf{sh}_{\mathcal{I}_1}), \dots, f_{\mathcal{I}_k}(\mathsf{sh}_{\mathcal{I}_k})$, a description of the functions $f_{\mathcal{I}_1}, \dots, f_{\mathcal{I}_k}$, and access to R, the tracing algorithm should output the identity of at least one corrupted server along with a proof π that this server is indeed corrupted.

In Section 2 we present a definition which requires tracing with less information, making it more realistic in our view. Concretely, in our model, a subset $\mathcal{I}$ of $f < t$ colluding servers outputs a reconstruction box R that depends on the shares held by the servers in $\mathcal{I}$. This reconstruction box takes in as input additional $t - f$ shares $\mathsf{sh}'_1, \dots, \mathsf{sh}'_{t-f}$ of servers not in $\mathcal{I}$, and outputs the secret that is reconstructed from the shares provided as input and the shares of the servers in $\mathcal{I}$. Then, the definition requires that there is a tracing algorithm, that given *black-box access* to R, traces it back to $\mathcal{I}$. Indeed, giving the tracing algorithm the code of R seems unlikely to help, since the corrupted servers might apply some heuristic code cloaking to R before publishing it. In any case, our tracing algorithm will only require black-box access to R. Also observe, that if $|\mathcal{I}| \geq t$ then tracing R is impossible, since the servers could reconstruct the secret s^* and sell it directly. This perfectly hides their identity. This is why our focus is on the case where $f < t$.[1]

We believe that our definition has a couple of advantages:

- First, the corrupted servers may try to thwart detection by combining their shares and applying some unknown post-processing to them. Modeling this leak as a reconstruction box R captures *any* post-processing that maintains the functionality of the shares. Importantly, our definition does not require that the tracing algorithm learns this post-processing function in order to trace. This is in line with the realistic scenario described above, in which a subset $\mathcal{I}$ of corrupted servers decides to sell their shares in the form of the reconstruction box R. In the case Alice or a law enforcement agency obtain R to trace it back to $\mathcal{I}$, it seems unlikely that they will learn the exact manner by which R was constructed from the secret shares of $\mathcal{I}$.
- Second, our definition is stronger than that of Goyal et al. in the following sense: given the information required to trace in their model, one can trace in our model as well. One caveat to that is that our definition requires that R output the complete reconstructed secret, whereas Goyal et al. only require that R learns any non-trivial information about the secret. A thorough comparison of the two definitions is presented in Section 2, where we argue that tracing a complete reconstruction box is a natural model.

Tracing leaks in Shamir secret sharing. In Section 3 we present an efficient tracing procedure for tracing leaks in standard Shamir t-out-of-n secret sharing, where each servers's share is the evaluation of a degree $t-1$ polynomial on a point x that is drawn *at random* from a field $\mathbb{F}$ satisfying $|\mathbb{F}| \gg n$. We explain in Section 3 why choosing the x points in this manner is necessary. In particular, shares in our schemes consist of just two field elements – just twice as long as standard Shamir shares.

The basic idea behind our tracing algorithm is the following. Suppose for simplicity that R is a perfect reconstruction box that has $f = t - 1$ shares $\{(x_i, y_i)\}_{i\in[t-1]}$ hardcoded in it, and it

[1] In the model of Goyal et al. this restriction is captured by requiring that all colluding subsets $\mathcal{I}_1, \dots, \mathcal{I}_k$ are of size less than t.

takes one additional share (x^*, y^*) as input. By perfect, we mean that R always outputs the result of Lagrange interpolation at the point 0, applied to $\{(x_i, y_i)\}_{i\in[t-1]}$ and (x^*, y^*). We observe that the Lagrange coefficient of y^* in this interpolation can be treated as an inverse polynomial in x^* whose roots are exactly the x-values $x_1, \ldots, x_{t-1}$ of the shares hardcoded in R. Then by carefully querying R a total of $\Omega(t)$ times, we are able to obtain t evaluations of this polynomial. Hence, we can interpolate it and then find its roots, giving us the x values of the leaked shares used by R. Since all x values are chosen at random from a large field, this information identifies the corrupted servers with overwhelming probability. More generally, to trace $f < t$ corruptions, our tracing algorithm issues $\Omega(f)$ queries to the reconstruction box R.

When R is not perfect, and may err with some probability ϵ, we need to use list decoding for Reed-Solomon codes (for example, the Guruswami-Sudan algorithm [42]) to obtain a list of candidate polynomials, such that one of them is guaranteed to have $x_1, \ldots, x_{t-1}$ as its roots. For more details, see Section 3, where we also explain how to trace when the number f of corruptions is not known in advance, and how to make our scheme accommodate non-imputability.

Tracing leaks in Blakley secret sharing. In Section 4 we present a very efficient tracing algorithm for t-out-of-n Blakley secret sharing. We let the secret s live in some finite field $\mathbb{F}$. To share s, the dealer first randomly extends it to a point $\boldsymbol{x} = (s, x_2, \ldots, x_t)$ in $\mathbb{F}^t$, by appending $t-1$ random $\mathbb{F}$-elements to it. Then, the share of server i is a random hyperplane H_i in $\mathbb{F}^t$ that passes through $\boldsymbol{x}$. The intersection of any t such hyperplanes uniquely defines $\boldsymbol{x}$, and hence s, with overwhelming probability, whereas every $t-1$ hyperplanes are (almost) statistically independent of s. Suppose we have a perfect reconstruction box R that has f hyperplanes $H_{i_1}, \ldots, H_{i_f}$ hardcoded in it which it uses to reconstruct the secret. Moreover, suppose that instead of just the secret s, this box outputs the entire point $\boldsymbol{x}$ that is the intersection of $H_{i_1}, \ldots, H_{i_f}$ and the $t-f$ hyperplanes that it receives as input. In this case, tracing becomes easy. To trace R back to servers $i_1, \ldots, i_f$, we simply invoke it on $t-f$ random hyperplanes $H'_1, \ldots, H'_{t-f}$. It is not hard to see that the intersection point that R outputs is a random point $\boldsymbol{r}$ in $\mathbb{F}^t$, conditioned on it intersecting $H_{i_1}, \ldots, H_{i_f}$. If $\mathbb{F}$ is large, then $\boldsymbol{r}$ will almost surely not lie on any of the honest servers' hyperplanes, and will therefore identify the corrupted servers. Hence, with a single query to R we can identify the corrupted servers.

The problem is that R does not output the entire intersection point $\boldsymbol{r}$, but just its first coordinate, which is the reconstructed secret. This is not enough information for the above tracing procedure, as it does not uniquely identify the corrupted servers. To remedy this situation, in Section 4.1 we extend Blakley's scheme such that the secret is a full point in $\mathbb{F}^t$. Doing so while still retaining the secrecy of the secret sharing scheme turns out to be non-trivial. Then, in Section 4.2, we present a tracing algorithm that relies on the above idea for our extended scheme. This tracing algorithm still requires just one query to the reconstruction box R, and is hence optimal in this sense.[2]

In Section 6 we also discuss how the fact that we use just one query to R may be beneficial for tracing *stateful* reconstruction services (as opposed to just *stateless* reconstruction boxes). Both the scheme of Goyal et al. [41] and our Shamir-based schemes cannot be used to trace stateful reconstruction services.

We emphasize that our Blakley-inspired traceable secret sharing scheme is also very efficient in terms of its share size. In Section 4 we discuss how to represent shares in this scheme with just $\lambda + |s|$, where λ is the security parameter and $|s|$ is the secret size. This essentially matches

[2] If R may sometimes err, then our tracing algorithm requires just one *successful* query to R, which is again optimal.

our Shamir-based construction for secrets in $\{0,1\}^\lambda$. The reader is referred to Section 4 for our full-fledged scheme that satisfies non-imputability, and for additional technical details.

Leaker confirmation and accountability in threshold VUFs. In Section 5, we discuss an easier task of *confirming* that a given "suspected" subset $\mathcal{I}'$ of servers is indeed the corrupted subset behind a reconstruction box R. We then explain how the same techniques underlying our tracing procedure for Shamir secret sharing can be used to solve the confirmation problem as well. Though this is unsurprising in and of itself (confirmation is easier than tracing), we show how this confirmation mechanism can be performed in the exponent of a cyclic group. As we explain in Section 5, this means that our confirmation mechanism can be used to confirm the identity of leaking parties in BLS-based threshold verifiable unpredictable functions (VUFs) [49, 8, 47].

Future work. Our work opens up many avenues for future work, which we discuss at length in Section 6. One such interesting direction is to explore the connection between traceable secret sharing and erasure codes. Both Shamir's and Blakley's schemes can be seen as first randomly embedding the secret in a higher dimension space, and then deterministically encoding it using some erasure code [48]. A fascinating question that our work leaves open is to come up with tracing procedures for other secret-sharing schemes that can be thought of in the same manner. These include schemes that are based on CRT-encodings [3, 50, 35], and schemes that are obtained from low-density parity-check (LDPC) codes [22, 2].

Another compelling direction for future research is to come up with a traceable secret-sharing scheme in which reconstruction is linear and tracing requires only degree-2 operations. As we explain in Section 6, this may have exciting applications for *tracing* traitors in threshold VUFs (which is a stronger guarantee than traitor confirmation).

Finally, Goyal et al. [41] showed how the techniques underlying their traceable secret sharing construction can be applied to obtain traceablity for leaked information in schemes for computation delegation. However, the construction is not black-box from any traceable secret sharing scheme, and it would be interesting to apply our techniques to traceable computation delegation as well.

1.2 Additional Related Work

There are several cryptographic primitives that resemble the notion of traceable secret sharing. We briefly discuss them here. The first notion that comes to mind in that respect is that of traitor tracing for encryption schemes, first introduced by Chor, Fiat, and Naor [21]. In such schemes, a central authority issues a single encryption key, and n decryption keys, one per receiver. If any coalition of receivers comes together and produces a pirate decoder box D that can decrypt well-formed ciphertexts, then this box can be traced back to at least one of the receivers who contributed to it. Traitor tracing, therefore, also deals with tracing leaked secrets back to the parties who leaked them. The techniques used to construct traitor tracing schemes are, however, fundamentally different the techniques we use here. Many traitor tracing schemes have been proposed over the years (see, for example, [46, 53, 6, 29, 64, 43, 51, 27, 16, 32, 13, 38, 19, 71, 70, 37] and the references therein). Almost all of these build on either private linear broadcast encryption (PLBE) [11] or fingerprinting codes [12] (on which we will elaborate shortly). Tracing using these techniques is very different from the tracing algorithms we construct in this paper. It is an interesting open question whether inspiration can be drawn from the rich traitor tracing literature to construct new traceable secret-sharing schemes.

Several works have considered the setting in which the tracer in a traitor tracing scheme might be malicious, and the task of preventing it from falsely accusing an innocent receiver [58, 59, 60]. This is analogous to the notion of non-imputability in traceable secret sharing.

Boneh, Partap, and Rotem [10] recently generalized traitor tracing to the threshold decryption setting, in which t out of the n receivers are needed for decryption. They gave formal definitions and efficient constructions. Their constructions also rely on fingerprinting codes, and hence on fundamentally different techniques than our traceable secret-sharing constructions.

Fingerprinting codes, introduced by Boneh and Shaw [12] (see also [9, 69, 57, 7]), are used for fingerprinting digital content. Such codes are "traceable" in the following sense: if a new word is constructed from a subset of the codewords, by way of mixing and matching symbols from these codewords, then this new word can be traced back to at least one of the codewords that contributed to it. Very roughly speaking, our constructions for traceable secret sharing can also be seen as tracing symbols of a codeword back to a codeword, but in a very different sense. We discuss this point at length in Section 6.

Finally, Goyal [39] (see also [40]) has suggested the notion of Accountable-Authority Identity-Based Encryption, where a misbehaving key authority that leaks the secret key of a server can be proven malicious and held accountable.

1.3 Notation And Basic Definitions

In this section, we present the basic notions and cryptographic primitives that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a distribution X we denote by $x \leftarrow\!\!\$\ X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$, we denote by $x \leftarrow\!\!\$\ \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For a pair X, Y of distributions defined over the same domain Ω, we denote by $\mathsf{SD}(X, Y)$ the *statistical distance* between them, defined as $\mathsf{SD}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|$. In Section 3, we make use of the standard cryptographic notion of a one-way function, and the reader is referred to [33] for a formal definition. All other notation and existing algorithms or cryptographic primitive that we rely on will be defined in their respective sections.

2 Traceable Secret Sharing

In this section, we present our definition of traceable threshold secret-sharing schemes (we may sometime call them traceable secret-sharing schemes for short). We follow the definitions of Goyal, Song, and Srinivasan [41], with several key changes that we discuss at the end of the section.

2.1 Syntax and Correctness

A traceable secret-sharing scheme is a 4-tuple $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ of PPT algorithms. The algorithms Share and Rec are the standard secret sharing and secret reconstruction algorithms of threshold secret sharing schemes. We do restrict the presentation, however, to secret sharing schemes that are **symmetric**. By that, we mean that the secret sharing procedure works in a particular fashion; instead of a secret sharing algorithm that samples all n shares at once, we consider a Share algorithm that produces one share at a time. To produce n shares, Share is invoked n times over. Note, however, that to ensure correctness, many secret sharing schemes require that the n shares are correlated beyond just the underlying secret s. For example, in Shamir secret sharing, all shares must lie on the same degree $t-1$ polynomial whose free coefficient is s. To account for this added correlation, we consider secret sharing algorithms that take in a **correlation string** ρ as an additional input. In Shamir secret sharing, for example, this ρ specifies the other $t-1$ coefficients of the said polynomial. Correctness then needs to hold for every choice of ρ, and security is defined over a random choice of ρ.

We now define the syntax of traceable secret-sharing schemes in more detail:

- $\mathsf{Share}(1^\lambda, s, n, t, \rho) \rightarrow (\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i)$ is the sharing algorithm.[3] It takes as input the security parameter 1^λ, the secret s, the number n of parties, the threshold $t \leq n$, and a correlation string $\rho \in \{0,1\}^\kappa$, where $\kappa = \kappa(\lambda, n, t) \in \mathbb{N}$.[4] It outputs a share sh_i, a tracing key component tk_i and a verification key component vk_i.
- $\mathsf{Rec}(\{\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_t}\}) \rightarrow s$ is the deterministic secret reconstruction algorithm. It takes as input t secret shares $\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_t}$ and outputs a secret s. We stress that we do not assume that the reconstruction algorithm has external knowledge regarding the origin of the shares it receives as input, since the scheme is symmetric and all shares come from the same distribution. We write the input as $\{\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_t}\}$ to emphasize that Rec is oblivious to the ordering of the shares given to it, but to simplify notation, we may sometime write $\mathsf{Rec}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_t})$ instead.
- $\mathsf{Trace}^R(\mathsf{tk}, 1^{1/\delta}) \rightarrow (\mathcal{I}, \pi)$ is the randomized tracing algorithm. It takes as input a tracing key $\mathsf{tk} = (\mathsf{tk}_1, \ldots, \mathsf{tk}_n)$ and an error parameter δ, and it gets oracle (black-box) access to a reconstruction box R. The algorithm outputs a subset $\mathcal{I} \subseteq [n]$ of identities of leaking parties and an associated proof π.
- $\mathsf{Verify}(\mathsf{vk}, \mathcal{I}, \pi) \rightarrow \{0,1\}$ is the deterministic verification algorithm. It takes as input the verification key $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, an alleged traitor subset $\mathcal{I}$, and an associated proof π, and outputs either 1, implying acceptance of the proof that the parties in $\mathcal{I}$ are guilty, or 0, implying rejection.

Correctness. The correctness requirement for a traceable secret-sharing scheme is the standard correctness property of secret-sharing schemes. That is, any t-tuple of the shares should suffice to correctly reconstruct the secret.

Definition 1. *Let* $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ *be a traceable secret-sharing scheme and let* $\epsilon = \epsilon(\lambda) \in [0,1]$ *be a function of the security parameter* λ. *We say that* TTSS *is* ***ϵ-correct*** *if for every* $\lambda \in \mathbb{N}$, *every secret* s, *every* $n \in \mathbb{N}$, *every* $0 < t \leq n$, *every correlation string* $\rho \in \{0,1\}^\kappa$, *and every subset* $\mathcal{J} = \{i_1, \ldots, i_t\} \subseteq [n]$ *of size* t, *it holds that*

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

where $(\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i) \leftarrow\!\$\ \mathsf{Share}(1^\lambda, s, n, t, \rho)$ *for every* $i \in [n]$ *and* $s' := \mathsf{Rec}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_t})$.

[3] In the original definition of [41], the sharing process is done by running a generic two-party computation protocol between the dealer and each of the shareholders. We chose to abstract this process away. We elaborate on the differences between the definitions below.

[4] We allow κ to depend on n for generality, but in our constructions, κ will only be a function of λ and t. This allows for sampling of shares "on the fly", without knowing n in advance. Throughout the paper, κ will always refer to the bit-length of the correlation string, and when clear from context, we will not mention this explicitly.

2.2 Security

A traceable secret-sharing scheme should satisfy three security properties: secrecy, traceability, and non-imputability. The first is the standard secrecy property of secret-sharing schemes, which stipulates that any subset of less than t parties should learn nothing about the secret s. This is captured by the following definition.

Definition 2 (Statistical secrecy). *Let* $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ *be a traceable secret-sharing scheme and let* $\epsilon = \epsilon(\lambda) \in [0, 1]$ *be a function of the security parameter* λ*. We say that* TTSS *satisfies* ϵ***-secrecy*** *if for every* $\lambda \in \mathbb{N}$*, any two secrets* s, s'*, every* $n \in \mathbb{N}$*, every* $0 < t \leq n$*, and every subset* $\mathcal{J} = \{i_1, \ldots, i_{t-1}\} \subseteq [n]$ *of size* $t - 1$*, it holds that*

$$\mathsf{SD}\big((\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_{t-1}}), (\mathsf{sh}'_{i_1}, \ldots, \mathsf{sh}'_{i_{t-1}})\big) \leq \epsilon(\lambda)$$

where $\rho \leftarrow\!\$\ \{0,1\}^\kappa$*,* $(\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i) \leftarrow\!\$\ \mathsf{Share}(1^\lambda, s, n, t, \rho)$ *for every* $i \in [n]$*, and* $(\mathsf{sh}'_i, \mathsf{tk}'_i, \mathsf{vk}'_i) \leftarrow\!\$\ \mathsf{Share}(1^\lambda, s', n, t, \rho)$ *for every* $i \in [n]$*.*

Traceability. In addition, a traceable secret-sharing should provide *traceability.* Suppose a coalition $\mathcal{I} \subseteq [n]$ of parties, of size $f < t$, gets together and constructs a reconstruction box R using their shares. This R is an algorithm that takes in additional $t - f$ secret shares and outputs a secret s. Intuitively, if this R is a "good" reconstruction box, then it should be possible to trace it back to at least one of the parties who "contributed" its share to it. As we discuss in the introduction, tracing R back to the corrupted parties should be done given only black-box access to it.

In more detail, say that the secret shared among the parties is s^*, and the shares of all parties are $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$. Then, R is a good reconstruction box if for random $t - f$ additional shares for the secret s^*, denoted $\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}$, it holds that $R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f})$ outputs s^* with high probability. Definition 3 below formally defines good reconstruction boxes.

Definition 3 (Good reconstruction boxes). *Let* TTSS *be a traceable secret-sharing scheme. Let* $\lambda \in \mathbb{N}$*, let* $n, t, f \in \mathbb{N}$ *such that* $0 < f < t \leq n$*, and let* $\kappa = \kappa(\lambda, n, t)$*. For* $\epsilon \in [0, 1]$*, a secret* s^**, correlation string* $\rho \in \{0,1\}^\kappa$*, we say that a reconstruction box* R *is* $(n, t, f, s^*, \rho, \epsilon)$*-good if*

$$\Pr\left[R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = s^*\right] \geq \epsilon,$$

where the probability is taken over $(\mathsf{sh}'_i, \mathsf{tk}'_i, \mathsf{vk}_i) \leftarrow\!\$\ \mathsf{Share}(1^\lambda, s^*, n, t, \rho)$ *for* $i = 1, \ldots t - f$ *and the random coins of* R*.*

Equipped with this notion, the traceability experiment is presented in Fig. 1 and the notion of traceability for secret-sharing is defined below.

Definition 4 (Traceability). *Let* $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ *be a traceable secret-sharing scheme with secret space* $\mathcal{SCRT} = \{\mathcal{SCRT}\}_{\lambda \in \mathbb{N}}$*, and let* $\epsilon = \epsilon(\lambda)$ *and* $\delta = \delta(\lambda)$ *be functions of the security parameter. We say that* TTSS *is* ***traceable****, if for every probabilistic polynomial time adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{trac}}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) := \left| \epsilon \cdot \Pr\left[\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) = 1\right] - \frac{1}{|\mathcal{SCRT}_\lambda|} \right|.$$

Some remarks are in order:

```
Experiment $\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda)$
1 : $\mathcal{J} \leftarrow \emptyset$
2 : $(n, t, \mathcal{I}, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$   // $\mathcal{A}$ chooses the parties $\mathcal{I} = \{i_1, \dots, i_f\} \subseteq [n]$ to corrupt
3 : $s^* \leftarrow\!\!\$\, \mathcal{SCRT}_\lambda$
4 : $\rho \leftarrow\!\!\$\, \{0,1\}^\kappa$
5 : for $i = 1, \dots, n : (\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i) \leftarrow\!\!\$\, \mathsf{Share}(1^\lambda, s^*, n, t, \rho)$
6 : $\mathsf{tk} := (\mathsf{tk}_1, \dots, \mathsf{tk}_n),\ \mathsf{vk} := (\mathsf{vk}_1, \dots, \mathsf{vk}_n)$
7 : $R \leftarrow\!\!\$\, \mathcal{A}(\mathsf{state}, \mathsf{sh}_{i_1}, \dots, \mathsf{sh}_{i_f})$ where $\mathcal{I} = \{i_1, \dots, i_f\}$
8 : if $R$ is not $(n, t, f, s^*, \rho, \epsilon)$-good then return 0
9 : $(\mathcal{I}', \pi) \leftarrow\!\!\$\, \mathsf{Trace}^{R(\cdot)}(\mathsf{tk}, 1^{1/\delta(\lambda)})$
10 : if $\mathcal{I} = \mathcal{I}'$ AND $\mathsf{Verify}(\mathsf{vk}, \mathcal{I}', \pi) = 1$ then return 0, else return 1
```

Fig. 1. The tracing experiment for a traceable secret-sharing scheme TTSS and an adversary $\mathcal{A}$. The set $\mathcal{SCRT} = \{\mathcal{SCRT}_\lambda\}_{\lambda\in\mathbb{N}}$ denotes the secret space of TTSS.

1. **The advantage term:** The advantage is defined to be

$$\epsilon \cdot \Pr\left[\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) = 1\right] - 1/|\mathcal{SCRT}_\lambda|$$

(instead of simply $\Pr\left[\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) = 1\right]$) to discount for the following trivial attack, which succeeds in "breaking" traceability with probability essentially $1/(\epsilon \cdot |\mathcal{SCRT}_\lambda|)$. Fix a set $\mathcal{S} \subseteq \mathcal{SCRT}_\lambda$. The adversary then outputs a reconstruction box $R_{\mathcal{S}}$ that ignores the shares it receives as input, and outputs a uniformly random secret from $\mathcal{S}$, i.e., $s \leftarrow\!\!\$\, \mathcal{S}$. Observe that this reconstruction box is essentially untraceable, since it is independent of the shares of the corrupted parties. It is also ϵ-good for any secret $s \in \mathcal{S}$ for $\epsilon = 1/|\mathcal{S}|$. Moreover, the secret s^* chosen by the challenger in $\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda)$ is in $\mathcal{S}$ with probability

$$\alpha = \frac{|\mathcal{S}|}{|\mathcal{SCRT}_\lambda|} = \frac{1}{\epsilon \cdot |\mathcal{SCRT}_\lambda|}.$$

Note that if $|\mathcal{SCRT}_\lambda|$ is super-polynomial in λ, then ϵ or α (or both) must be negligible.
2. **Adaptive adversaries:** For simplicity, our definition requires that the adversary chooses the set of corrupted parties all at once, non-adaptively. However, observe that in symmetric secret sharing schemes, as we consider here, all shares come from the same distribution, and hence querying for shares adaptively would give no additional power to the adversary.
3. **The output of R:** Our definition requires that a good reconstruction R outputs the reconstructed secret s^*, and not some function thereof. For example, if R outputs $H(s^*)$ for some hash function H, our definition makes no guarantees as to the ability to trace R back to the corrupted subset. Looking ahead, our tracing algorithms will indeed rely on R outputting the secret s^* and not, say, $H(s^*)$. This might pose an issue if $H(s^*)$ is also very sensitive information; for instance, if $H(s^*)$ is used by Alice, the secret owner, as her Bitcoin secret key. We argue, however, that this can be justified. The secret owner is the one that gets to decide what the secret is. Therefore, Alice can set the secret s^* to be her secret key, and not a hash of it, in which case a reconstruction R that outputs $H(s^*)$ is of little use.

Non-imputability. Lastly, we require that a traceable secret-sharing scheme satisfy the notion of non-imputability, stating that honest parties cannot be wrongly blamed. In more detail, this means that even a malicious tracer cannot produce an accepting proof for the culpability of an honest party. This is formally captured by the security experiment in Fig. 2. There, the adversary gets the tracing and verification keys tk and vk, and may obtain the secret shares of any party of its choosing, as long as it is not the honest party it tries to blame.

Definition 5 (Non-imputability). *We say that a traceable secret-sharing scheme* $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ *satisfies **non-imputability**, if for every probabilistic polynomial time adversary $\mathcal{A}$, the following function is negligible in λ:*

$$\mathsf{Adv}^{\mathrm{ni}}_{\mathcal{A},\mathsf{TTSS}}(\lambda) := \Pr\left[\mathbf{ExpNI}_{\mathcal{A},\mathsf{TTSS}}(\lambda) = 1\right]$$

```
Experiment $\mathbf{ExpNI}_{\mathcal{A},\mathsf{TTSS}}(\lambda)$
1 : $\mathcal{J} \leftarrow \emptyset$
2 : $(n, t, i^*, s, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
3 : $\rho \leftarrow\!\!\$\ \{0,1\}^\kappa$
4 : for $i = 1, \ldots, n : (\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i) \leftarrow\!\!\$\ \mathsf{Share}(1^\lambda, s^*, n, t, \rho)$
5 : $\mathsf{tk} := (\mathsf{tk}_1, \ldots, \mathsf{tk}_n),\ \mathsf{vk} := (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$
6 : $(\mathcal{I}^*, \pi) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{state}, \{\mathsf{sh}_i\}_{i \in [n] \setminus \{i^*\}}, \mathsf{tk}, \mathsf{vk})$
        // output a subset $\mathcal{I}^*$ and a proof for its culpability
7 : if $i^* \in \mathcal{I}^*$ AND $\mathsf{Verify}(\mathsf{vk}, \mathcal{I}^*, \pi) = 1$ then return 1, else return 0
```

Fig. 2. The non-imputability experiment for a traceable secret-sharing scheme TTSS and an adversary $\mathcal{A}$.

Comparison with the definition of Goyal, Song, and Srinivasan [41]. As mentioned above, our definition takes after that of Goyal et al. [41]. There are, however, a few differences that make our definition incomparable to theirs. On the one hand, our definition gives the tracer much less information to work with, and hence provides stronger security guarantees in this sense. Concretely, in the work of Goyal et al. the leakage of secrets has more structure to it: each corrupted party i submits a function $f(\mathsf{sh}_i)$ of its secret share for an adversarially-chosen f.[5] The tracer then gets $\{f(\mathsf{sh}_i)\}_{i \in \mathcal{J}}$, where $\mathcal{J}$ is the set of corrupted parties of size at least t, together with a description of the function f. It also gets access to a reconstruction box R, that takes in at least t share encodings $\{f(\mathsf{sh}_i)\}$ as input. On its own, the reconstruction box R is independent of the secret shares. Observe that, indeed, given this information, it is easy to construct a reconstruction box R' in our model, with respect to any subset $\mathcal{I} \subset \mathcal{J}$ of corrupted parties. To do that, one simply hardcodes $\{f(\mathsf{sh}_i)\}_{i \in \mathcal{I}}$ and the function f into R'. Then, on input secret shares $(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-|\mathcal{I}|})$, the reconstruction box R', computes $R(\{f(\mathsf{sh}_i)\}_{i \in \mathcal{I}}, \{f(\mathsf{sh}'_j)\}_{j \in [t-|\mathcal{I}|]})$.

On the other hand, in our setting, to trace a reconstruction box R back to a corrupted party, we require that R fully recovers the underlying secret. Goyal et al., however, only require that the

[5] In the syntax of Goyal et al. each party i may submit a different, party-specific, function f_i of its share. In our setting, however, any information that pertains to the identity of the party who owns the share is considered part of the share. Hence, using just one function f for all shares is without loss of generality.

reconstruction box R is able to extract some non-trivial information regarding the secret in order to trace it back to a corrupted party (this is formalized by R accomplishing some distinguishing task; see [41] for the formal details). These two requirements are equivalent in the random oracle model, assuming that the user always hashes the secret with a random oracle prior to using it. This way, a reconstruction box that outputs partial information about the hash of the secret, must query the random oracle at the secret. By observing R's queries to the random oracle, one can extract the full secret s thereby obtaining a reconstruction box in our model. Even without the random oracle model, the user can apply a randomness extractor to the secret before using it, thus making a box R that only returns a few bits of information about the secret of no value to the adversary.

On a technical level, Goyal et al. consider a distributed and interactive secret-sharing protocol between the dealer and the n shareholders. The dealer's view of this interaction serves as both the tracing key tk and the verification key vk. In order to instantiate this sharing step, they use a generic protocol for secure 2-party computation (instantiated n times over, once between the dealer and each of the shareholders). To simplify the presentation, we chose to consider a static and centralized secret sharing algorithm Share, but we note that one can always transform it into a distributed protocol using generic multiparty computation as well.

2.3 A Useful Fact About Good Reconstruction Boxes

In the rest of this section, we introduce a different traceability notion, which we call *universal traceability.* This notion is easier to work with when proving traceability of traceable secret sharing schemes, and — as we will shortly see — any scheme that is universally traceable is also traceable in some formal sense. As we will argue, universal traceability also makes sense as a notion of traceability in its own right.

Looking ahead, universal traceability says that a *universally good* reconstruction box can be traced back to the subset of parties who manufactured it. Very informally, we call a reconstruction box universally good if it correctly reconstructs a random secret from a random sharing of it with high probability.

We begin by defining universally good reconstruction boxes. Our definition makes use of the following notation. For a subset $\mathcal{C} \subseteq \{0,1\}^\kappa$ of correlation strings, we denote by $\mathsf{sh} \leftarrow\!\!\$\ \mathcal{SH}_{\lambda,n,t}(\mathcal{C})$ the process of sampling a random share for a random secret, conditioned on the correlation string being in $\mathcal{C}$; that is, the process: $\rho \leftarrow\!\!\$\ \mathcal{C}$, $s \leftarrow\!\!\$\ \mathcal{SCRT}_\lambda$, $\mathsf{sh} \leftarrow\!\!\$\ \mathsf{Share}(1^\lambda, s, n, t, \rho)$. When λ, n and t are clear from context, we may write $\mathsf{sh} \leftarrow\!\!\$\ \mathcal{SH}(\mathcal{C})$.

Definition 6 (Universally-good reconstruction boxes). *Let* TTSS *be a traceable secret-sharing scheme. Let $\lambda \in \mathbb{N}$, let $n, t, f \in \mathbb{N}$ such that $0 < f < t \leq n$, and let $\kappa = \kappa(\lambda, n, t)$. For $\epsilon \in [0,1]$, f shares $\mathbf{sh} = (\mathsf{sh}_1, \ldots, \mathsf{sh}_f)$, and a subset $\mathcal{C} \subseteq \{0,1\}^\kappa$, we say that a reconstruction box R is $(n, t, \mathbf{sh}, \mathcal{C}, \epsilon)$-good if*

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

where the probability is taken over $\mathsf{sh}'_i \leftarrow\!\!\$\ \mathcal{SH}_{\lambda,n,t}(\mathcal{C})$ for $i = 1, \ldots t - f$ and the random coins of R.

We now use the notion of universally good reconstruction boxes to define universal traceability. Our definition makes use of the following conventions. Let $\mathit{\Gamma} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$ be a partition of the space $\{0,1\}^\kappa$ of correlation strings. For a correlation string $\rho \in \{0,1\}^\kappa$, we denote by $\mathit{\Gamma}(\rho)$ the unique subset $\mathcal{C}_i$ that contains ρ.

Using this notation, the universal traceability experiment is presented in Fig. 3. It is almost identical to the tracing security experiment from Fig. 1 other than the fact that the reconstruction box R is required to be universally good with respect to $\Gamma(\rho)$, where ρ is the correlation string used to generate the secret shares in the experiment (rather than just good with respect to the secret s^*). For simplicity, we only parameterize the experiment by ϵ, and not δ, by setting $\epsilon = \delta$.

```
Experiment $\mathbf{ExpUniTrace}_{\mathcal{A},\mathsf{TTSS},\Gamma,\epsilon}(\lambda)$
1 :  $\mathcal{J} \leftarrow \emptyset$
2 :  $(n, t, \mathcal{I}, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
3 :  $s^* \leftarrow\!\!\$\ \mathcal{SCRT}_\lambda$
4 :  $\rho \leftarrow\!\!\$\ \{0,1\}^\kappa$
5 :  for $i = 1, \dots, n : (\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i) \leftarrow\!\!\$\ \mathsf{Share}(1^\lambda, s^*, n, t, \rho)$
6 :  $\mathsf{tk} := (\mathsf{tk}_1, \dots, \mathsf{tk}_n),\ \mathsf{vk} := (\mathsf{vk}_1, \dots, \mathsf{vk}_n)$
7 :  $R \leftarrow\!\!\$\ \mathcal{A}(\mathsf{state}, \mathsf{sh}_{i_1}, \dots, \mathsf{sh}_{i_f})$ where $\mathcal{I} = (i_1, \dots, i_f)$
8 :  $\mathbf{sh} := (\mathsf{sh}_{i_1}, \dots, \mathsf{sh}_{i_f})$
9 :  if $R$ is not $(n, t, \mathbf{sh}, \Gamma(\rho), \epsilon)$-universally-good then return $0$
10 : $(\mathcal{I}', \pi) \leftarrow\!\!\$\ \mathsf{Trace}^{R(\cdot)}(\mathsf{tk}, 1^{1/\epsilon(\lambda)})$
11 : if $\mathcal{I} = \mathcal{I}'$ AND $\mathsf{Verify}(\mathsf{vk}, \mathcal{I}', \pi) = 1$ then return $0$, else return $1$
```

Fig. 3. The universal tracing experiment for a traceable secret-sharing scheme TTSS and an adversary $\mathcal{A}$. Changes from the tracing security experiment in Fig. 1 are marked in blue.

Definition 7 (Universal traceability). *Let* $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ *be a traceable secret-sharing scheme with secret space* $\mathcal{SCRT} = \{\mathcal{SCRT}_\lambda\}_{\lambda\in\mathbb{N}}$*, let* $\epsilon = \epsilon(\lambda)$ *be a function of the security parameter, and let* Γ *be a partition of the space* $\{0,1\}^\kappa$ *of correlation strings. We say that* TTSS *satisfies universal traceability, if for every probabilistic polynomial time adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\text{uni-trac}}_{\mathcal{A},\mathsf{TTSS},\Gamma,\epsilon}(\lambda) := \left| \Pr\left[\mathbf{ExpUniTrace}_{\mathcal{A},\mathsf{TTSS},\Gamma,\epsilon}(\lambda) = 1\right] - \frac{1}{|\mathcal{SCRT}_\lambda|} \right|.$$

If Γ is the trivial partition (that is, $\Gamma = \{\{0,1\}^\kappa\}$), we may omit it from the notation. Observe that the trivial reconstruction box R that outputs a uniformly random secret $s^* \leftarrow\!\!\$\ \mathcal{SCRT}_\lambda$, independently of its inputs, is ϵ-universally good for $\epsilon = 1/|\mathcal{SCRT}_\lambda|$ and is essentially untraceable. This is why we subtract a $1/|\mathcal{SCRT}_\lambda|$ term from the probability that the experiment outputs 1 when defining the advantage of $\mathcal{A}$ above.

We argue that Defintion 7 is a very reasonable one. Intuitively, the definition says that if R is good with respect to the equivalence class $\Gamma(\rho)$ of the *real correlation string* ρ (i.e., the one used to sample the real secret shares sampled by the challenger), then it should be traced back to the corrupted subset. Intuitively, since $\mathcal{A}$ only sees at most $t-1$ shares, and does not have any information about the real correlation string ρ, it is very reasonable to consider a reconstruction box R that almost always fails on $\Gamma(\rho)$ to be a bad reconstruction box. To make this intuition precise, we prove below that for a certain natural class of traceable secret sharing schemes (as the ones we will construct), if a traceable secret sharing scheme satisfies universal traceability (Definition 7), then it also satisfies standard traceability (Definition 4) with related parameters.

Looking ahead, we will use universal traceability with respect to two extremes of the partition Γ. In Section 3, we will use the trivial partition $\Gamma = \{\{0,1\}^\kappa\}$; meaning, a universally good reconstruction box should reconstruct random sharings of a random secret. In Section 4 we will use the complete partition into singletons $\Gamma = \{\{\rho\} \ : \ \rho \in \{0,1\}^\kappa\}$; meaning, a universally good reconstruction box should reconstruct random sharings (of random secrets), generated with the *real reconstruction string* ρ sampled by the challenger.

Bidirectional secret sharing schemes. To relate the notion of universal traceability to that of standard traceability, it will be convenient to define a subclass of threshold secret sharing schemes, which we call *bidirectional*. Let Γ be a partition of the space of correlation strings. In such schemes, sampling a secret s^* uniformly at random from the secret space, and then generating t shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_t$ of it conditioned on the correlation string being in some $\mathcal{C} \in \Gamma$, is statistically close to sampling t shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_t \leftarrow\!\!\$\ \mathcal{SH}(\mathcal{C})$, and then computing $s^* \leftarrow \mathsf{Rec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_t)$. Both of our constructions will satisfy this property with respect to the different partitions discussed above.

Definition 8 (Bidirectional secret sharing). *Let* TTSS *be a traceable secret-sharing scheme with secret space* $\mathcal{SCRT} = \{\mathcal{SCRT}_\lambda\}_{\lambda \in \mathbb{N}}$*, and let* $\epsilon = \epsilon(\lambda) \in [0,1]$*,* $n = n(\lambda)$ *and* $t = t(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$*. Let* Γ *be a partition of the space* $\{0,1\}^\kappa$ *of correlation strings for* TTSS*. We say that* TTSS *is* (Γ, ϵ)-bidirectional *if for every* $\lambda \in \mathbb{N}$ *and every* $\mathcal{C} \in \Gamma$*,*

$$\mathsf{SD}\left((s, \mathsf{sh}_1, \ldots, \mathsf{sh}_t), (s', \mathsf{sh}'_1, \ldots, \mathsf{sh}'_t)\right) \leq \epsilon(\lambda),$$

where:

1. $s \leftarrow\!\!\$\ \mathcal{SCRT}_\lambda$*,* $\rho \leftarrow\!\!\$\ \mathcal{C}$*, and* $(\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i) \leftarrow\!\!\$\ \mathsf{Share}(1^\lambda, s, n, t, \rho)$ *for* $i = 1, \ldots, t$*.*
2. $\mathsf{sh}'_i \leftarrow\!\!\$\ \mathcal{SH}_{\lambda,n,t}(\mathcal{C})$ *for* $i = 1, \ldots, t$ *and* $s' \leftarrow \mathsf{Rec}(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_t)$*.*

From universal traceability to traceability. As promised, we will now relate standard traceability (Definition 4) to universal traceability (Definition 7) for bidirectional schemes.

Lemma 1. *Let* $\mathsf{TTSS} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ *be a traceable secret-sharing scheme that satisfies* ν_1*-correctness and* (Γ, ν_2)*-bidirectionality for negligible functions* ν_1, ν_2 *of the security parameter* $\lambda \in \mathbb{N}$ *and a partition* Γ *of* $\{0,1\}^\kappa$*. Let* $\epsilon = \epsilon(\lambda)$ *and* $\delta = \delta(\lambda)$ *be functions of the security parameter such that* $\delta \leq \min\{2\epsilon, 1/2\}$*, and let* $\mathcal{A}$ *be an adversary. Assume that* $\mathsf{Adv}^{\mathrm{trac}}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) \geq 2\delta$ *and* $\Pr\left[\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) = 1\right] \geq \frac{1}{\epsilon|\mathcal{SCRT}_\lambda|}$*. Then, there exists a negligible function* $\nu = \nu(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$ *it holds that*

$$\mathsf{Adv}^{\mathrm{uni\text{-}trac}}_{\mathcal{A},\mathsf{TTSS},\Gamma,\delta'}(\lambda) \geq \mathsf{Adv}^{\mathrm{trac}}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda)/2 - \nu(\lambda)$$

where $\delta' \geq \delta - \nu(\lambda)$*.*

The proof of the lemma can be found in Appendix A. The crux of the argument is this: since the adversary sees at most $t-1$ shares, it has essentially no information about the secret s^*. Hence, if it outputs a reconstruction box that is ϵ-good for the real secret s^* with probability α, then it is good for $\alpha/2$ fraction of the secrets with probability $\alpha/2$. In particular, with probability $\alpha/2$, it is δ'-universally-good for $\delta' \approx \alpha/2$. This means that if α is non-negligible, then the adversary outputs a δ'-universally-good reconstruction box with non-negligible probability, and for a non-negligible δ'. In particular, setting $\alpha \geq 2\delta$ gives $\delta' \approx \delta$.

3 A Scheme Based on Shamir Secret Sharing

In this section, we present our traceable secret-sharing scheme based on Shamir secret sharing. Recall that Shamir secret sharing is defined over a finite field $\mathbb{F}$. To share a secret $s \in \mathbb{F}$ among n parties, with a threshold of t, the dealer samples a uniformly random polynomial q of degree $t-1$ over $\mathbb{F}$, conditioned on its free coefficient being s. Each party $i \in [n]$ is uniquely associated with some field element $x_i \in \mathbb{F}$, and its share is $\mathsf{sh}_i = (x_i, q(x_i)) \in \mathbb{F}$.[6] Any t evaluations of q can be used to reconstruct the polynomial, and hence s, e.g. via Lagrange interpolation. In contrast, any tuple of $t-1$ shares is uniformly random over $\mathbb{F}^{t-1}$, and hence reveals nothing about the secret.

We begin with an informal overview of our traceable Shamir secret sharing. Then, we will present a basic scheme that does not satisfy non-imputability, but already captures the main ideas behind our tracing procedure. Finally, we will show how to amend this scheme so that it also satisfies non-imputability.

3.1 Overview of our scheme

Random evaluation points. Typically in Shamir secret sharing, party i is deterministically associated with its evaluation point x_i. The simplest example is when the field $\mathbb{F}$ is $\mathbb{F}_p$, the finite field of integers modulo p. Then, a natural choice is $x_i = i$, when i is interpreted as an $\mathbb{F}_p$ element. Our first observation is that this approach cannot admit efficient tracing according to our definition. The reason is that an adversary may corrupt a random subset $\mathcal{I}$ of the parties and construct a reconstruction box R with the secret shares of $\mathcal{I}$ embedded in it. Let $f = |\mathcal{I}|$. Then, R will output the secret s if it gets exactly $t-f$ shares associated with parties outside of $\mathcal{I}$; the box R can link a share $(x_i, q(x_i))$ given to it as input to party i, because x_i is a deterministic function of the party's index i. Whenever R is given more than $t-f$ inputs, or if one of its input shares "belongs" to a party in $\mathcal{I}$, the box R outputs $\perp$ and refuses to output anything meaningful. In this scenario, regardless of the tracing strategy, R outputs anything other than $\perp$ with probability at most $\approx Q\binom{n-(t-f)}{f}/\binom{n}{f}$, where Q is the number of queries that Trace makes to R.[7] If, for example, $t = c_1 n$ and $f = c_2 t$ for constants $c_1, c_2 \in (0,1)$ and Q is polynomial in n, making R output anything other than $\perp$ would require $2^{\Omega(n)}$ queries in expectation.

To avoid this problem, we consider a variant of Shamir secret sharing in which each x_i is sampled uniformly at random from the field $\mathbb{F}$. If the field is large enough, this has a very small impact on the correctness of the scheme. Rejection sampling may also be used to reduce, or even eliminate, the small correctness error that results from the possibility of assigning the same x to two different parties. The benefit is that now a share $(x_i, q(x_i))$ cannot be linked to party i without knowing the dealer's randomness. As we will now see, this allows us to overcome the impossibility argument sketched above.

[6] If x_i is a deterministic function of some identity information associated with the party, then x_i need not be explicitly included as part of the share. Looking ahead, we will choose the x_is randomly, and hence they will need to be included in the share.

[7] To see why that is, consider a different experiment, in which the tracer always gets $\perp$ in response from R, but we still say that it wins if it ever queries R at a subset that does not intersect the corrupted subset $\mathcal{I}$. The probability that the tracer wins in this modified experiment is the same as in the original experiment since its view is identical in both experiments as long as it has not yet won. In this modified experiment, the queries of the tracer are independent of the corrupted subset $\mathcal{I}$, and so we can think of $\mathcal{I}$ as being chosen uniformly at random after the queries are determined. In this case, the probability that $\mathcal{I}$ does not intersect a certain subset of size $t-f$ is $\binom{n-(t-f)}{f}/\binom{n}{f}$, and the overall probability that the tracer wins is bounded by $Q \cdot \binom{n-(t-f)}{f}/\binom{n}{f}$.

Tracing via polynomial interpolation and factorization. To understand the basic idea behind our tracing procedure, it is instructive to consider the task that a pirate reconstruction box R for Shamir secret sharing needs to accomplish. For simplicity, assume that the number of corruptions is $f = t - 1$. In this case, R gets an additional share as input and outputs the secret s. This secret is the result of reconstructing the value $q(0)$ using the $t-1$ evaluations of q hardcoded in R and the additional evaluation given to it as input. Suppose that the shares hardocded in R are $(x_i, y_i = q(x_i))$ for $i = 1, \dots, t-1$. When feeding it an additional share of the form (x_t, y_t), R needs to output

$$s = \sum_{i \in [t]} \left(\prod_{k \in [t] \setminus \{i\}} \frac{x_k}{x_k - x_i} \right) \cdot y_i.$$

Otherwise, it is not a good reconstruction box. This can be re-written as

$$s = \sum_{i \in [t-1]} \left(\prod_{k \in [t] \setminus \{i\}} \frac{x_k}{x_k - x_i} \right) \cdot y_i + \left(\prod_{k \in [t-1]} \frac{x_k}{x_k - x_t} \right) \cdot y_t. \tag{1}$$

If we now feed R with the share $(x_t, y_t + 1)$, where x_t and y_t are as before, we obtain some s' satisfying

$$s' = \sum_{i \in [t-1]} \left(\prod_{k \in [t] \setminus \{i\}} \frac{x_k}{x_k - x_i} \right) \cdot y_i + \left(\prod_{k \in [t-1]} \frac{x_k}{x_k - x_t} \right) \cdot (y_t + 1). \tag{2}$$

Subtracting Eq. (1) from Eq. (2) and rearranging, we obtain

$$\prod_{k \in [t-1]} \frac{x_k - x_t}{x_k} = (s' - s)^{-1}. \tag{3}$$

We now consider the univariate polynomial $h(X) = \prod_{k \in [t-1]} \frac{x_k - X}{x_k}$ in the indeterminate X. Observe that the roots of the polynomial h are exactly the x_i values of the corrupted parties. Moreover, we can interpret Eq. (3) as an evaluation of h at the point x_t we fed to R. Repeating the above with additional $t-1$ fresh x_t values, would give us t evaluations of h. Since h is of degree $t-1$, this is enough to interpolate h and factor it to find its roots via polynomial factorization [4, 62, 45]. If the true x_i values of all the parties are given to the tracer as part of tk, the tracer can now trace the roots of h back to the corrupted parties.

Sharing and reconstruction. In what follows, we formally describe the Share and Rec algorithms for our scheme, which we denote TS (for "Traceable Shamir"). This is still a simplified version of our scheme that does not provide non-imputability, which we will handle later in this section. The scheme is parameterized by a finite field $\mathbb{F} = \mathbb{F}(\lambda, n, t)$.

We begin with the sharing algorithm. The algorithm receives the security parameter 1^λ, the secret $s \in \mathbb{F}$, integers n and t, and a correlation string ρ that is interpreted as $t-1$ coefficients in $\mathbb{F}$.

```
Share(1^λ, s, n, t, ρ) :

1. Parse $\rho$ as $(a_1, \dots, a_{t-1}) \in \mathbb{F}^{t-1}$.
   // observe that $q(X) = a_{t-1}X^{t-1} + \cdots + a_1 X + s$ is a polynomial of degree $t-1$ such that $q(0) = s$
2. For $i = 1, \dots, n$:
   (a) Sample $x_i \leftarrow\$ \mathbb{F}$.
```

```
    (b) Compute $y_i \leftarrow s + \sum_{j=1}^{t-1} a_j x_i^j$.
        // observe that $y_i = q(x_i)$
    (c) Set $\mathsf{sh}_i \leftarrow (x_i, y_i)$, $\mathsf{tk}_i \leftarrow x_i$ and $\mathsf{vk}_i \leftarrow x_i$.
3. Output $(\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i)$.
```

The reconstruction algorithm is the standard Shamir reconstruction algorithm, taking into account the random choices of the x_is. For simplicity of presentation, we present with shares numbered $1, \dots, t$, and we assume that $x_i \neq x_j$ for every $1 \leq i < j \leq t$.

```
$\mathsf{Rec}(\mathsf{sh}_1, \dots, \mathsf{sh}_t)$:
1. Parse $\mathsf{sh}_i$ as $(x_i, y_i)$ for $i = 1, \dots, t$.
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. Output $s$.
```

3.2 Tracing Imperfect Reconstruction Boxes

We now describe our tracing algorithm. Observe, that the informal overview from the previous section inherently assumed that the reconstruction box R is always correct, and in particular, that all evaluations of the polynomial $h(X)$ are correct. To work for imperfect reconstruction boxes, that output the correct reconstructed secret only with a certain non-negligible probability, we need an additional idea. This is because standard polynomial interpolation with erroneous evaluations might fail or give us the wrong polynomial. To resolve this issue, we observe that this problem of interpolating a polynomial of bounded degree from a set of evaluation points with errors is equivalent to the list decoding problem for Reed Solomon codes [67, 42]. Specifically, given a list of evaluations $\{(x'_j, z_j)\}$, 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. We formally define list decoding of Reed-Solomon codes in Definition 9.

Definition 9 (The RS list decoding problem). *Let $\mathbb{F}$ be a finite field, and let $k, N, C \in \mathbb{N}$ such that $C \leq N < |\mathbb{F}|$. The list decoding problem is defined as follows. Given k, C and N pairs of elements $\{x_i, y_i\}_{i \in [N]} \subseteq \mathbb{F}^2$, output a list H of all univariate polynomials of degree at most k that agree with at least C pairs $\{x_i, y_i\}$. Specifically,*

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

Our scheme uses list decoding for Reed-Solomon codes in a black-box way. For concreteness, we consider the seminal Guruswami-Sudan list decoding algorithm [42] in Theorem 1, but any algorithm that solves the Reed-Solomon list decoding problem defined in Definition 9 in polynomial time may be used instead.

Theorem 1 (Guruswami-Sudan [42]). *Let $\mathbb{F}$ be a finite field, and let $k, N, C \in \mathbb{N}$ such that $C \leq N < |\mathbb{F}|$ and $C \geq \sqrt{kN}$. 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 $\log |\mathbb{F}|$.*

By the fact that the decoding algorithm runs in polynomial time, the list H that it outputs is also of polynomial length (see also [36]). Denote by $\tau(N, k, \log|\mathbb{F}|)$ the polynomial bounding the list length.[8]

The tracing algorithm. Our full-fledged tracing algorithm obtains a list of evaluations of the polynomial $h(X)$ as described in the previous section. But then, instead of exact polynomial interpolation, it runs a list decoding algorithm to get a list of candidate polynomials. We factor each polynomial in the list one by one. For each polynomial, we check if all its roots belong to the list of true x_i values of all parties. With high probability, only one of the polynomials in the list will satisfy this condition. The roots of *this* polynomial will hence identify our traitors. We formally describe the tracing algorithm in Figure 4. It is parameterized by N and C (the parameters for Guruswami-Sudan decoding), and we will discuss how to set them later in this section.

```
Trace^R(tk, f, 1^{1/ε}):
1. Parse $\mathsf{tk}$ as $x_1, \ldots, x_n$.
2. For $\ell = 1, \ldots, N$:
   (a) For $i = f+1, \ldots, t-1$: sample $x_{\ell,i}, y_{\ell,i} \leftarrow\$ \mathbb{F}$ and set $\mathsf{sh}_{\ell,i} \leftarrow (x_{\ell,i}, y_{\ell,i})$.
   (b) Sample $x'_\ell, y'_\ell, \delta_\ell \leftarrow\$ \mathbb{F}$. Let $\mathsf{sh}'_\ell \leftarrow (x'_\ell, y'_\ell)$ and $\mathsf{sh}''_\ell \leftarrow (x'_\ell, y'_\ell + \delta_\ell)$.
   (c) Query $R$ on $(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}'_\ell)$ and on $(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}''_\ell)$. Let $s_\ell$ and $s'_\ell$ be $R$'s responses, respectively.
   (d) 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, let $z_\ell \leftarrow \frac{\delta_\ell}{s'_\ell - s_\ell} \cdot \prod_{i=f+1}^{t-1} \frac{x_{\ell,i}}{x_{\ell,i} - x'_\ell}$.
3. Let $L = \{x'_j, z_j\}_{j \in [N]}$. Run the list decoding algorithm $\mathcal{D}_{\mathsf{GS}}$ on the following inputs: the degree bound $f$, the integer $C$ and the list $L$ of pairs in $\mathbb{F}^2$. $\mathcal{D}_{\mathsf{GS}}$ returns a list of all candidate polynomials $H = \{h_i\}$ of degree at most $f$ that agree with at least $C$ evaluation points out of $L$.
4. For each $h_i(x) \in H$, find all roots $w_{i,1}, \ldots, w_{i,f}$ of $h_i$. If, for any $j \in [f]$, there is no $k \in [n]$ such that $w_{i,j} = x_k$, then remove $h_i$ from the list $H$. If $H$ is empty, terminate and output $\perp$.
5. Let $h_{j^*}$ be the first polynomial in $H$, let $\mathcal{I}$ be the set of indices $i \in [n]$ for which $x_i = w_{j^*,k}$ for some $k \in [f]$. Set $\pi \leftarrow \{x_i\}_{i \in \mathcal{I}}$.
6. Output $\mathcal{I}$ and $\pi$.
```

Fig. 4. The tracing algorithm for the Traceable Shamir Secret Sharing scheme TS.

Theorem 2. *For every adversary $\mathcal{A}$, for every $\lambda \in \mathbb{N}$, for every $N, C \in \mathbb{N}$ and $\epsilon \in \left[\sqrt{\frac{2(N+f+t^2)}{p}}, 1\right]$, such that $\epsilon > \sqrt{2C/N}$ and $\sqrt{fN} \le C \le N < p$, it holds that*

$$\mathsf{Adv}^{\text{uni-trac}}_{\mathcal{A},\mathsf{TS},\epsilon}(\lambda) \le e^{-\frac{\epsilon^2 N}{2}\cdot\left(1-\frac{1}{r}\right)^2} + \frac{f \cdot n \cdot \tau}{p}$$

where $p = |\mathbb{F}|$, $r = \frac{\epsilon^2 N}{2C}$, $n = n(\lambda)$ and $f = f(\lambda)$ are upper bounds on the number of parties and corruptions, respectively, and $\tau = \tau(N, f, \log p)$ is the polynomial upper bounding the length of the output of the Guruswami-Sudan algorithm.

[8] To solve the list decoding problem with probability 1, the Guruswami-Sudan algorithm runs in expected polynomial time. If we insist that it runs in strict polynomial time, this incurs a negligible error probability. To avoid overcluttering notation, we will ignore this negligible error in our analysis.

The parameters N and C can be set so that the advantage of the adversary is exponentially small, while meeting the list decoding constraints. Specifically for the Guruswami-Sudan algorithm, C can be set to be $\lceil \sqrt{Nf} \rceil$ and N can be set to $\lceil 16f\lambda/\epsilon^4 \rceil$, where λ is the security paramter. Observe that in this case $r \geq 2$, and hence the term $e^{-\frac{\epsilon^2 N}{2}\cdot(1-1/r)^2}$ is at most $e^{-\frac{\epsilon^2 N}{8}}$. Moreover, since $N \geq 8\lambda/\epsilon^2$, we get that the advantage of the adversary is bounded by $e^{-\lambda} + f \cdot n \cdot \tau/p$, which is negligible in λ whenever p is super-polynomial in λ.

Proof (of Theorem 2). Let $\lambda \in \mathbb{N}$, let $\epsilon \in [\sqrt{2(N+f+t^2)/p}, 1]$, and let $\mathcal{A}$ be an adversary taking part in the $\mathbf{ExpUniTrace}_{\mathcal{A},\mathsf{TS},\epsilon}(\lambda)$ security experiment. Let G denote the event in which $\mathcal{A}$ outputs a reconstruction box R that is $(n, t, \mathsf{sh}, \epsilon)$-good, where n and t are chosen by $\mathcal{A}$ at the beginning of the experiment, and $\mathsf{sh} = (\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f})$ are the shares given to it by the challenger. Conditioned on $\neg\mathsf{G}$, the output of the experiment is 0 with probability 1, and 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. $\{x_j^*\}_{j\in[|\mathcal{I}|]} = \{x_i\}_{i\in\mathcal{I}}$.

For $\ell = 1, \ldots, N$, let $\mathsf{E}_{\delta,\ell}$ be the event $\delta_\ell = 0$. For $\ell \in [N]$ and $j \in \{f+1, \ldots, t-1\}$, let $\mathsf{E}_{x,j,\ell,1}$ be the event that $x_{\ell,j} = x'_\ell$, let $\mathsf{E}_{x,j,\ell,\mathcal{I}}$ be the event that $x_{\ell,j} = x_k^*$ for some $k \in [f]$ and let $\mathsf{E}_{x,j,\ell,2}$ be the event that $x_{\ell,j} = x_{\ell,i}$ for some $i \in \{f+1, \ldots, j-1\}$. For simplicity of notation, let us use $\mathsf{E}_{x,j,\ell}$ to denote the event $\mathsf{E}_{x,j,\ell,1} \vee \mathsf{E}_{x,j,\ell,\mathcal{I}} \vee \mathsf{E}_{x,j,\ell,2}$ for all $j \in \{f+1, \ldots, t-1\}$ and $\ell \in [N]$. Let E_ℓ be the event that $x'_\ell = x'_j$ for some $0 < j < \ell$ and let $\mathsf{E}_{\ell,\mathcal{I}}$ be the event that $x'_\ell = x_j^*$ for some $j \in [f]$. Lastly, let $\mathsf{E}_{1,\ell}$ and $\mathsf{E}_{2,\ell}$ be the events that $R(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}'_\ell) = \mathsf{Rec}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f}, \mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}'_\ell)$ and $R(\mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}''_\ell) = \mathsf{Rec}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f}, \mathsf{sh}_{\ell,f+1}, \ldots, \mathsf{sh}_{\ell,t-1}, \mathsf{sh}''_\ell)$ respectively.

Let $\mathcal{J}^*$ denote the set $\{x_j^*\}_{j\in[f]}$, let $\mathcal{J}_\ell = \{x_{\ell,j}\}_{j\in[f+1,\ldots,t-1]}$ and let $\mathcal{J}_\ell^*$ denote the set $\mathcal{J}^* \cup \mathcal{J}_\ell \cup \{x'_\ell\}$. Then, in the event $\neg\mathsf{E}_{\delta,\ell} \bigwedge_{j\in\{f+1,\ldots,t-1\}}(\neg\mathsf{E}_{x,j,\ell}) \wedge \neg\mathsf{E}_\ell \wedge \neg\mathsf{E}_{\ell,\mathcal{I}} \wedge \mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}$, we have that $|\mathcal{J}_\ell^*| = t$ and

$$\begin{aligned} s_\ell = \textstyle\sum_{x_j^*\in\mathcal{J}^*} \left(\prod_{x\in\mathcal{J}_\ell^*\setminus\{x_j^*\}} \frac{x}{x-x_j^*}\right) y_j^* + \sum_{x_{\ell,j}\in\mathcal{J}_\ell} \left(\prod_{x\in\mathcal{J}_\ell^*\setminus\{x_{\ell,j}\}} \frac{x}{x-x_{\ell,j}}\right) y_{\ell,j} \\ + \left(\textstyle\prod_{x\in\mathcal{J}_\ell^*\setminus\{x'_\ell\}} \frac{x}{x-x'_\ell}\right) y'_\ell \end{aligned}$$

and

$$\begin{aligned} s'_\ell = \textstyle\sum_{x_j^*\in\mathcal{J}^*} \left(\prod_{x\in\mathcal{J}_\ell^*\setminus\{x_j^*\}} \frac{x}{x-x_j^*}\right) y_j^* + \sum_{x_{\ell,j}\in\mathcal{J}_\ell} \left(\prod_{x\in\mathcal{J}_\ell^*\setminus\{x_{\ell,j}\}} \frac{x}{x-x_{\ell,j}}\right) y_{\ell,j} \\ + \left(\textstyle\prod_{x\in\mathcal{J}_\ell^*\setminus\{x'_\ell\}} \frac{x}{x-x'_\ell}\right) (y'_\ell + \delta_\ell) \end{aligned}$$

This gives us the following:

$$s'_\ell - s_\ell = \delta_\ell \cdot \prod_{x_{\ell,j}\in\mathcal{J}_\ell} \frac{x_{\ell,j}}{x_{\ell,j} - x'_\ell} \cdot \prod_{x_j^*\in\mathcal{J}^*} \frac{x_j^*}{x_j^* - x'_\ell}.$$

Let us define a polynomial $h^*(X) = \prod_{x_j^*\in\mathcal{J}^*} \frac{x_j^*-X}{x_j^*}$. Then, the above equation implies that z_ℓ, as computed by Trace in the ℓth iteration, is the correct evaluation of $h^*(X)$ at $X = x'_\ell$. Hence, for each $\ell \in [N]$, we get a correct evaluation of $h^*(X)$ at a unique point x'_ℓ, in the event $\neg\mathsf{E}_{\delta,\ell} \bigwedge_{j\in\{f+1,\ldots,t-1\}} \neg\mathsf{E}_{x,j,\ell} \wedge \neg\mathsf{E}_\ell \wedge \neg\mathsf{E}_{\ell,\mathcal{I}} \wedge \mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}$. We will now bound the probability of this event.

We have that $\Pr[\mathsf{E}_{\delta,\ell}] = 1/p$ for all $\ell \in [N]$, since δ_ℓ is uniformly randomly sampled from $\mathbb{F}$ for each ℓ. Next, since $x_{\ell,j}$ is sampled uniformly at random from $\mathbb{F}$, we have that $\Pr[\mathsf{E}_{x,j,\ell,1}] = 1/p$, $\Pr[\mathsf{E}_{x,j,\ell,\mathcal{I}}] \leq f/p$ and $\Pr[\mathsf{E}_{x,j,\ell,2}] \leq (t-f-1)/p$ for all $\ell \in [N]$ and all $j \in \{f+1, \dots, t-1\}$. Hence, we have that $\Pr[\mathsf{E}_{x,j,\ell}] \leq t/p$. Similarly, $\Pr[\mathsf{E}_\ell \vee \mathsf{E}_{\ell,\mathcal{I}}] \leq (N+f)/p$ for all $\ell \in [N]$. Lastly, since R is $(n, t, \textsf{\textbf{sh}}, \epsilon)$-good, we have that, $\Pr[\mathsf{E}_{1,\ell}]$ and $\Pr[\mathsf{E}_{2,\ell}]$ are both at least ϵ. Let us denote the tuple of random variables $(\{(x_{\ell,j}, y_{\ell,j})\}_{j\in[f+1,t-1]}, x'_\ell)$ as $W \in \mathbb{F}^{2(t-f)-1}$. Then,

$$\begin{aligned}
\Pr[\mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}] &= \Sigma_{w\in\mathbb{F}^{2(t-f)-1}} \Pr[W = w] \cdot \Pr_{y'_\ell,\delta_\ell}[\mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell} | W = w] \\
&= \Sigma_w \Pr[W = w] \cdot \Pr_{y'_\ell}[\mathsf{E}_{1,\ell} | W = w] \cdot \Pr_{y'_\ell,\delta_\ell}[\mathsf{E}_{2,\ell} | W = w] \qquad (4) \\
&= \mathbb{E}_w \left[\Pr_{y'_\ell}[R(w, y'_\ell) = \mathsf{Rec}(\textsf{\textbf{sh}}, w, y'_\ell)]^2 \right] \\
&\geq \left(\mathbb{E}_w \left[\Pr_{y'_\ell}[R(w, y'_\ell) = \mathsf{Rec}(\textsf{\textbf{sh}}, w, y'_\ell)] \right] \right)^2 \qquad (5) \\
&= \left(\Pr_{w,y'_\ell}[R(w, y'_\ell) = \mathsf{Rec}(\textsf{\textbf{sh}}, w, y'_\ell)] \right)^2 \\
&\geq \epsilon^2
\end{aligned}$$

Eq. (4) follows from the fact that the events $\mathsf{E}_{1,\ell}$ and $\mathsf{E}_{2,\ell}$ are independent once they are conditioned on the value of W. Eq. (5) follows from Jensen's inequality.

By combining the above and applying the union bound, we get:

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

The last equation follows from our assumption that $\epsilon^2 \geq 2(N+f+t^2)/p$.

Let us define an indicator random variable $Z_\ell = \mathbb{1}[\neg\mathsf{E}_{\delta,\ell} \bigwedge_{j\in[t-1]\setminus[f]} \neg\mathsf{E}_{x,j,\ell} \wedge \neg\mathsf{E}_\ell \wedge \neg\mathsf{E}_{\ell,\mathcal{I}} \wedge \mathsf{E}_{1,\ell} \wedge \mathsf{E}_{2,\ell}]$, which is 1 if and only if we get a correct evaluation of $h^*(X)$ at a unique point x'_ℓ. Since all the shares are sampled independently for each $\ell \in [N]$, we get that $\Pr[Z_\ell = 1] = E[Z_\ell] \geq \epsilon^2/2$.

Let $Z = \Sigma_{\ell\in[N]} Z_\ell$. By the Chernoff bound, we have that for every $\eta > 0$,

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

Since $N = (2 \cdot r \cdot C)/\epsilon^2$, we can set $\eta = 1 - 1/r$, to get that $Z > C$ with probability at least $1 - e^{-\frac{\epsilon^2 N}{2} \cdot \left(1-\frac{1}{r}\right)^2}$. Hence, with this probability, the list decoding algorithm $\mathcal{D}_{\mathsf{GS}}$ will be able to output all polynomials that agree with at least C evaluations out of $\{x'_\ell, z_\ell\}_{\ell\in[N]}$, including the polynomial $h^*(X)$. Next, since all the roots of $h^*(X)$ correspond to x_i values of the parties in $\mathcal{I}$, h^* will not be eliminated from H in Step 4.

Additionally, we claim that with high probability, there will be no other polynomial in the list H after Step 4. To prove this, let us first define an event $\mathsf{E}_{h,i}$ for all honest parties $i \in [n] \setminus \mathcal{I}$, denoting that x_i is a root of some polynomial in the list H. Next, observe that the shares of honest parties, i.e. $\{x_i, y_i\}_{i\notin\mathcal{I}}$ are statistically independent from both the view of $\mathcal{A}$ and the shares sampled by the Trace algorithm. Hence, the probability that x_i is a root of any polynomial h_j in H is bounded by f/p, since the degree of h_j is bounded by f for all $h_j \in H$. Since H has upto τ polynomials, we get that $\Pr[\mathsf{E}_{h,i}] \leq f \cdot \tau/p$.

Then, by applying a union bound over all honest x_i, we get that

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

Lastly, observe that $\mathcal{A}$ can win the game only if (a) $Z < C$ so that the list decoding algorithm fails to find h^* or (b) if any of the events $\mathsf{E}_{h,i}$ occur, causing an honest party to be blamed. So we get that,

$$\mathsf{Adv}^{\text{uni-trac}}_{\mathcal{A},\mathsf{TS},\epsilon}(\lambda) \leq e^{-\frac{\epsilon^2 N}{2}\cdot\left(1-\frac{1}{r}\right)^2} + \frac{f\cdot(n-f)\cdot\tau}{p}$$

This proves the theorem. □

Learning f. If the number f of corruptions is not known in advance by the tracing algorithm, it can learn it by simply trying $f = t-1, t-2, \ldots$ until it reaches a value of f that works; that is, a value of f for which the above tracing algorithm indeed finds exactly f corrupted x_i values. Suppose that the real number of corruptions is f^*. For each value $f > f^*$ that Trace tries, outputting a subset $\mathcal{I}$ of size f means outputting at least one honest party. By the analysis in the proof of Theorem 2, the probability that it outputs such a subset is at most $f\cdot(n-f)\cdot\tau/p \leq n^2\tau/p$. Moreover, when Trace tries $f = f^*$, then Theorem 2 tells us that it will fail to output the correct subset with probability at most $e^{-\lambda} + n^2\tau/p$ (for the choices of parameters discussed above). Hence, by a union bound, the probability that Trace correctly traces R back the corrupted subset is at least $1-(e^{-\lambda} + 2n^3\tau/p)$.

3.3 Adding Non-Imputability

We now present the changes needed to make our Traceable Shamir scheme TS satisfy non-imputability, yielding our full-fledged traceable secret sharing scheme based on Shamir secret sharing. In the simplified scheme presented in the previous sections, the fact that tk contains $x_1, \ldots, x_n$ explicitly, allows a malicious tracer to falsely accuse any party i by including their x_i as part of the proof π. To remedy this situation, we need to a-priori hide the x_is, but in a way that still allows the tracer (and, later on, the verifier) to link an x_i value, once it has been extracted from a reconstruction box R, back to party i. Thus, instead of including $x_1, \ldots, x_n$ in the clear in tk and vk, we include $F(x_1), \ldots, F(x_n)$ where F is a one-way function. The proof would still consist of the x_is of the corrupted parties. This way, intuitively speaking, falsely accusing an honest party i amounts to inverting $F(x_i)$ for a randomly chosen x_i.

Specifically, the changes to our scheme are as follows:

1. The share algorithm $\mathsf{Share}(1^\lambda, s, n, t, \rho)$ now also computes $u_i \leftarrow F(x_i)$. The tracing key component tk_i and verification key component vk_i are set to be $\mathsf{tk}_i = \mathsf{vk}_i = u_i$ (instead of x_i).
2. The tracing algorithm Trace computes the list $H = \{h_j\}_j$ of polynomials as before. For each j, Trace now checks if h_j is the correct polynomial by factoring it to find its roots, $w_1, \ldots, w_f$. It then computes $u'_i \leftarrow F(w_i)$ for every $i \in [f]$, and looks for u'_i in tk. If for some index $i \in [f]$, u'_i does not appear in the tracing key tk, then Trace eliminates h_j from the list H. Suppose that at the end of this process, there is at least one polynomial left in H. Let h^* be the first polynomial in H and let $w_1^*, \ldots, w_f^*$ be its roots. For each $i \in [f]$, Trace finds the index $k \in [n]$ for which $F(w_i^*) = u_k$, and adds k to the subset $\mathcal{I}$ of corrupted parties. The proof π is set to be $(w_1^*, \ldots, w_f^*)$.
3. To verify a proof $\pi = \{w_1, \ldots, w_f\}$ against a subset $\mathcal{I} = \{i_1, \ldots, i_f\}$, the verification algorithm Verify checks that $F(w_j) = u_{i_j}$ for $j \in [f]$, where $\mathsf{vk} = (u_1, \ldots, u_n)$.

We denote the resulting scheme by NITS (for "Non-Imputable Traceable Shamir"). The following theorem establishes its non-imputability.

Theorem 3. *For every adversary $\mathcal{A}$ for the non-imputabiltiy of NITS, there exists an algorithm $\mathcal{B}$ such that for every $\lambda \in \mathbb{N}$, it holds that*

$$\mathsf{Adv}^{\mathrm{ni}}_{\mathcal{A},\mathsf{NITB}}(\lambda) = \Pr\left[F(\mathcal{B}(F(x^*))) = F(x^*)\right],$$

where $x^* \leftarrow\!\!\$\ \mathbb{F}$ and the probability is also over the random coins of $\mathcal{B}$.

Proof. Let $\mathcal{A}$ be as in the statement of the theorem and consider the following adversary $\mathcal{B}$ trying to invert F. The inverter $\mathcal{B}$ gets $u^* = F(x^*)$ as input for a uniformly random $x^* \leftarrow\!\!\$\ \mathbb{F}_p$. It then simulates the non-imputability experiment $\mathbf{ExpNI}_{\mathcal{A},\mathsf{NITS}}(\lambda)$ to $\mathcal{A}$:

1. $\mathcal{B}$ invokes $\mathcal{A}(\lambda)$ and gets n, t, i^* and the secret s from $\mathcal{A}$.
2. $\mathcal{B}$ samples a correlation string $\rho = (a_1, \ldots, a_{t-1}) \leftarrow\!\!\$\ \mathbb{F}^{t-1}$. It defines the polynomial $q(X) = s + \sum_{i=1}^{t-1} a_i X^i$.
3. For each $i \in [n] \setminus \{i^*\}$, $\mathcal{B}$ samples $x_i \leftarrow\!\!\$\ \mathbb{F}$.
4. $\mathcal{B}$ then generates the secret shares, tracing key, and verification key to pass to $\mathcal{A}$:
   (a) To generate secret shares for every $i \in [n] \setminus \{i^*\}$, $\mathcal{B}$ computes $y_i \leftarrow q(x_i)$ and sets $\mathsf{sh}_i = (x_i, y_i)$. Note that $\mathcal{B}$ does not need to pass to $\mathcal{A}$ the secret share of party i^*.
   (b) To generate the tracing key tk and verification key:
      – For every $i \in [n] \setminus \{i^*\}$, $\mathcal{B}$ computes $u_i \leftarrow F(x_i)$ and sets $\mathsf{tk}_i = \mathsf{vk}_i = u_i$.
      – For party i^*, $\mathcal{B}$ sets $\mathsf{tk}_{i^*} = \mathsf{vk}_{i^*} = u^*$.
   The tracing key is $\mathsf{tk} = (\mathsf{tk}_1, \ldots, \mathsf{tk}_n)$ and the verification key is $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$.
5. $\mathcal{A}$ then outputs a subset $\mathcal{I}^*$ and a proof π. If $i^* \in \mathcal{I}^*$ and π is accepting, then it must include a field element x' such that $F(x') = u^*$. $\mathcal{B}$ thus outputs this x'. If $\mathcal{A}$ does not output $\mathcal{I}^*$ and π satisfying these conditions, $\mathcal{B}$ aborts.

Observe that $\mathcal{B}$ perfectly simulates $\mathbf{ExpNI}_{\mathcal{A},\mathsf{NITS}}(\lambda)$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ outputs a preimage of u^* if and only if $\mathcal{A}$ outputs a subset that contains i^* and an accepting proof. Hence,

$$\mathsf{Adv}^{\mathrm{ni}}_{\mathcal{A},\mathsf{NITB}}(\lambda) = \Pr\left[F(\mathcal{B}(F(x^*))) = F(x^*)\right],$$

and the theorem follows. □

4 A Scheme Based on Blakley Secret Sharing

In this section, we show how to trace leakage in a variant of Blakley's seminal secret sharing scheme [5]. We begin by putting forth an extension of Blakley's original scheme. We then present a very efficient tracing algorithm for this extended scheme.

4.1 An Extended Blakley Scheme

We begin by recalling Blakley's original scheme [5], and then present our extended scheme.

Blakley's original scheme and its limitations. In Blakley's scheme, the secret is encoded as one coordinate of a point in a field $\mathbb{F}_p^t$, where p is an appropriately chosen modulus, and t is the threshold. Concretely, the dealer chooses a random point $\boldsymbol{x} \in \mathbb{F}_p^t$ conditioned on its first coordinate x_1 being equal to the secret $s \in \mathbb{F}_p$. The secret of party i is then a uniformly random hyperplane $H_i \subset \mathbb{F}_p^t$ that passes through $\boldsymbol{x}$. For a large enough modulus p, a collection of t dealt hyperplanes intersect in a single point with high probability. Since by design, all hyperplanes pass through $\boldsymbol{x}$, this unique intersection point must be $\boldsymbol{x}$.

As for secrecy, observe that the intersection of any k hyperplanes is a subspace of dimension at least $t-k$. Hence, if $k<t$, this subspace is of dimension at least 1. Suppose for simplicity that it is of dimension exactly 1; that is, it is a line ℓ in $\mathbb{F}_p^t$. If ℓ is not perpendicular to the axis of the first coordinate (which happens with very small probability), then all points on ℓ have different first coordinates. Intuitively, this means that the line reveals no information about what is the first coordinate of the secret point $\boldsymbol{x}$, and hence no information about the secret $s \in \mathbb{F}_p$. We will make everything precise when we present our extension of Blakely's scheme.

For our tracing procedure, however, we will need to encode the secret as a full point in $\mathbb{F}_p^t$. A naive extension of Blakley's scheme might set the secret $\boldsymbol{s} \in \mathbb{F}_p^t$ as the intersection point of all hyperplanes. This naive extension is, however, completely insecure! Observe that even one hyperplane – that is, a single secret share – reveals much information about the secret, as it contains only a $1/p$-fraction of the points in $\mathbb{F}_p^t$. More generally speaking, it is not hard to see that if we set the secret to be the first k coordinates of the intersection point $\boldsymbol{x} \in \mathbb{F}_p^t$, then already $t-k+1$ hyperplanes reveal information about the secret.

Our extended Blakley scheme. Instead of the naive approach above, we consider a more delicate way of "smearing" the secret across multiple entries of the intersection point $\boldsymbol{x}$, by introducing more randomness to the secret shares. Suppose that we want to encode the secret as a full point in $\mathbb{F}_p^t$. The idea is to randomly split $\boldsymbol{s}$ to t shares $\boldsymbol{r_1},\ldots,\boldsymbol{r_t}$ via standard additive secret sharing; that is, $\boldsymbol{r_1},\ldots,\boldsymbol{r_t}$ are uniformly random in $\mathbb{F}_p^t$ subject to $\boldsymbol{r_1}+\cdots+\boldsymbol{r_t}=\boldsymbol{s}$. Then, we share each of these $\boldsymbol{r_i}$s using the naive extension of Blakley's scheme described above. Meaning, the ith secret share now includes t random hyperplanes $H_i^{(1)},\ldots,H_i^{(t)}$ that pass through $\boldsymbol{r_1},\ldots,\boldsymbol{r_t}$, respectively. Per the security of this scheme, envision the t instances of "naive Blakley" as taking place in t orthogonal subspaces of $\mathbb{F}_p^{t^2}$. Then, any subset of $t-1$ parties holds $t\cdot(t-1)$ hyperplanes in $\mathbb{F}_p^{t^2}$. This allows us to hide exactly t elements of $\mathbb{F}_p$. Intuitively, this is since each "candidate secret" $\boldsymbol{s^*} \in \mathbb{F}_p^t$ adds t affine equations over $\mathbb{F}_p^{t^2}$, and so the affine system of equations induced by this secret and the $t\cdot(t-1)$ hyperplanes contains exactly t^2 equations. Hence, it is not overdetermined, making $\boldsymbol{s^*}$ plausible. The formal analysis requires much more care. We now present the scheme in detail.

The scheme, which we denote by $\mathsf{Blakley}^+$, is parameterized by a function $p=p(n,t,\lambda)$, determining the size of the finite field. The secret sharing algorithm takes in n,t, a secret vector $\boldsymbol{s} \in \mathbb{F}_p^t$, and a correlation string ρ that is interpreted as the randomness for the additive secret sharing of $\boldsymbol{s}$.

```
Share(1^λ, n, t, s, ρ) :

1. Parse $\rho$ as $\boldsymbol{r_2}, \dots, \boldsymbol{r_t} \in \mathbb{F}_p^t$ .
2. Set $\boldsymbol{r_1} \leftarrow \boldsymbol{s} - (\boldsymbol{r_2} + \cdots + \boldsymbol{r_t})$.
3. For $j = 1, \dots, t$: sample $\boldsymbol{a_j} \leftarrow\$ \mathbb{F}_p^t$ and set $b_j \leftarrow \langle \boldsymbol{a_j}, \boldsymbol{r_j} \rangle$.
4. Set the share to be $\mathsf{sh} = \{(\boldsymbol{a_j}, b_j)\}_{j\in[t]}$.
5. Output $\mathsf{sh}$.
```

The reconstruction algorithm takes in t shares and outputs a secret $\boldsymbol{s} \in \mathbb{F}_p^k$. For simplicity of presentation, we number the shares it takes in by $1, \dots, t$, but the algorithm is defined exactly the same for every subset of t shares.

```
Rec(sh_1, ..., sh_t):

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

Correctness. Let $n \in \mathbb{N}$ and $t \le n$, and let $\boldsymbol{s} \in \mathbb{F}_p^t$ be the secret. Let $\mathsf{sh}_1, \dots, \mathsf{sh}_t$ be any t shares out of the n shares generated by Share. We show that with high probability, $\mathsf{Rec}(\mathsf{sh}_1, \dots, \mathsf{sh}_t) = \boldsymbol{s}$. Fix $j \in [t]$, and consider the system of equations $\{\langle \boldsymbol{a_{i,j}}, \boldsymbol{R_j} \rangle = b_{i,j}\}_{i\in[t]}$. If $\{\boldsymbol{a_{i,j}}\}_{i\in[t]}$ are linearly independent, then this system has a unique solution that must be equal to the share $\boldsymbol{r_j}$ sampled by Share. The probability that $\{\boldsymbol{a_{i,j}}\}_{i\in[t]}$ are linearly independent is

$$\frac{\prod_{\ell=0}^{t-1}(p^t - p^\ell)}{p^{t^2}} = \prod_{\ell=1}^{t}(1 - p^{-\ell}) \ge 1 - \sum_{\ell=1}^{t} p^{-\ell} > 1 - \frac{1}{p-1}.$$

By union bound, the probability that Rec computes $\boldsymbol{r_1}, \dots, \boldsymbol{r_k}$ correctly is at least $1 - t/(p-1)$. Moreover, whenever $\boldsymbol{r_1}, \dots, \boldsymbol{r_t}$ are computed correctly, so is the secret $\boldsymbol{s}$. All in all, we get that the correctness error of our scheme can be bounded by $t/(p-1)$, which is negligible whenever t is polynomial and p is super-polynomial.

Succinctly representing hyperplanes. Consider the share dealt to party i in our scheme. It consists of t pairs $\{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j\in[t]}$, and hence its size is $t \cdot (t+1) \cdot \log p$. However, observe that for each $j \in [t]$, $\boldsymbol{a_{i,j}}$ a uniformly random vector in $\mathbb{F}_p^t$, independent of $\{\boldsymbol{a_{i,j'}}\}_{j'\neq j}$. Hence, instead of representing the $\boldsymbol{a_{i,j}}$s explicitly, we can replace them by a seed σ_i to a pseudorandom generator G. Now, to reconstruct the secret, one first expands σ_i to the $\boldsymbol{a_{i,j}}$s by computing $(\boldsymbol{a_{i,1}}, \dots, \boldsymbol{a_{i,t}}) \leftarrow G(\sigma_i)$ and then proceeds as before. This reduces the share size to $\lambda + t \cdot \log p$ (since the $b_{i,j}$s still need to be represented explicitly), at the expense of having only computational secrecy. Another option is to succinctly represent $(\boldsymbol{a_{i,1}}, \dots, \boldsymbol{a_{i,t}})$ using ϵ-biased sets [52, 1, 68]. This approach can reduce the share size to $\log(t^2 \log p) + O(\alpha) + k \cdot \log p$, while degrading statistical security by an additive $n \cdot 2^{-\alpha}$ factor. Finally, as observed by Brikell [15], all the $\boldsymbol{a_{i,j}}$s (or the succinct representations thereof) can be made public without affecting security.[9] If this is done, then the secret information each party needs to store is only of size $t \log p$, which exactly matches the size of the secret, making our Extended Blakley scheme ideal (meaning, secret shares are the same size as the secret).[10]

Secrecy. We now analyze the security of our extended scheme. We prove the following theorem.

Theorem 4. *For every $p = p(n, t, \lambda)$, the scheme* $\mathsf{Blakley}^{+}$ *described above is ϵ-secret for $\epsilon = (t+1)/(p-1)$.*

Proof. We prove that with high probability over the shares generation process, the shares of parties $1, \ldots, t-1$ reveal no information about the secret $\boldsymbol{s} \in \mathbb{F}_p^k$. The proof is identical for every other subset $\mathcal{J} \subset [n]$ of size t.

Let $\{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j\in[t]}$ denote the share of party i for $i = 1, \ldots, t-1$. For $i \in [t-1]$ and $j \in [t]$, let $\boldsymbol{\mu_{i,j}}^T = \left(0^{t\times(j-1)}, \boldsymbol{a_{i,j}}^T, 0^{t\times(t-j)}\right) \in \mathbb{F}_p^{t^2}$. That is, $\boldsymbol{\mu_{i,j}}$ is a vector of length t^2 composed of t blocks of length t each; the jth block is $\boldsymbol{a_{i,j}}$ and all other blocks are all 0s. For $\ell \in [t]$, let $\boldsymbol{\psi_\ell}^T = (\boldsymbol{e_\ell}^T, \boldsymbol{e_\ell}^T, \ldots, \boldsymbol{e_\ell}^T) \in \mathbb{F}_p^{t^2}$, where $\boldsymbol{e_\ell} \in \mathbb{F}_p^t$ is the ℓth standard basis vector for $\mathbb{F}_p^t$. That is, $\boldsymbol{\psi_\ell}$ is a vector of length t^2, where each entry $m \in [t^2]$ is 1 if $m = \ell \mod t$, and is 0 otherwise. Let $A_j = \{\boldsymbol{\mu_{i,j}}\}_{i\in[t-1]}$ for $j \in [t]$, let $A = \bigcup_{j\in[t]} A_j$, let $B = \{\boldsymbol{\psi_\ell}\}_{\ell\in[t]}$, and let $C = A \cup B$. Note that C contains at most t^2 vectors in $\mathbb{F}_p^{t^2}$.

The proof proceeds in two steps. First, we argue that if C spans $\mathbb{F}_p^{t^2}$ then every secret $\boldsymbol{s} \in \mathbb{F}_p^t$ has a unique collection of additive shares $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t} \in \mathbb{F}_p^t$ such that $\boldsymbol{s} = \boldsymbol{r_1} + \cdots + \boldsymbol{r_t}$, and these additive shares form a solution to the set of linear equations induced by the $t-1$ secret shares. We deduce that in this case, the shares reveal no information about the secret $\boldsymbol{s}$. Secondly, we bound the probability that C spans $\mathbb{F}_p^{t^2}$.

Suppose C spans $\mathbb{F}_p^{t^2}$, and let $\boldsymbol{s} = (s_1, \ldots, s_t) \in \mathbb{F}_p^t$ be some secret. Consider the following system of (affine) linear equations

$$\begin{cases} \langle \boldsymbol{a_{i,j}}, \boldsymbol{R_j} \rangle = b_{i,j} \text{ for } j \in [t] \text{ and } i \in [t-1] \\ R_{1,\ell} + \cdots + R_{t,\ell} = s_\ell \text{ for } \ell \in [t] \end{cases}$$

where $\boldsymbol{R_j} = (R_{j,1}, \ldots, R_{j,t})$. This is a system with t^2 equations in t^2 variables $\{R_{j,\ell}\}_{j\in[t],\ell\in[t]}$. Observe that this system can be expressed in matrix form as

$$M\boldsymbol{z} = \boldsymbol{b},$$

where M is a $t^2 \times t^2$ matrix whose rows are the vectors in the set C, $\boldsymbol{z} = (\boldsymbol{R_1}, \ldots, \boldsymbol{R_t}) \in \mathbb{F}_p^{t^2}$ is the vector of indeterminates, and $\boldsymbol{b} \in \mathbb{F}_p^{t^2}$ is the solution vector that depends on $\{b_{i,j}\}_{i\in[t-1],j\in[t]}$ and the secret $\boldsymbol{s}$. Since we assumed C spans $\mathbb{F}_p^{t^2}$ and M is a square matrix, it follows that M is full rank. Hence, the system has a unique solution. Since this holds for every secret $\boldsymbol{s}$, it follows that conditioned on C spanning $\mathbb{F}_p^{t^2}$, the distribution over the $t-1$ shares is identical for any two shares $\boldsymbol{s}, \boldsymbol{s'} \in \mathbb{F}_p^t$.

[9] Brikell [15] made this observation with respect to Blakley's original scheme, but it equally applies to our scheme.
[10] In our full-fledged scheme with non-imputability, it is not possible to publish all $\boldsymbol{a_{i,j}}$s in the clear to obtain an ideal secret sharing scheme. However, for secrets in $\{0,1\}^\lambda$, our derandomization approach gives shares of size 2λ, coming close to it.

Lemma 2. *The set C is linearly independent with probability at least $1-(t+1)/(p-1)$, where the probability is over the choice of vectors in A.*

Proof (Lemma 2). We prove the lemma in two steps. First, we lower bound the probability that each A_j is linearly independent. Then, we upper bound the probability that there is a linear combination of the vectors in C that assigns non-zero coefficients to any of the vectors in B and evaluates to $\mathbf{0}$.

First, observe that for each $j \in [t]$, A_j is linearly independent if and only if the set of vectors $\{\boldsymbol{a_{1,j}}, \ldots, \boldsymbol{a_{t-1,j}}\}$ is linearly independent over $\mathbb{F}_p^t$. This probability is at least $1-1/(p-1)$, as proven in our correctness analysis. By a union bound, the probability that each A_j is linearly independent is at least $1-t/(p-1)$. Observe that conditioned on each A_j being linearly independent, any linear combination of the vectors in C that assigns zero coefficient to all the vectors in B **cannot** give the zero vector $\mathbf{0} \in \mathbb{F}_p^{t^2}$.

Now consider a linear combination of the vectors in C that assigns a non-zero coefficient to at least one vector in B. Such a linear combination can be written as

$$\boldsymbol{v} = \sum_{j\in[t]} \alpha_j \boldsymbol{\psi_j} + \sum_{j\in[t]} \sum_{i\in[t-1]} \beta_{i,j} \boldsymbol{\mu_{i,j}},$$

where at least one α_j is non-zero. By the structure of $\{\boldsymbol{\psi_1}, \ldots, \boldsymbol{\psi_t}\}$, it holds that

$$\sum_{j\in[t]} \alpha_j \boldsymbol{\psi_j} = (\boldsymbol{\eta}, \boldsymbol{\eta}, \ldots, \boldsymbol{\eta}),$$

for some non-zero vector $\boldsymbol{\eta} \in \mathbb{F}_p^t$. Now, if $\boldsymbol{v} = 0$, this means that $\boldsymbol{\eta}$ is in the linear span of $\{\boldsymbol{a_{1,j}}, \ldots, \boldsymbol{a_{t-1,j}}\}$ for **every** index $j \in [t]$. In other words $\boldsymbol{\eta}$ is in the intersection of the subspaces spanned by $\{\boldsymbol{a_{1,j}}, \ldots, \boldsymbol{a_{t-1,j}}\}$ for $j \in [t]$. That is, if we denote

$$V = \bigcap_{j\in[t]} \mathsf{Span}\left(\{\boldsymbol{a_{1,j}}, \ldots, \boldsymbol{a_{t-1,j}}\}\right),$$

then it holds that $\boldsymbol{\eta} \in V$. If $V = \{\mathbf{0}\}$, then we have arrived at a contradiction, and it must be the case that $\boldsymbol{\eta} = \mathbf{0}$. Note that for every $j \in [t]$, the dimension of $\mathsf{Span}\left(\{\boldsymbol{a_{1,j}}, \ldots, \boldsymbol{a_{t-1,j}}\}\right)$ is at most $t-1$. Hence, the orthogonal subspace is of dimension at least 1; let w_j be a uniformly random vector in this subspace. Observe that if $w_1, \ldots, w_t$ span $\mathbb{F}_p^t$ then V is trivial. Since $w_1, \ldots, w_t$ are independent and uniform in $\mathbb{F}_p^t$, the same analysis as before shows that they span $\mathbb{F}_p^t$ with probability at least $1-1/(p-1)$.

We have shown that:

1. If each A_j is linearly independent, there can be no linear combination of C that yields $\mathbf{0}$ and assigns all zero coefficients to B. The probability that at each A_j is linearly independent is $1-t/(p-1)$.
2. If the subspace V is trivial, there can be no linear combination of C that yields $\mathbf{0}$ and assigns a non-zero coefficient to a vector in B. This occurs with probability at least $1-1/(p-1)$.

By a union bound, we obtain that the probability that C is linearly independent is at least $1-(t+1)/(p-1)$. □

We proved that C is linearly independent with probability at least $1-(t+1)/(p-1)$, and that conditioned on C being linearly independent, the distributions over any tuple of $t-1$ shares are identical for any two secrets $\boldsymbol{s}$ and $\boldsymbol{s'}$. Hence, the statistical distance between the two distributions is at most $(t+1)/(p-1)$, concluding the proof of the theorem. □

4.2 The Basic Tracing Procedure

We now present our tracing algorithm for the extended Blakley scheme described above. We begin by presenting a tracing algorithm that relies on the full knowledge of all secret shares $\mathsf{sh}_1, \dots, \mathsf{sh}_n$ dealt to the parties. In the following section, we show how to (slightly) modify the tracing key and the tracing algorithm so that tracing does not require explicit knowledge of the secret shares, and the scheme can thus provide non-imputability.

We begin with an informal overview of the construction. In $\mathsf{Blakeley}^+$, the secret is a point $\boldsymbol{s} \in \mathbb{F}_p^t$. To derive the n shares, the Share algorithm samples $\boldsymbol{r_1}, \dots, \boldsymbol{r_t}$ that sum up to $\boldsymbol{s}$, and for $i \in [n]$, the ith share consists of t random hyperplanes $H_i^{(1)}, \dots, H_i^{(t)}$, passing through $\boldsymbol{r_1}, \dots, \boldsymbol{r_t}$, respectively. The tracing key tk will consist of $\boldsymbol{r_2}, \dots, \boldsymbol{r_t}$, and in addition, the first hyperplane of each party, $H_1^{(1)}, \dots, H_n^{(1)}$. The verification key vk consists of $H_i^{(1)}, \dots, H_i^{(t)}$. For the sake of simplicity, suppose in this informal overview that we know that $t-1$ secret shares are hardcoded to the reconstruction box R, which takes in one additional share as input. To do that, we sample a random share to feed into R, consistently with $\boldsymbol{r_1}, \dots, \boldsymbol{r_t}$. That is, for every $j \in [t]$, we sample a hyperplane $H^{(j)}$ as follows:

1. For $j = 2, \dots, t$: Sample a hyperplane $H^{(j)}$ uniformly at random, conditioned on passing through $\boldsymbol{r_j}$.
2. For $j = 1$, sample $H^{(1)}$ uniformly at random.

The tracing algorithm then feeds the share $\mathsf{sh} = (H^{(1)}, H^{(2)}, \dots, H^{(t)})$ to R, which returns some secret $\boldsymbol{s'} \in \mathbb{F}_p^t$. Suppose for simplicity that R is perfectly correct. Then, by construction, we know that for each $j \geq 2$, $H^{(j)}$ intersects with the jth hyperplanes of the shares embedded in R at $\boldsymbol{r_j}$. This means that $H^{(1)}$ intersects the first hyperplanes of the shares embedded in R at $\boldsymbol{r'_1} = \boldsymbol{s'} - (\boldsymbol{r_2} + \cdots + \boldsymbol{r_t})$. The tracing algorithm thus outputs all parties $i \in [n]$ for which $\boldsymbol{r'_1}$ is on their first hyperplane $H_i^{(1)}$. Since R is correct, for every party i whose share is hardcoded to R, $\boldsymbol{r'_1}$ will indeed be on $H_i^{(1)}$, and so all corrupted parties are caught by our tracing algorithm. On the other hand, if party i is uncorrupted, then $\boldsymbol{r'_1}$ will be statistically independent from $H_i^{(1)}$, and hence will lie on it with very small probability, so no honest parties are wrongly accused.

We now present the tracing algorithm in detail. For simplicity of presentation, we assume that the number of corruptions $f < t$ hardcoded into the reconstruction box R is given as input to Trace. This assumption may be removed in the same manner as in Section 3.

```
Trace^R(tk, f, 1^{1/ε}):

1. Parse tk as $(\boldsymbol{r_2}, \dots, \boldsymbol{r_k}, (\boldsymbol{a_{1,1}}, b_{1,1}), \dots, (\boldsymbol{a_{n,1}}, b_{n,1}))$.
2. For $\ell = 1, \dots, 2\lambda/\epsilon$:
   (a) Sample $t-f$ shares to feed to $R$. For $i = 1, \dots, t-f$:
       i. For $j = 2, \dots, t$: sample $\boldsymbol{a'_{i,j}} \leftarrow\$ \mathbb{F}_p^t$ and set $b'_{i,j} \leftarrow \left\langle \boldsymbol{a'_{i,j}}, \boldsymbol{r_j} \right\rangle$.
          // sample a random hyperplane that passes through $\boldsymbol{r_j}$
       ii. For $j = 1$: $\boldsymbol{a'_{i,1}} \leftarrow\$ \mathbb{F}_p^t$ and $b'_{i,1} \leftarrow\$ \mathbb{F}_p$.
          // sample a uniformly random hyperplane
       iii. Set $\mathsf{sh}'_i = \left\{ (\boldsymbol{a'_{i,j}}, b'_{i,j}) \right\}_{j \in [t]}$.
   (b) Query $R$ on $(\mathsf{sh}'_1, \dots, \mathsf{sh}'_{t-f})$ and get back a secret $\boldsymbol{s} \in \mathbb{F}_p^t$.
```

```
    (c) Compute $\boldsymbol{r_1} \leftarrow \boldsymbol{s} - (\boldsymbol{r_2} + \cdots + \boldsymbol{r_k})$.
    (d) Compute the subset $\mathcal{I} \subseteq [n]$ as the set of indices $i \in [n]$ for which $\langle \boldsymbol{a_{i,1}}, \boldsymbol{r_1} \rangle = b_{i,1}$. Set
        $\pi \leftarrow \{(\boldsymbol{a_{i,1}}, b_{i,1})\}_{i \in \mathcal{I}}$.
    (e) If $|\mathcal{I}| = f$, output $\mathcal{I}$ and $\pi$ and terminate.
3. If reached, output $\emptyset$.
```

The verification algorithm Verify takes in a verification key $\mathsf{vk} = (\boldsymbol{a_{1,1}}, b_{1,1}), \ldots, (\boldsymbol{a_{n,1}}, b_{n,1}))$, a subset $\mathcal{I}$ and a proof $\pi = \left\{\left(\boldsymbol{a'_{i,1}}, b'_{i,1}\right)\right\}_{i \in \mathcal{I}}$ and outputs 1 if and only if $\mathcal{I}$ and π are consistent with vk; that is $(\boldsymbol{a'_{i,1}}, b'_{i,1}) = (\boldsymbol{a_{i,1}}, b_{i,1})$ for every $i \in \mathcal{I}$. This makes it trivial for the tracer to frame innocent parties, but as mentioned, we will show how to address this fact in the following section.

Let $\mathsf{TB} = (\mathsf{Share}, \mathsf{Rec}, \mathsf{Trace}, \mathsf{Verify})$ (TB for "Traceable Blakley") be the traceable secret sharing scheme described above. That is, Share and Rec are as defined in Section 4.1, when Share additionally outputs tk and vk as described above. The algorithms Trace and Verify are as described in this section.

The traceability of TB is established by the following theorem. It proves that TB is universally traceable with respect to the partition of the correlation strings space into singletons; that is $\mathit{\Gamma}_{\mathsf{sngltn}} = \{\{(\boldsymbol{r_2}, \ldots, \boldsymbol{r_t})\} \ : \ \boldsymbol{r_2}, \ldots, \boldsymbol{r_t} \in \mathbb{F}_p^t\}$. Standard traceability then follows by Lemma 1.

Theorem 5. *For every adversary $\mathcal{A}$ and for every $\lambda \in \mathbb{N}$, and every $\epsilon \in [0,1]$ it holds that*

$$\mathsf{Adv}^{\text{uni-trac}}_{\mathcal{A},\mathsf{TB},\mathit{\Gamma}_{\mathsf{sngltn}},\epsilon}(\lambda) \leq e^{-\lambda} + \frac{2n\lambda}{\epsilon p},$$

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

Proof. Let $\lambda \in \mathbb{N}$, let $\epsilon \in [0,1]$, and let $\mathcal{A}$ be an adversary taking part in the $\mathbf{ExpUniTrace}_{\mathcal{A},\mathsf{TB},\mathit{\Gamma}_{\mathsf{sngltn}},\epsilon}(\lambda)$ security experiment. Let G denote the event in which $\mathcal{A}$ outputs a reconstruction box R that is $(n, t, \mathbf{sh}, \mathit{\Gamma}_{\mathsf{sngltn}}, \epsilon)$-good, where n and t are chosen by $\mathcal{A}$ at the beginning of the experiment, and $\mathbf{sh} = (\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f})$ are the shares given to it by the challenger. Conditioned on $\neg\mathsf{G}$, the output of the experiment is 0 with probability 1, and so we condition the rest of the analysis on G.

Note that in each iteration of Step 2, the tracing algorithm feeds R with uniformly random shares $\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}$ conditioned on the correlation string being $\boldsymbol{r_2}, \ldots, \boldsymbol{r_t}$. Hence, since R is $(n, t, \mathbf{sh}, \mathit{\Gamma}_{\mathsf{sngltn}}, \epsilon)$-good, in each iteration it holds that

$$R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \mathsf{Rec}(\mathsf{sh}_{i_1}, \ldots, \mathsf{sh}_{i_f}, \mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) \tag{6}$$

with probability at least ϵ. By definition of the reconstruction algorithm, whenever Eq. (6) holds, we know that

$$R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \boldsymbol{r_1} + \cdots + \boldsymbol{r_t}, \tag{7}$$

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

$$\{\langle \boldsymbol{a_{i_k,j}}, \boldsymbol{R_j} \rangle = b_{i_k,j}\}_{k \in [f]} \bigcup \{\langle \boldsymbol{a'_{i,j}}, \boldsymbol{R_j} \rangle = b'_{i,j}\}_{i \in [t-f]}$$

in the variables $\boldsymbol{R_j} = (R_{j,1}, \ldots, R_{j,t})$. Recall that Trace computes $\boldsymbol{r'_1}$ as

$$\boldsymbol{r'_1} = R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) - (\boldsymbol{r_2} + \cdots + \boldsymbol{r_t}). \tag{8}$$

Hence, whenever Eq. (7) holds, it also holds that $\boldsymbol{r'_1} = \boldsymbol{r_1}$. Hence, $\boldsymbol{r'_1}$ satisfies $\langle \boldsymbol{a_{i_k,1}}, \boldsymbol{r'_1} \rangle = b_{i_k,1}$ for $k = 1, \ldots, f$. This means that all corrupted parties – that is, parties $i_1, \ldots, i_f$ – are included in the set $\mathcal{I}'$ computed by Trace.

On the other hand, for every party $i \in [n] \setminus \{i_1, \ldots, i_f\}$, it holds that its share $\{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j\in[t]}$ is statistically independent from both the view of $\mathcal{A}$ and the shares $\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}$ sampled by Trace. Hence, it is independent of $\boldsymbol{r_1}$ and $\Pr\left[\langle \boldsymbol{a_{i,1}}, \boldsymbol{r'_1} \rangle = b_{i,1}\right] = 1/p$. By a union bound, the probability that any honest party is included in $\mathcal{I}'$ is at most t/p.

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

$$1 - (1 - \epsilon/2)^\ell \geq 1 - e^{-\epsilon\ell/2} = 1 - e^{-\lambda}. \tag{9}$$

Moreover, the probability that it outputs the wrong subset $\mathcal{I}' \neq \mathcal{I}$ in a given iteration is at most n/p, and hence the probability that it outputs the wrong subset in any of the $\ell = 2\lambda/\epsilon$ iterations is at most $2n\lambda/(\epsilon p)$.

Overall, we get that the advantage of $\mathcal{A}$ is bounded by $e^{-\lambda} + 2n\lambda/(\epsilon p)$, concluding the proof. □

4.3 Adding Non-Imputability

In the scheme as presented above, knowledge of the tracing key tk trivially lets one falsely accuse an innocent party of contributing its share to a pirate reconstruction box R. This violates the non-imputability property (Definition 5). The reason is that tk contains the first hyperplane $(\boldsymbol{a_{i,1}}, b_{i,1})$ of each party in the clear. In what follows, we show that this is not necessary. We observe, that full knowledge of $(\boldsymbol{a_{i,1}}, b_{i,1})$ is not required for tracing; all that is needed is the ability to determine whether a certain point – namely, $\boldsymbol{r_1}$ reconstructed by Trace – lies on the hyperplane induced by $(\boldsymbol{a_{i,1}}, b_{i,1})$. To this end, it is sufficient to include in the tracing key $f((\boldsymbol{a_{i,1}})_1), \ldots, f((\boldsymbol{a_{i,1}})_t)$ and $f(b_{i,1})$, where $\boldsymbol{a_{i,1}} = ((\boldsymbol{a_{i,1}})_1, \ldots, (\boldsymbol{a_{i,1}})_t)$ and f is a *homomorphic* one-way function. The tracing key tk and verification key vk then include these f-evaluations, where tk additionally includes $\boldsymbol{r_2}, \ldots, \boldsymbol{r_t}$ as before. The homomorphism allows Trace to check whether $\langle \boldsymbol{a_{i,1}}, \boldsymbol{r_1} \rangle = b_{i,1}$ ("under the hood" of f), and Verify can do the same. Intuitively, if party i is honest, then the reconstruction box R is statistically independent of $\boldsymbol{a_{i,1}}$. Hence, falsely accusing party i involves finding some non-trivial linear relation among the elements of $f((\boldsymbol{a_{i,1}})_1), \ldots, f((\boldsymbol{a_{i,1}})_t)$ and $f(b_{i,1})$, which is equivalent to inverting the one-way function f.

We now present the changes to the tracing and verification algorithm in detail, and claim that non-imputability holds. For concreteness, we focus on the case where f is the exponentiation function in a cyclic group, and hence its one-wayness is based on the hardness of the discrete log problem in the group. For asymptotic reasoning, we consider a distribution ensemble over discrete-log hard groups. This is formalized by the existence of a group generation algorithm $\mathcal{G}$ that takes the security parameter as input and outputs a triple $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a description of a group of order p generated by g. We also use the following standard notation. For a group element $h \in \mathbb{G}$ and a vector $\boldsymbol{x} = (x_1, \ldots, x_\ell)$ of $\mathbb{Z}_p$ elements, we write $h^{\boldsymbol{x}}$ to denote the vector $(h^{x_1}, \ldots, h^{x_\ell})$. For a vector $\boldsymbol{h} = (h_1, \ldots, h_\ell)$ of group elements and a vector $\boldsymbol{x}$ as before, we write denote $\boldsymbol{h}^{\boldsymbol{x}} := \prod_i h_i^{x_i}$.

The changes to the scheme are as follows:

1. The share algorithm $\mathsf{Share}(1^\lambda, s, n, t)$ now also samples a group $(\mathbb{G}, g, p) \leftarrow\$\ \mathcal{G}(1^\lambda)$. It computes $\boldsymbol{r_1}, \ldots, \boldsymbol{r_t}$ and $\mathsf{sh}_i = \{(\boldsymbol{a_{i,j}}, b_{i,j})\}_{j\in[t]}$ as before. Share then computes $\boldsymbol{y_i} \leftarrow g^{\boldsymbol{a_{i,1}}}$ and $z_i \leftarrow g^{b_{i,1}}$

for each $i \in [n]$. It sets the tracing key to be

$$\mathsf{tk} \leftarrow ((\mathbb{G}, g, p), \boldsymbol{r_2}, \ldots, \boldsymbol{r_t}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_n}, z_1, \ldots, z_n)$$

and the verification key $\mathsf{vk} \leftarrow ((\mathbb{G}, g, p), \boldsymbol{y_1}, \ldots, \boldsymbol{y_n}, z_1, \ldots, z_n)$.

2. The tracing algorithm Trace computes $\boldsymbol{r_1}$ as before, and sets the set $\mathcal{I}$ of corrupted party to be the set of all indices i that satisfy $\boldsymbol{y_i}^{\boldsymbol{r_1}} = z_i$. The proof π now consists of the vector $\boldsymbol{r_1}$.
3. To verify a proof $\pi = \boldsymbol{r_1}$ against a subset $\mathcal{I}$, the verification algorithm Verify checks that $\boldsymbol{y_i}^{\boldsymbol{r_1}} = z_i$ for every $i \in \mathcal{I}$ and that there exists an $i' \in [n] \backslash \mathcal{I}$ for which $\boldsymbol{y_{i'}}^{\boldsymbol{r_1}} \neq z_{i'}$. Verify outputs 1 if and only if both of these conditions are met.

We denote the scheme resulting from these changes as NITB (for "Non-Imputable Traceable Blakeley"). The fact that NITB satisfies correctness, secrecy, and traceability follows from the same analysis as in previous subsections. We now prove that NITB additionally satisfies non-imputability. We do so, by reducing it to the problem of finding a non-trivial linear-in-the-exponent relation among uniformly-random group elements in the group $\mathbb{G}$.

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*

$$\mathsf{Adv}^{\mathrm{dlr}}_{\mathcal{A},\mathcal{G}}(\lambda) := \Pr\left[\langle \boldsymbol{a}, \boldsymbol{r} \rangle = 0 \ \wedge\ \boldsymbol{r} \neq \boldsymbol{0} \ :\ \begin{array}{c} (\mathbb{G}, p, g) \leftarrow\$ \mathcal{G}(1^\lambda) \\ \boldsymbol{a} \leftarrow\$ \mathbb{Z}_p^\ell \\ \boldsymbol{r} \leftarrow\$ \mathcal{A}((\mathbb{G}, p, g), g^{\boldsymbol{a}}) \end{array} \right] < \nu(\lambda)$$

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 $\mathbb{G}$ (see, for example [14]).

Theorem 6. *For every adversary $\mathcal{A}$ for the non-imputabiltiy of* NITB*, there exists an algorithm $\mathcal{B}$ for the DLR problem, such that for every $\lambda \in \mathbb{N}$, it holds that*

$$\mathsf{Adv}^{\mathrm{ni}}_{\mathcal{A},\mathsf{NITB}}(\lambda) = \mathsf{Adv}^{\mathrm{dlr}}_{\mathcal{B},\mathcal{G}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an adversary participating in the non-imputability experiment for NITB. We construct an algorithm $\mathcal{B}$ that breaks the discrete log relation problem with respect to $\mathcal{G}$. $\mathcal{B}$ gets as input a vector $g^{\boldsymbol{x}} = (g^{x_1}, \ldots, g^{x_t})$ of group elements sampled uniformly at random from $\mathbb{G}$. 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 tracing key, verification key, 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 $i \in [n] \setminus \{i^*\}$, $\mathcal{B}$ 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 $b_{i,j} = \langle \boldsymbol{a_{i,j}}, \boldsymbol{r_j} \rangle$. $\mathcal{B}$ also computes $\boldsymbol{y_i} \leftarrow g^{\boldsymbol{a_{i,1}}}$ and $z_i \leftarrow g^{b_{i,1}}$ for every $i \neq i^*$.
3. For $i = i^*$, $\mathcal{B}$ sets $\boldsymbol{y_{i^*}} \leftarrow g^{\boldsymbol{x}}$ and $z_{i^*} \leftarrow \boldsymbol{y_{i^*}}^{\boldsymbol{r_1}}$.
4. $\mathcal{B}$ gives $\mathcal{A}$ the shares $\{\mathsf{sh}_i\}_{i\in[n]\setminus\{i^*\}}$, as well as the tracing key

$$\mathsf{tk} \leftarrow ((\mathbb{G}, g, p), \boldsymbol{r_2}, \ldots, \boldsymbol{r_t}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_n}, z_1, \ldots, z_n)$$

and the verification key $\mathsf{vk} \leftarrow ((\mathbb{G}, g, p), \boldsymbol{y_1}, \ldots, \boldsymbol{y_n}, z_1, \ldots, z_n)$.

$\mathcal{A}$ then outputs a proof π implicating a subset 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 a vector $\boldsymbol{r^*}$ such that

$$\boldsymbol{y_{i^*}}^{\boldsymbol{r^*}} = z_{i^*}. \tag{10}$$

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

$$\boldsymbol{y_{i^*}}^{\boldsymbol{r_1}} = z_{i^*}. \tag{11}$$

Dividing Eq. (10) by Eq. (11), we obtain that

$$\boldsymbol{y_{i^*}}^{\boldsymbol{r^*} - \boldsymbol{r_1}} = 1_{\mathbb{G}}.$$

Recalling that $y_{i^*} = g^{\boldsymbol{x}}$, 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, $\mathsf{Adv}^{\mathrm{ni}}_{\mathcal{A},\mathsf{NITB}}(\lambda) = \mathsf{Adv}^{\mathrm{dlr}}_{\mathcal{B},\mathcal{G}}(\lambda)$, completing the proof of the theorem. $\square$

5 Leaker Confirmation and Applications to Threshold VUFs

In this section, we discuss the easier problem of *confirming* the identity of the corrupted parties who contributed to a reconstruction box R, rather than tracing R back to them. We will discuss how our tracing procedure for Shamir's secret sharing can be cast as a confirmation mechanism. We will present how this confirmation mechanism can be used to construct an *accountable threshold verifiable unpredictable function (VUF)*. Since our main objective in this section is to highlight the applicability of confirmation to threshold VUFs, we keep the presentation informal and high-level.

Confirmation vs. tracing. Consider the case where a coalition $\mathcal{I}$ of malicious share holders leaks their shares in the form of a reconstruction box R, and the secret owner, Alice, has a strong suspicion as to the identity of the parties in $\mathcal{I}$. In this setting, we would like Alice to be able to test the suspected subset $\mathcal{I}'$ against the reconstruction box R, to check if $\mathcal{I}'$ is indeed corrupted. In more detail, a secret-sharing scheme that supports confirmation comes equipped with a confirmation algorithm Test, that takes in a confirmation key ck and a subset $\mathcal{I}' \subseteq [n]$ of the parties, and gets oracle access to a reconstruction box R. Test outputs either 1, implying that $\mathcal{I}'$ is indeed the corrupted subset or 0, implying rejection of this assertion. The confirmation key ck is outputted by the sharing algorithm Share, similarly to tk in traceable secret sharing schemes. Informally, the guarantee should be that if $\mathcal{I}'$ is the corrupted subset (in the same sense as in Definition 4), then Trace outputs 1 with overwhelming probability. We assume that the reconstruction box R is available for whoever wants to confirm the fact that $\mathcal{I}'$ is corrupted, and we do not consider an additional notion of non-imputability.

Confirmation in Shamir secret sharing. Our tracing procedure for Shamir secret sharing readily gives a confirmation mechanism. In our confirmation mechanism, the confirmation key ck will be the same as the tracing key from Section 3, and will include the evaluation points $x_1, \ldots, x_n$. Consider first the case where Test is given access to a perfect reconstruction box R, that always outputs the correctly reconstructed secret. Moreover, suppose for simplicity that the subset $\mathcal{I}$ of corrupted parties is of size $t-1$. In this case, Test operates similarly to Trace in Section 3.1. Namely, it queries R on (x,y) and on $(x,y+1)$ for a uniformly random $x \leftarrow\!\!\$\ \mathbb{F}$ and some $y \in \mathbb{F}$. As explained in Section 3.1, this information allows Test to compute an evaluation of the function $h(X) = \prod_{j\in\mathcal{I}} \frac{x_j}{x_j - X}$ at the point $X = x$, where $\mathcal{I}$ is the real set of corrupted parties. Now consider the function $h'(X) = \prod_{j\in\mathcal{I}'} \frac{x_j}{x_j - X}$. Test outputs 1 if and only if $h(x) = h'(x)$. Without knowledge of the x_i values assigned to the parties, an adversary cannot construct a box R that tests against an innocent party. To see why that is, say that party i^* is innocent and consider a reconstruction box R and a subset $\mathcal{I}'$ that contains i^*. For Test to accept $\mathcal{I}'$ as corrupt, it must be that

$$R(x,y+1) - R(x,y) = \frac{x_{i^*}}{x_{i^*} - x} \cdot \prod_{j\in\mathcal{I}'\setminus\{i^*\}} \frac{x_j}{x_j - x} \tag{12}$$

Since i^* is innocent, the value $R(x,y+1) - R(x,y)$ computed by Test is independent of x_{i^*}. By definition, x and x_j for $j \in \mathcal{I}' \setminus \{i^*\}$ are also independent of x_{i^*}. Hence, since (12) simplifies to a linear equation in x_{i^*}, it holds with probability $1/|\mathbb{F}|$.

In case R is not a perfect reconstruction box, but only outputs the reconstructed secret with probability ϵ, the above procedure needs to be repeated $\lambda \log(1/\epsilon)$ times (with values y and $y+\delta$ for random y and δ as in our full-fledged Shamir tracing procedure). Test will then output 1 if at least one of these tests succeeded.

Accountability in threshold VUFs. A verifiable unpredictable function (VUF) [49] is a family $f = \{f_{\mathsf{ek}}\}$ of functions (keyed by an evaluation key ek), in which an output $w = f_{\mathsf{ek}}(z)$ of the function comes equipped with a proof π asserting its validity. Crucially, π can be publicly verified using a public verification key vk that does not hamper the unpredictability of the function. A threshold VUF [54, 56, 47, 26] is a VUF in which the evaluation key ek is shared among n parties, each of which holds a key share ek_i. On input z, the share ek_i may be used to produce a partial evaluation w_i of the function. Any t partial evaluation can then be combined to give $w = f_{\mathsf{ek}}(z)$ (and the corresponding proof). Informally, the security guarantee is that the function remains unpredictable even for adversaries holding up to $t-1$ key shares. Threshold VUFs recently found important applications in blockchains, including for randomness beacons [30, 17, 20, 25] and deterministic wallets [24, 55].

Several constructions of threshold VUFs have been suggested over the years (see, for example, [47, 26, 28, 30, 25] and the references therein). Among these, one of the most efficient constructions builds on BLS signatures [8, 47] in bilinear groups. The evaluation key ek is an element of $\mathbb{Z}_p$, and the value of an input z to the function is $f_{\mathsf{ek}}(z) = H(z)^{\mathsf{ek}}$ where H is a hash function (modeled as a random oracle) mapping f-inputs to elements in the source group. The verification key is g^{ek}, where g is a generator of the source group, and to verify an output w against an input z, one can check that $e(H(z), \mathsf{vk}) = e(w, g)$, where e is the pairing operation. Making this into a threshold VUF can be done using techniques that have become standard by now: share ek using Shamir secret sharing. If party i holds the secret evaluation key (x_i, y_i), then its partial evaluation on input z is $(x_i, H(z)^{y_i})$. Combining partial evaluations into the function's output can be done using Lagrange interpolation in the exponent.

As in secret sharing, one can think of the worrisome scenario in which evaluators of a threshold VUF leak their secret evaluation keys. Similarly to the secret sharing case, we can model such leakage as an evaluation box E that has $f < t$ shares of the evaluation key hardcoded in it. It takes in an f-input z along with $t-f$ partial evaluations of f on z, and outputs the value of the function $f_{\mathsf{ek}}(z)$. In the case of the BLS-based threshold VUF, we can use our confirmation mechanism to confirm that a subset $\mathcal{I}'$ of parties is behind the evaluation box E. To do so, the confirmation algorithm Test queries E twice on an arbitrary input z, once with the partial evaluation $(x, H(z)^y)$ and once with the partial evaluation $(x, H(z)^{y+1})$. Dividing the two responses gives $g^{h(x)}$, where $h(X) = \prod_{j \in \mathcal{I}} \frac{x_j}{x_j - X}$ is the function defined above. Test can then test whether $g^{h(x)} = g^{h'(x)}$ where $h'(X) = \prod_{j \in \mathcal{I}'} \frac{x_j}{x_j - X}$ is the same as defined above with respect to the suspected subset $\mathcal{I}'$. As before, if E is not perfect, the probability of successful confirmation can be amplified by repetition.

6 Discussion and Future Directions

This work raises several open questions regarding traceable secret sharing and coding theory.

Tracing other secret sharing schemes. In this work, we presented tracing procedures for two classic secret sharing schemes (or variants thereof) – those of Shamir [65] and of Blakley [5]. An interesting open question is to devise tracing procedures for other existing secret-sharing schemes. One prime candidate that comes to mind is the Asmuth-Bloom secret sharing scheme, based on the Chinese Remainder Theorem [3, 50, 35]. Coming up with such a tracing procedure is not only a very interesting number theoretic problem, it may also pave the way for traceable secret sharing beyond standard threshold access structures. In particular, as was previously observed, the Asmuth-Bloom scheme can be efficiently extended to give secret sharing for the *weighted threshold* access structure, in which each party is associated with a weight and the secret can be reconstructed if and only if the cumulative weight of the reconstruction quorum clears a certain threshold [72, 31].

Traceable threshold VUFs. In Section 5 we presented an efficient confirmation mechanism for confirming the identity of leaking parties in Shamir secret sharing and discussed how this mechanism may be applied to confirm the identity of leakers in BLS-based threshold VUFs [49, 8, 47]. An important and interesting open question is to come up with a VUF that has an efficient *tracing* procedure for finding out who the leaking parties are, rather than just confirming their identities. Though it is tempting to try and use our tracer for Shamir secret sharing to solve this task, this encounters a problem; this tracing procedure is not linear, and hence cannot be performed in the exponent. Our tracing procedure for Blakley's scheme *is* linear, but reconstruction in this scheme is not. This puts forth the following problem: construct a traceable secret sharing scheme that has linear reconstruction and linear or quadratic tracing.[11] Such a traceable secret sharing scheme will immediately yield a traceable threshold VUF from BLS.

Connection to erasure codes. Many works have observed a tight connection between secret sharing and erasure codes (e.g., [48, 18, 22, 23, 2]). It is easy to see that a t-out-of-n threshold secret sharing scheme immediately gives rise to an erasure code with words of length n that can tolerate up to $n - t$ erasures. The encoding of a word w is the concatenation of its n secret shares (with some fixed randomness for the sharing algorithm). Efficient decoding is guaranteed by the correctness of the secret-sharing scheme. The other direction is less clear: to share a secret s, one can encode it into a codeword of $m \geq n$ symbols. The share of each party is then a random index $i \in [m]$ together with the ith symbol in the codeword.[12] However, since encoding is deterministic, this naive approach cannot provide secrecy. A general template for fixing this issue is to first randomly embed the secret s in a higher dimension space, and only then encode it. If this embedding is reversible,

[11] Tracing may be quadratic, since the bilinear map can be used to compute quadratic functions in the exponent. Reconstruction, however, cannot be quadratic, since the output of the VUF needs to be an element of the source group to allow for efficient verification.

[12] If the code supports decoding from any n coordinates, indices can be assigned deterministically.

then correctness is preserved. Indeed, the previous works that drew a connection between secret sharing and erasure codes can be seen as falling within this paradigm. We observe that the seminal secret-sharing schemes tackled in this work, Shamir's and Blakley's, can also be viewed as special cases of this general paradigm. In Shamir's scheme, the secret $s \in \mathbb{F}_p$ is first randomly extended into a degree $t-1$ polynomial, which is then encoded using the Reed-Solomon code [63]. In Blakley's scheme, the secret $s \in \mathbb{F}_p$ is first randomly extended to a point $\boldsymbol{x}$ in $\mathbb{F}_p^t$. Then, $\boldsymbol{x}$ is encoded using a Hadamard-like code over the field $\mathbb{F}_p$. That is, positions in the codeword that encode $\boldsymbol{x}$ are indexed by vectors $\boldsymbol{a} \in \mathbb{F}_p^t$ (a codeword is of length p^t) and the $\boldsymbol{a}$-th position of the codeword is the inner product $\langle \boldsymbol{a}, \boldsymbol{x} \rangle$ (our extension of Blakley's scheme from Section 4.1 can be seen in similar terms).

Adopting this perspective, a fascinating open problem is to devise tracing procedures for other secret sharing schemes that are obtained from erasure codes. Of particular interest are secret-sharing schemes obtained from low-density parity-check codes (LDPC codes) [2, 22]. If such a tracing procedure is linear or quadratic, it might also help make progress towards traceable VUFs. More generally speaking, it may be interesting to study notions of tracing for error-correcting codes, and their potential applications beyond just secret sharing.

In our construction of traceable secret sharing, the codewords were of exponential size, which was necessary due to the non-imputability requirement. However, if we do not care about non-imputability, then the size m of the codeword may be as small as $\approx n^2$, where n is the number of parties. In particular, this means that the problem is meaningful already for explicit LDPC codes with polynomial-length codewords. Observe, that the tracing problem becomes essentially impossible to solve efficiently if $m \ll n^2$, for the same reasons as the lower bound sketched in Section 3.1.[13]

Tracing a reconstruction service. The literature on the related notion of traitor tracing for encryption schemes distinguishes between two types of pirate decoders that need to be traced back to a corrupted party. On the one hand, most works on traitor tracing (e.g., [46, 53, 6, 29, 64, 43, 51, 27, 16, 32, 13, 38, 19, 71, 70, 37]) consider a "decoder box" D_{box}, which is a *stateless* algorithm, whose output distribution is the same across different queries issued to it. On the other hand, some works [44, 61, 66] consider a "decoding service", which can be thought of as a *stateful* algorithm D_{service}, that can keep state across queries, altering its output distribution from one query to another. This makes tracing harder, since D_{service} may shut down if it notices that the queries issued to it are part of a tracing attempt.

The same distinction can be drawn in the setting of traceable secret sharing. The previous construction of Goyal, Song, and Srinivasan [41] can only trace back stateless reconstruction boxes. Our Blakley-based construction (Section 4) gives a solution to this problem, since tracing requires only one successful query. This is in contrast to our Shamir-based scheme (Section 3), in which tracing requires many different correlated queries.

Acknowledgments. This work was funded by NSF, DARPA, the Simons Foundation, UBRI, and NTT Research. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

[13] If $m \ll n^2$, it becomes hard to even make the reconstruction box R output anything other than $\perp$, since it requires an exponential number of queries in expectation to query R on a subset of positions such that none of which is corrupted. If $m = \Omega(n^2)$, though, a random subset will not intersect the corrupted subset with high probability.

References

1. N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *31st FOCS*, pages 544–553, St. Louis, MO, USA, Oct. 22–24, 1990. IEEE Computer Society Press.
2. B. Applebaum, O. Nir, and B. Pinkas. How to recover a secret with $o(n)$ additions. In H. Handschuh and A. Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 236–262, Santa Barbara, CA, USA, Aug. 20–24, 2023. Springer, Heidelberg, Germany.
3. C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29(2):208–210, 1983.
4. E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970.
5. G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.
6. D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353, Santa Barbara, CA, USA, Aug. 15–19, 1999. Springer, Heidelberg, Germany.
7. D. Boneh, A. Kiayias, and H. W. Montgomery. Robust fingerprinting codes: A near optimal construction. In *Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management*, DRM '10, page 3–12, New York, NY, USA, 2010. Association for Computing Machinery.
8. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532, Gold Coast, Australia, Dec. 9–13, 2001. Springer, Heidelberg, Germany.
9. D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 2008*, pages 501–510, Alexandria, Virginia, USA, Oct. 27–31, 2008. ACM Press.
10. D. Boneh, A. Partap, and L. Rotem. Accountability for misbehavior in threshold decryption via threshold traitor tracing. Cryptology ePrint Archive, Paper 2023/1724, 2023. https://eprint.iacr.org/2023/1724.
11. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.
12. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In D. Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465, Santa Barbara, CA, USA, Aug. 27–31, 1995. Springer, Heidelberg, Germany.
13. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499, Santa Barbara, CA, USA, Aug. 17–21, 2014. Springer, Heidelberg, Germany.
14. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
15. E. F. Brickell. Some ideal secret sharing schemes. In J.-J. Quisquater and J. Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 468–475, Houthalen, Belgium, Apr. 10–13, 1990. Springer, Heidelberg, Germany.
16. H. Chabanne, D. H. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 542–558, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.
17. Chainlink vrf: On-chain verifiable randomness. link.
18. H. Chen, R. Cramer, S. Goldwasser, R. de Haan, and V. Vaikuntanathan. Secure computation from random error correcting codes. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 291–310, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.
19. Y. Chen, V. Vaikuntanathan, B. Waters, H. Wee, and D. Wichs. Traitor-tracing from LWE made simple and attribute-based. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369, Panaji, India, Nov. 11–14, 2018. Springer, Heidelberg, Germany.
20. K. Choi, A. Manoj, and J. Bonneau. SoK: Distributed randomness beacons. In *2023 IEEE Symposium on Security and Privacy*, pages 75–92, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press.
21. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270, Santa Barbara, CA, USA, Aug. 21–25, 1994. Springer, Heidelberg, Germany.

22. R. Cramer, I. B. Damgård, N. Döttling, S. Fehr, and G. Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 313–336, Sofia, Bulgaria, Apr. 26–30, 2015. Springer, Heidelberg, Germany.
23. R. Cramer and C. Xing. Blackbox secret sharing revisited: A coding-theoretic approach with application to expansionless near-threshold schemes. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 499–528, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.
24. P. Das, S. Faust, and J. Loss. A formal treatment of deterministic wallets. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 651–668, London, UK, Nov. 11–15, 2019. ACM Press.
25. S. Das, B. Pinkas, A. Tomescu, and Z. Xiang. Distributed randomness using weighted vrfs. Cryptology ePrint Archive, Paper 2024/198, 2024. https://eprint.iacr.org/2024/198.
26. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 1–17, Miami, FL, USA, Jan. 6–8, 2003. Springer, Heidelberg, Germany.
27. Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 100–115, Miami, FL, USA, Jan. 6–8, 2003. Springer, Heidelberg, Germany.
28. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431, Les Diablerets, Switzerland, Jan. 23–26, 2005. Springer, Heidelberg, Germany.
29. A. Fiat and T. Tassa. Dynamic traitor training. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 354–371, Santa Barbara, CA, USA, Aug. 15–19, 1999. Springer, Heidelberg, Germany.
30. D. Galindo, J. Liu, M. Ordean, and J.-M. Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 88–102, 2021.
31. S. Garg, A. Jain, P. Mukherjee, R. Sinha, M. Wang, and Y. Zhang. Cryptography with weights: MPC, encryption and signatures. In H. Handschuh and A. Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 295–327, Santa Barbara, CA, USA, Aug. 20–24, 2023. Springer, Heidelberg, Germany.
32. S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 2010*, pages 121–130, Chicago, Illinois, USA, Oct. 4–8, 2010. ACM Press.
33. O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
34. O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32, Seattle, WA, USA, May 15–17, 1989. ACM Press.
35. O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. In *31st ACM STOC*, pages 225–234, Atlanta, GA, USA, May 1–4, 1999. ACM Press.
36. O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries: The highly noisy case. In *36th FOCS*, pages 294–303, Milwaukee, Wisconsin, Oct. 23–25, 1995. IEEE Computer Society Press.
37. J. Gong, J. Luo, and H. Wee. Traitor tracing with $N^{1/3}$-size ciphertexts and $O(1)$-size keys from k-Lin. In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 637–668, Lyon, France, Apr. 23–27, 2023. Springer, Heidelberg, Germany.
38. R. Goyal, V. Koppula, and B. Waters. Collusion resistant traitor tracing from learning with errors. In I. Diakonikolas, D. Kempe, and M. Henzinger, editors, *50th ACM STOC*, pages 660–670, Los Angeles, CA, USA, June 25–29, 2018. ACM Press.
39. V. Goyal. Reducing trust in the PKG in identity based cryptosystems. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 430–447, Santa Barbara, CA, USA, Aug. 19–23, 2007. Springer, Heidelberg, Germany.
40. V. Goyal, S. Lu, A. Sahai, and B. Waters. Black-box accountable authority identity-based encryption. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 2008*, pages 427–436, Alexandria, Virginia, USA, Oct. 27–31, 2008. ACM Press.
41. V. Goyal, Y. Song, and A. Srinivasan. Traceable secret sharing and applications. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 718–747, Virtual Event, Aug. 16–20, 2021. Springer, Heidelberg, Germany.
42. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *39th FOCS*, pages 28–39, Palo Alto, CA, USA, Nov. 8–11, 1998. IEEE Computer Society Press.
43. A. Kiayias and M. Yung. Self protecting pirates and black-box traitor tracing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 63–79, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer, Heidelberg, Germany.

44. A. Kiayias and M. Yung. On crafty pirates and foxy tracers. In *Security and Privacy in Digital Rights Management: ACM CCS-8 Workshop DRM 2001 Philadelphia, PA, USA, November 5, 2001 Revised Papers*, pages 22–39. Springer, 2002.
45. D. E. Knuth. *Art of computer programming, volume 2: Seminumerical algorithms*. Addison-Wesley Professional, 2014.
46. K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 145–157, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.
47. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer, Heidelberg, Germany.
48. J. L. Massey. Some applications of coding theory in cryptography. *Codes and Ciphers: Cryptography and Coding IV*, pages 33–47, 1995.
49. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130, New York, NY, USA, Oct. 17–19, 1999. IEEE Computer Society Press.
50. M. Mignotte. How to share a secret? In T. Beth, editor, *EUROCRYPT'82*, volume 149 of *LNCS*, pages 371–375, Burg Feuerstein, Germany, Mar. 29 – Apr. 2, 1983. Springer, Heidelberg, Germany.
51. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer, Heidelberg, Germany.
52. J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *22nd ACM STOC*, pages 213–223, Baltimore, MD, USA, May 14–16, 1990. ACM Press.
53. M. Naor and B. Pinkas. Threshold traitor tracing. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 502–517, Santa Barbara, CA, USA, Aug. 23–27, 1998. Springer, Heidelberg, Germany.
54. M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 327–346, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.
55. J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*, pages 1717–1731, Virtual Event, USA, Nov. 9–13, 2020. ACM Press.
56. J. B. Nielsen. A threshold pseudorandom function construction and its applications. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 401–416, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer, Heidelberg, Germany.
57. K. Nuida. A general conversion method of fingerprint codes to (more) robust fingerprint codes against bit erasure. In K. Kurosawa, editor, *ICITS 09*, volume 5973 of *LNCS*, pages 194–212, Shizuoka, Japan, Dec. 3–6, 2010. Springer, Heidelberg, Germany.
58. B. Pfitzmann. Trials of traced traitors. In R. Anderson, editor, *Information Hiding*, pages 49–64, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.
59. B. Pfitzmann and M. Schunter. Asymmetric fingerprinting (extended abstract). In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 84–95, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.
60. B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In R. Graveman, P. A. Janson, C. Neuman, and L. Gong, editors, *ACM CCS 97*, pages 151–160, Zurich, Switzerland, Apr. 1–4, 1997. ACM Press.
61. D. H. Phan. Traitor tracing for stateful pirate decoders with constant ciphertext rate. In *International Conference on Cryptology in Vietnam*, pages 354–365. Springer, 2006.
62. M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on computing*, 9(2):273–280, 1980.
63. I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.
64. R. Safavi-Naini and Y. Wang. Sequential traitor tracing. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 316–332, Santa Barbara, CA, USA, Aug. 20–24, 2000. Springer, Heidelberg, Germany.
65. A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, Nov. 1979.
66. T. Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. Cryptology ePrint Archive, Paper 2006/383, 2006. https://eprint.iacr.org/2006/383.
67. M. Sudan. Maximum likelihood decoding of reed solomon codes. In *37th FOCS*, pages 164–172, Burlington, Vermont, Oct. 14–16, 1996. IEEE Computer Society Press.

68. A. Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In H. Hatami, P. McKenzie, and V. King, editors, *49th ACM STOC*, pages 238–251, Montreal, QC, Canada, June 19–23, 2017. ACM Press.
69. G. Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2), may 2008.
70. H. Wee. Functional encryption for quadratic functions from k-lin, revisited. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 210–228, Durham, NC, USA, Nov. 16–19, 2020. Springer, Heidelberg, Germany.
71. M. Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 652–682, Santa Barbara, CA, USA, Aug. 17–21, 2020. Springer, Heidelberg, Germany.
72. X. Zou, F. Maino, E. Bertino, Y. Sui, K. Wang, and F. Li. A new approach to weighted multi-secret sharing. In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6. IEEE, 2011.

A Proof of Lemma 1

Proof (Lemma 1). Let $\mathcal{A}$, ϵ, δ, and λ be as in the statement of the lemma. Consider the tracing security experiment $\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda)$, and denote by $\alpha := \mathsf{Adv}^{\mathrm{trac}}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda)$. By definition 4 and our assumption that $\Pr\left[\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) = 1\right] \geq \frac{1}{\epsilon|\mathcal{SCRT}_\lambda|}$, we get that

$$\Pr\left[\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) = 1\right] = \frac{\alpha + \frac{1}{|\mathcal{SCRT}_\lambda|}}{\epsilon}. \tag{13}$$

Let G denote the event in which the reconstruction box R outputted by $\mathcal{A}$ is $(n, t, f, s^*, \rho, \epsilon)$-good, i.e., the probability that the experiment does not return 0 in line 8. Let T denote the event in which the tracing algorithm Trace outputs the correct subset $\mathcal{I}$ and an accepting proof in line 9. Then, by definition

$$\Pr\left[\mathbf{ExpTrace}_{\mathcal{A},\mathsf{TTSS},\epsilon,\delta}(\lambda) = 1\right] = \Pr\left[\mathsf{G} \wedge \neg\mathsf{T}\right] \tag{14}$$

where the probability is over the choice of $s^* \leftarrow\$\ \mathcal{SCRT}_\lambda$, $\rho \leftarrow\$\ \{0,1\}^\kappa$, $(\mathsf{sh}_i, \mathsf{tk}_i, \mathsf{vk}_i) \leftarrow\$\ \mathsf{Share}(1^\lambda, s^*, n, t, \rho)$, and the random coins of $\mathcal{A}$ and Trace.

We now consider the probability that the reconstruction box R outputs s^* on $t - f$ random well-formed secret shares of s^* *and* is not traced back to the subset $\mathcal{I}$ of corrupted parties. From Eq. (13) and (14), the definition of a good reconstruction box, and the law of total probability, we have that

$$\Pr\left[R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = s^* \wedge \neg\mathsf{T}\right] \geq \alpha + \frac{1}{|\mathcal{SCRT}_\lambda|}, \tag{15}$$

where the probability is over the same choices as in Eq. 14 and also over the choice of $(\mathsf{sh}'_i, \mathsf{tk}'_i, \mathsf{vk}'_i) \leftarrow\$\ \mathsf{Share}(1^\lambda, s^*, n, t, \rho)$ for $i = 1, \ldots, t - f$ and the random coins of R.

The correctness guarantee of TTSS implies that

$$\begin{aligned}\Pr\left[R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \mathsf{Rec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_f, \mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) \wedge \neg\mathsf{T}\right]&\\ \geq \alpha + \tfrac{1}{|\mathcal{SCRT}_\lambda|} - \nu_1,&\end{aligned} \tag{16}$$

Now consider a different way of sampling the random variables underlying the probability in Eq. (16). We sample random coins for $\mathcal{A}$, R and Trace as before. The f shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_f$ given to $\mathcal{A}$ are sampled by:

1. Sample a random subset $\mathcal{C} \in \Gamma$ by sampling $\rho \leftarrow\$\ \{0,1\}^\kappa$ and setting $\mathcal{C} \leftarrow \Gamma(\rho)$.
2. For each $i \in [f]$ we sample $\mathsf{sh}_i \leftarrow\$\ \mathcal{SH}(\mathcal{C})$.

We then sample $\mathsf{sh}'_1 \ldots, \mathsf{sh}'_{t-f}$ independently according to $\mathcal{SH}(\mathcal{C})$ as well.

By ν_2-bidirectionality, we obtain that

$$\Pr\left[R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \mathsf{Rec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_f, \mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) \wedge \neg\mathsf{T}\right] \geq \alpha + \tfrac{1}{|\mathcal{SCRT}_\lambda|} - (\nu_1 + \nu_2), \tag{17}$$

where the distribution is over the modified sampling of $\mathsf{sh}_1, \ldots, \mathsf{sh}_f$, $\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}$ as described above, and over the random coins of $\mathcal{A}, R$ and Trace.

By total probability, with probability at least $\alpha/2 + 1/|\mathcal{SCRT}_\lambda|$ over the choice of $\mathcal{C}$, the choice of $\mathsf{sh}_1, \ldots, \mathsf{sh}_f \leftarrow\$\ \mathcal{SH}(\mathcal{C})$ and the random coins of $\mathcal{A}$ and Trace it holds that $\neg\mathsf{T}$ occurs and

$$\Pr\left[R(\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f}) = \mathsf{Rec}(\mathsf{sh}_1, \ldots, \mathsf{sh}_f, \mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f})\right] \geq \frac{\alpha}{2} - (\nu_1 + \nu_2) \tag{18}$$

over the choice of $\mathsf{sh}'_1, \ldots, \mathsf{sh}'_{t-f} \leftarrow\$\ \mathcal{SH}(\mathcal{C})$ and the random coins of R. The lemma then follows from invoking bidirectionality one more time, and sampling the shares and secret as in the original experiment. □