Efficient Leakage Resilient Secret Sharing[*]

Peihan Miao   Akshayaram Srinivasan   Prashant Nalini Vasudevan

March 4, 2019

Abstract

A t-out-of-n threshold secret sharing scheme allows a dealer to share a secret among a set of n parties such that any subset of the parties of size at least t can recover the secret, while any subset of smaller size learns no information about the secret. A leakage resilient secret sharing scheme (introduced in independent works by (Goyal and Kumar, STOC 18) and (Benhamouda, Degwekar, Ishai and Rabin, Crypto 18)) additionally requires the secrecy to hold against every unauthorized set of parties even if they obtain some bounded leakage from the other shares. The leakage is said to be local if it is computed independently for each share. So far, the only known constructions of local leakage resilient threshold secret sharing schemes are for very low ($O(1)$) or very high ($n - o(\log n)$) thresholds.

In this work, we present local leakage resilient threshold secret sharing schemes with constant rate for any threshold. Furthermore, our scheme has optimal leakage-resilience rate, i.e., the ratio between the leakage tolerated and the size of each share can be made arbitrarily close to 1. We implement a variant of our scheme (that has worse rate and leakage-resilience rate, but better computational efficiency), and present comparisons of its performance to that of Shamir's secret sharing scheme (Shamir, Commun. ACM 1979).

Our construction generalizes to a rate-preserving compiler that adds local leakage-resilience to any secret sharing scheme for any monotone access structure.

1 Introduction

Secret sharing [Sha79, Bla79] is a fundamental cryptographic primitive that allows a secret to be shared among a set of parties in such a way that only certain authorized subsets of parties can recover the secret by pooling their shares together; and any subset of parties that is not authorized learn nothing about the secret from their shares. Secret sharing has had widespread applications across cryptography, ranging from secure multiparty computation [GMW87, BGW88, CCD88], threshold cryptographic systems [DF90, Fra90, DDFY94] and leakage resilient circuit compilers [ISW03, FRR$^+$10, Rot12].

While sufficient in idealized settings, in several practically relevant scenarios (as illustrated by the recent Meltdown and Spectre attacks [LSG$^+$18, KGG$^+$18], for instance), it is not satisfactory to assume that the set of unauthorized parties have no information at all about the remaining shares.

[*]UC Berkeley. Emails: {`peihan, akshayaram, prashvas`}`@berkeley.edu`. Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for LongTerm Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

They could, for instance, have access to some side-channel on the devices storing the other shares that leaks some information about them, and we would like for the secret to still remain hidden in this case. Such *leakage resilience* has been widely studied in the past as a desirable property in various settings and cryptographic primitives [MR04, DP08, AGV09, NS09]. In this paper, we study secret sharing that is resilient to leakage – we ask the secret remain hidden from unauthorized subsets of parties even if they have access to some small amount of information about the shares of the remaining parties.

More specifically, we are interested in *local* leakage resilience, which means that secrets are hidden from an adversary that works as follows. First, it specifies an unauthorized subset of parties, and for each of the remaining parties, it specifies a leakage function that takes its share as input and outputs a small pre-determined number of bits. Once the shares are generated, the adversary is given all the shares of the unauthorized subset, and the output of the corresponding leakage function applied to the each of the remaining shares.

This form of leakage resilience for secret sharing was formalized in recent work by Goyal and Kumar [GK18], and Benhamouda, Degwekar, Ishai and Rabin [BDIR18]. These papers showed examples of leakage-resilient threshold secret sharing schemes (where subsets above a certain size are authorized) for certain thresholds. They then showed applications of such schemes to constructions of leakage-resilient multi-party computation protocols and non-malleable secret sharing schemes. Given the prevalence of secret-sharing in cryptographic constructions and the importance of resilience to leakage, one may reasonably expect many more applications to be discovered in the future.

In this work, we are interested in constructing local leakage resilient secret sharing schemes for a larger class of access structures[1] (in particular, for all thresholds). Beyond showing feasibility, our focus is on optimizing the following parameters of our schemes:

- the *rate*, which is the ratio of the size of the secret to the size of a share, and,
- the *leakage-resilience rate*, which is the ratio of the number of bits of leakage tolerated per share to the size of a share.

Our primary result is a transformation that converts a secret sharing scheme for any monotone access structure $\mathcal{A}$ into a local leakage resilient secret sharing scheme for $\mathcal{A}$ whose rate is a small constant factor less than that of the original scheme, and which has an optimal asymptotic leakage-resilience rate (that tends to 1 as the length of the secret increases).

Informal Theorem 1.1 *There is a compiler that, given a secret sharing scheme for a monotone access structure $\mathcal{A}$ with rate R, produces a secret sharing scheme for $\mathcal{A}$ that has rate $R/3.01$ and is local leakage resilient with leakage-resilience rate tending to* 1.

In particular, for any $t \leq n$, starting from t-out-of-n Shamir secret sharing [Sha79] gives us a t-out-of-n threshold secret sharing scheme with rate $1/3.01$ and leakage rate 1. The only results known even for threshold access structures before our work were for either very small or very large thresholds. Goyal and Kumar [GK18] presented a construction for $t = 2$, which had both rate and leakage-resilience rate $\Theta(1/n)$. This was extended to any constant t by Badrinarayanan and Srinivasan [BS18], with rate $\Theta(1/\log(n))$ and leakage-resilience rate $\Theta(1/n\log(n))$. Benhamouda et al. [BDIR18] showed that t-out-of-n Shamir secret sharing over certain fields is local leakage-resilient if $t = n - o(n)$, and this has rate 1 and leakage-resilience rate roughly 1/4.

[1]The access structure of a secret sharing scheme is what we call the set of authorized subsets of parties.

Outline of Construction. We will now briefly describe the functioning of our compiler for the case of a t-out-of-n threshold secret sharing scheme, for simplicity. It makes use of a strong seeded randomness extractor Ext, which is an algorithm that takes two inputs – a seed s and a source w – and whose output $\mathrm{Ext}(s, w)$ is close to being uniformly random if s is chosen at random and w has sufficient entropy. The extractor being "strong" means that the output remains close to uniform even if the seed is given.

We take any threshold secret sharing scheme (such as Shamir's [Sha79]), and share our secret m with it to obtain the set of shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$. We first choose a uniform seed s, and for each $i \in [n]$, we choose a uniformly random source w_i (all of appropriate lengths), and mask Sh_i using $\mathrm{Ext}(s, w_i)$. That is, we compute $\mathsf{Sh}'_i = \mathsf{Sh}_i \oplus \mathrm{Ext}(s, w_i)$. We then secret share s using a 2-out-of-n secret sharing scheme to get the set of shares $S_1, \ldots, S_n$. The share corresponding to party i in our scheme is now set to $(w_i, \mathsf{Sh}'_i, S_i)$.

Given t such shares, to recover the secret, we first reconstruct the seed from any two S_i's and then unmask Sh'_i by XORing with $\mathrm{Ext}(s, w_i)$ to obtain Sh_i. We then use the reconstruction procedure of the underlying secret sharing to recover the message.

The correctness and privacy of the constructed scheme are straightforward to check. To argue the local leakage resilience of this construction, we go over a set of $n - t + 1$ hybrids where in each hybrid, we will replace one Sh_i with the all 0's string. Once we have replaced $n - t + 1$ such shares with the 0's string, we can then rely on the secrecy of the underlying secret sharing scheme to show that the message is perfectly hidden. Thus, it is now sufficient to show that any two adjacent hybrids in the above argument are statistically close. To argue that the adjacent hybrids, say Hyb_i and Hyb_{i+1}, are statistically close, we rely on the randomness property of the extractor. Note that even given bounded leakage from the source w_i, we can show that w_i has sufficient entropy so that the output of the extractor on the weak source w_i is statistically close to random. This allows us to argue that $\mathrm{Ext}(s, w_i)$ acts as a one-time pad and thus, we can replace Sh_i with the all 0's string.

However, in order to make the argument work, we must ensure that the leakage from the source is independent of the seed. This is where we will be using the fact that the seed is secret shared using a 2-out-of-n secret sharing scheme. In our reduction, we fix the share S_i to be independent of the seed and then leak from the source w_i. Once the seed is known[2], we can sample the other shares $(S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n)$ as a valid 2-out-of-n secret sharing of s that is consistent with the fixed share S_i. This allows us to argue that the leakage on w_i is independent of the source. There is a small caveat here that the masked value Sh'_i is dependent on the seed and hence we cannot argue independence of the leakage on the source and the seed. However, we use a simple trick of masking Sh'_i by another one-time pad and then secret share the one-time pad key along with the seed s and use this argue that this masked value is independent of the seed.

1.1 Related Work

In concurrent and independent work, Aggarwal et al. [ADN+18] also construct leakage-resilient secret sharing schemes for any monotone access structure (from any secret sharing scheme for that access structure), and use this to construct non-malleable secret sharing for general monotone access structures and threshold signatures that are resilient to leakage and mauling attacks. Their transformation loses a factor of n in the rate if a constant leakage-resilience rate is desired.

In another concurrent and independent work, Kumar et al. [KMS18] also consider the problem

[2]As the extractor is a strong seeded extractor, $\mathrm{Ext}(s, w_i)$ is statistically close to uniform even given the seed.

of obtaining leakage-resilient secret sharing scheme in a stronger leakage model. In particular, they consider a leakage model where every bit of the leakage can depend on the shares of an adaptive chosen set of $O(\log n)$ parties. They give constructions of such secret sharing schemes for general access structures via a connection to problems that have large communication complexity. The rate and the leakage-resilience rate of the construction of Kumar et al. are both $\Theta(1/\mathrm{poly}(n))$.

Local leakage resilient secret sharing (in the sense in which we use this term) was first studied by Goyal and Kumar [GK18] and Benhamouda et al [BDIR18] (independently of each other). Goyal and Kumar constructed a local leakage resilient 2-out-of-n threshold secret sharing scheme with rate and leakage-resilience rate both $\Theta(1/n)$. Benhamouda et al showed that, for large enough characteristic and large enough number of parties n, the Shamir secret sharing scheme is leakage-resilient (with leakage-resilience rate close to $1/4$) as long as the threshold is large (at least $n - o(\log(n))$). Badrinarayanan and Srinivasan [BS18] later constructed local leakage resilient t-out-of-n secret sharing schemes for any constant t that had rate $\Theta(1/\log(n))$ and leakage-resilience rate $\Theta(1/n\log(n))$.

Boyle et al. [BGK14] define and construct leakage-resilient verifiable secret sharing schemes where the sharing and reconstruction are performed by interactive protocols (as opposed to just algorithms). They also show that a modification of the Shamir secret sharing scheme satisfies a weaker notion of leakage-resilience than the one we consider here, where it is only required that a random secret retain sufficient entropy given the leakage on the shares.

Dziembowski and Pietrzak [DP07] construct secret sharing schemes (that they call intrusion-resilient) that are resilient to adaptive leakage where the adversary is allowed to iteratively ask for leakage from different shares. Their reconstruction procedure is also interactive, however, requiring as many rounds of interaction as the adaptivity of the leakage tolerated.

Leakage-resilience of secure multiparty computation has been studied in the past in various settings [BGJK12, GIM$^+$16, DHP11]. More broadly, leakage-resilience of various cryptographic primitives have been quite widely studied – we refer the reader to the survey by Alwen et al [ADW09] and the references therein.

2 Preliminaries

Notation. We use capital letters to denote distributions and their support, and corresponding lowercase letters to denote a sample from the same. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ and U_r denote the uniform distribution over $\{0,1\}^r$. For a finite set S, we denote $x \overset{\$}{\leftarrow} S$ as sampling x uniformly at random from the set S. For any $i \in [n]$, let x_i denote the symbol at the i-th co-ordinate of x, and for any $T \subseteq [n]$, let $x_T \in \{0,1\}^{|T|}$ denote the projection of x to the co-ordinates indexed by T. We write $\circ$ to denote concatenation.

For distributions D_1 and D_2 over the same domain S, we denote their statistical distance by $|D_1 - D_2|$. We will use the notation $D_1 \approx_\varepsilon D_2$ to denote that the statistical distance between D_1 and D_2 is at most ε. We now recall the definition of min-entropy, average conditional min-entropy [DORS08], and extractors.

Definition 2.1 (Min-Entropy) *The min-entropy of a random variable X is defined to be*

$$H_\infty(X) = \min_{s \in \mathrm{support}(X)} \{\log(1/\Pr[X = s])\}$$

Definition 2.2 (Average Conditional Min-Entropy [DORS08]) *For random variables (X, W), the average conditional min-entropy of X given W is defined as*

$$\widetilde{H}_\infty(X|W) = \log\left(E_{w \leftarrow W}\left[\max_x \Pr[X = x|W = w]\right]\right) = -\log E\left[2^{-H_\infty(X|W=w)}\right]$$

Definition 2.3 (Strong seeded extractor) *A function* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is called a strong seeded extractor for min-entropy k and error ε if for any (n,k)-source X and an independent uniformly random string U_d, we have*

$$|\mathrm{Ext}(X, U_d) \circ U_d - U_m \circ U_d| < \varepsilon,$$

where U_m is independent of U_d. Further if the function $\mathrm{Ext}(\cdot, u)$ *is a linear function over $\mathbb{F}_2$ for every $u \in \{0,1\}^d$, then* Ext *is called a linear seeded extractor.*

An average case seeded extractor requires that if a source X has average case conditional min-entropy $\widetilde{H}_\infty(X|Z) \geq k$ then the output of the extractor is uniform even when Z is given. We recall the following lemma from [DORS08] which states that every strong seeded extractor is also an average-case strong extractor.

Lemma 2.4 ([DORS08]) *For any $\delta > 0$, if* Ext *is a (k, ε)-strong seeded extractor then it is also a $\left(k + \log\left(\frac{1}{\delta}\right), \varepsilon + \delta\right)$ average case strong extractor.*

2.1 Secret Sharing Scheme

We first give the definition of a k-monotone access structure, then define a sharing function and finally define a secret sharing scheme.

Definition 2.5 (k-Monotone Access Structure) *An access structure $\mathcal{A}$ is said to be monotone if for any set $S \in \mathcal{A}$, any superset of S is also in $\mathcal{A}$. We will call a monotone access structure $\mathcal{A}$ as k-monotone if for any $S \in \mathcal{A}$, $|S| \geq k$.*

Definition 2.6 (Sharing Function [Bei11]) *Let $[n] = \{1, 2, \ldots, n\}$ be a set of identities of n parties. Let $\mathcal{M}$ be the domain of secrets. A sharing function* Share *is a randomized mapping from $\mathcal{M}$ to $\mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_n$, where $\mathcal{S}_i$ is called the domain of shares of party with identity i. For a set $T \subseteq [n]$, we denote* $\mathsf{Share}(m)_T$ *to be a restriction of* $\mathsf{Share}(m)$ *to its T entries.*

Definition 2.7 ($(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$-Secret Sharing Scheme [Bei11]) *Let $\mathcal{M}$ be a finite set of secrets, where $|\mathcal{M}| \geq 2$. Let $[n] = \{1, 2, \ldots, n\}$ be a set of identities (indices) of n parties. A sharing function* Share *with domain of secrets $\mathcal{M}$ is a $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$-secret sharing scheme with respect to monotone access structure $\mathcal{A}$ if the following two properties hold :*

- ***Correctness:*** *The secret can be reconstructed by any set of parties that are part of the access structure $\mathcal{A}$. That is, for any set $T \in \mathcal{A}$, there exists a deterministic reconstruction function* $\mathsf{Rec} : \otimes_{i \in T} \mathcal{S}_i \to \mathcal{M}$ *such that for every $m \in \mathcal{M}$,*

$$\Pr[\mathsf{Rec}(\mathsf{Share}(m)_T) = m] = 1 - \varepsilon_c$$

  where the probability is over the randomness of the Share *function.*

- ***Statistical Privacy:*** *Any collusion of parties not part of the access structure should have almost no information about the underlying secret. More formally, for any unauthorized set $U \subseteq [n]$ such that $U \notin \mathcal{A}$, and for every pair of secrets $m_0, m_1 \in M$, for any computationally unbounded distinguisher D with output in $\{0,1\}$, the following holds :*

$$|\Pr[D(\mathsf{Share}(m_0)_U) = 1] - \Pr[D(\mathsf{Share}(m_1)_U) = 1]| \leq \varepsilon_s$$

We define the rate of the secret sharing scheme as $\lim_{|m|\to\infty} \frac{|m|}{\max_{i\in[n]} |\mathsf{Share}(m)_i|}$

Remark 2.8 (Threshold Secret Sharing Scheme) *For ease of notation, we will denote a t-out-of-n threshold secret sharing scheme as $(t, n, \varepsilon_c, \varepsilon_s)$-secret sharing scheme.*

3 Leakage Resilient Secret Sharing Scheme

In this section, we will define and construct a leakage resilient secret sharing scheme against a class of local leakage functions. We first recall the definition of a leakage resilient secret sharing scheme from [GK18].

Definition 3.1 (Leakage Resilient Secret Sharing Scheme [GK18]) *An $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$ secret sharing scheme* $(\mathsf{Share}, \mathsf{Rec})$ *for message space $\mathcal{M}$ is said to be ε-leakage resilient against a leakage family $\mathcal{F}$ if for all functions $f \in \mathcal{F}$ and for any two messages $m_0, m_1 \in \mathcal{M}$:*

$$|f(\mathsf{Share}(m_0)) - f(\mathsf{Share}(m_1))| \leq \varepsilon$$

We will transform any secret sharing scheme to a leakage resilient secret sharing scheme against the local leakage function family. We first recall the definition of this function family.

Local Leakage Function Family. Let $\mathsf{Share} : \mathcal{M} \to \mathcal{S}_1 \times \mathcal{S}_2 \ldots \times \mathcal{S}_n$. We are interested in constructing leakage resilient secret sharing schemes against the specific function family $\mathcal{F}_{\mathcal{A},\mu} = \{f_{K,\vec{\tau}} : K \subseteq [n], K \notin \mathcal{A}, \tau_i : \mathcal{S}_i \to \{0,1\}^\mu\}$ where $f_{K,\vec{\tau}}$ on input $(\mathsf{share}_1, \ldots, \mathsf{share}_n)$ outputs share_i for each $i \in K$ in the clear and outputs $\tau_i(\mathsf{share}_i)$ for every $i \in [n] \setminus K$. Following [BDIR18], we will call secret sharing schemes resilient to $\mathcal{F}_{\mathcal{A},\vec{\tau}}$ as local leakage resilient secret sharing. We will define the leakage-resilience rate of such secret sharing schemes to be $\lim_{\mu\to\infty} \frac{\mu}{\max_{i\in[n]} |\mathsf{Share}(m)_i|}$.

Remark 3.2 *We remark that definition 3.1 is satisfiable against the leakage function class $\mathcal{F}_{\mathcal{A},\mu}$ (for any $\mu > 0$) only if the access structure is 2-monotone (see Definition 2.5). Hence, in the rest of the paper, we will concentrate on 2-monotone access structures.*

3.1 The Compiler

We will give a compiler that takes any $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$ secret sharing scheme for any 2-monotone $\mathcal{A}$ and outputs a local leakage resilient secret sharing scheme for $\mathcal{A}$. We give the description of the compiler in Figure 1.

Theorem 3.3 *Consider any 2-monotone access structure $\mathcal{A}$ and $\mu \in \mathbb{N}$ and a secret domain $\mathcal{M}$ with secrets of length m. Suppose for some $\eta, d, \rho \in \mathbb{N}$ and $\varepsilon_c, \varepsilon_s, \varepsilon \in [0,1)$, the following exist:*

Let $(\mathsf{Share}, \mathsf{Rec})$ be a $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$ secret sharing scheme for sharing secrets from $\mathcal{M}$ with share size equal to ρ bits. Let $(\mathsf{Share}_{(2,n)}, \mathsf{Rec}_{(2,n)})$ be a 2-out-of-n Shamir Secret sharing. Let $\mathrm{Ext} : \{0,1\}^\eta \times \{0,1\}^d \to \{0,1\}^\rho$ be a $(\eta - \mu, \varepsilon)$-average-case, strong seeded extractor.

```
LRShare : To share a secret $m \in \mathcal{M}$:
    1. Run $\mathsf{Share}(m)$ to obtain the shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$.
    2. Choose an uniform seed $s \xleftarrow{\$} \{0,1\}^d$ and a masking string $r \xleftarrow{\$} \{0,1\}^\rho$.
    3. For each $i \in [n]$ do:
        (a) Choose $w_i \xleftarrow{\$} \{0,1\}^\eta$.
        (b) Set $\mathsf{Sh}'_i = \mathsf{Sh}_i \oplus \mathrm{Ext}(w_i, s)$.
    4. Run $\mathsf{Share}_{(2,n)}(s, r)$ to obtain $S_1, \ldots, S_n$.
    5. Output $\mathsf{share}_i$ as $(w_i, \mathsf{Sh}'_i \oplus r, S_i)$.

LRRec : Given the shares $\mathsf{share}_{j_1}, \mathsf{share}_{j_2}, \ldots, \mathsf{share}_{j_\ell}$ where $K = \{j_1, \ldots, j_k\} \in \mathcal{A}$ do:
    1. For each $i \in K$, parse $\mathsf{share}_i$ as $(w_i, S'_i, S_i)$.
    2. Run $\mathsf{Rec}_{(2,n)}(S_{j_1}, S_{j_2})$ to recover $(s, r)$
    3. For each $i \in K$ do:
        (a) Compute $\mathsf{Sh}'_i = S'_i \oplus r$.
        (b) Recover $\mathsf{Sh}_i$ by computing $\mathsf{Sh}'_i \oplus \mathrm{Ext}(w_i, s)$.
    4. Run $\mathsf{Rec}(\mathsf{Sh}_{j_1}, \ldots, \mathsf{Sh}_{j_k})$ to recover the secret $m$.
```

Figure 1: Local Leakage-Resilient Secret Sharing

- *A $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$ secret sharing scheme for the secret domain $\mathcal{M}$ with share length ρ.*
- *A $(\eta - \mu, \varepsilon)$-average-case strong seeded extractor* $\mathrm{Ext} : \{0,1\}^\eta \times \{0,1\}^d \to \{0,1\}^\rho$.

Then, the construction in Figure 1, when instantiated with these, is a $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$ secret sharing scheme for $\mathcal{M}$ that is $2(\varepsilon_s + n \cdot \varepsilon)$-leakage resilient against $\mathcal{F}_{\mathcal{A},\mu}$. It has share size $(\eta + 2\rho + d)$.

We leave the proof of Theorem 3.3 to the full version of this paper [SV18].

3.2 Instantiation

Using state-of-the-art explicit constructions of strong seeded extractors from the work of Guruswami, Umans, and Vadhan [GUV09] (along with Lemma 2.4), we get the following corollary that represents the best asymptotics our construction achieves.

Corollary 3.4 *If there exists a secret sharing scheme for a 2-monotone access structure $\mathcal{A}$ with rate R, then there exists an ε-local leakage resilient secret sharing for $\mathcal{A}$ for some negligible ε with rate $R/3.01$ and leakage-resilience rate 1.*

Note that here, negligible means negligible in the length of the secret m and the amount of leakage μ. For the special case of threshold secret sharing scheme for which we know constructions with rate 1 [Sha79], we obtain the following corollary.

Corollary 3.5 *For any $n, t \in \mathbb{N}$ such that $t \leq n$ and some negligible ε, there is an ε-local leakage resilient t-out-of-n threshold secret sharing scheme with rate $\Omega(1)$, and leakage-resilience rate* 1.

A Simpler Scheme. We next present an alternative instantiation of our LRSS scheme using the inner product function over finite fields as the extractor. While this instantiation has asymptotically worse rate and leakage-resilience rate than the one above, it is also much simpler, computationally more efficient, and easier to analyze for concrete parameters. Details regarding the performance of our implementation of this scheme for threshold secret sharing are presented in Section 4.

Let $\mathbb{F}$ be a finite field, and $IP : \mathbb{F}^\eta \times \mathbb{F}^\eta \to \mathbb{F}$ be the inner product function given by:

$$IP((x_1, \ldots, x_\eta), (s_1 \ldots, s_\eta)) = \sum_i x_i s_i$$

The leftover hash lemma implies that IP is a (k, ε) strong seeded extractor for any k and ε such that $\log |\mathbb{F}| < k - 2\log(1/\varepsilon)$ (see, for instance, Theorem 6.18 and its proof in [Vad12]; see also [CG88]). Note that here, the source of the extractor is in $\mathbb{F}^\eta$, and its output is ε-close to the uniform distribution over $\mathbb{F}$.

We now instantiate the construction in Figure 1 as follows (note that below, μ is still the number of bits leaked, while η is the number of field elements):

- We pick $\mathbb{F}$ that is larger than both the message space and the number of shares n.
- We use Shamir secret sharing over the field $\mathbb{F}$ as the underlying secret sharing scheme. Without loss of generality, we treat the secrets to be shared as elements of $\mathbb{F}$.
- Given μ and ε, we set η so that the above inner product is a $(\eta \cdot \log|\mathbb{F}| - \mu, \varepsilon/2n)$-average-case strong seeded extractor. Following Lemma 2.4, this can be done by setting it so that the inner product is a $(\eta \cdot \log |\mathbb{F}| - \mu - \log(4n/\varepsilon), \varepsilon/4n)$-strong seeded extractor. That is, we set η to be $\left\lceil 1 + \frac{\mu}{\log|\mathbb{F}|} + \frac{3\log(4n/\varepsilon)}{\log|\mathbb{F}|} \right\rceil$.

We also make the following modifications in order for the construction to be compatible with our extractor:

- The seed s and the w_i's are now drawn at random from $\mathbb{F}^\eta$, and the masking string r is a random element of $\mathbb{F}$.
- All XOR operations are replaced with addition over $\mathbb{F}$.

The share size above is $(2\eta + 2)\log|\mathbb{F}|$. This gives us the following corollary.

Corollary 3.6 *For any $n, t \in \mathbb{N}$ such that $t \leq n$ and $\varepsilon \in (0, 1)$, the construction in Figure 1, when instantiated with the inner product function (and modified) as above, is an ε-local leakage resilient t-out-of-n threshold secret sharing scheme that has rate $\Omega(1)$, and leakage-resilience rate* $1/2$.

4 Performance Evaluation

We implement our leakage resilient secret sharing scheme instantiated with Shamir secret sharing scheme [Sha79] and the inner product function. In this section, we report on the performance of the instantiation. We will evaluate the (multiplicative) computational overhead and storage overhead of our scheme compared to the Shamir secret sharing scheme in different settings.

4.1 Implementation Details

We choose $\mathbb{F} = \mathbb{Z}_p$ for a prime p with bit length 128 as the finite field for Shamir secret sharing scheme, namely $\log |\mathbb{F}| = 128$. Secrets to be shared are elements in the field, of length 128. We fix the statistical distance $\varepsilon = 2^{-80}$. The big number arithmetic operations are implemented using the Relic library [AG]. All the experiments are run on a machine with 4-core 2.9 GHz CPU and 16 GB of RAM.

n	η	$\lvert\mathsf{share}\rvert$	μ	f	Storage Overhead
2	3	1024	≤ 7	$\leq 0.6\%$	8
	4	1280	≤ 135	$\leq 10.5\%$	10
	5	1536	≤ 263	$\leq 17.1\%$	12
	6	1792	≤ 391	$\leq 21.8\%$	14
	9	2560	≤ 775	$\leq 30.2\%$	20
	19	5120	≤ 2055	$\leq 40.1\%$	40
	39	10240	≤ 4615	$\leq 45.1\%$	80
	197	50688	≤ 24839	$\leq 49.0\%$	396
100	4	1280	≤ 118	$\leq 9.2\%$	10
	5	1536	≤ 246	$\leq 16.0\%$	12
	6	1792	≤ 374	$\leq 20.8\%$	14
	10	2816	≤ 886	$\leq 31.5\%$	22
	20	5376	≤ 2166	$\leq 40.3\%$	42
	40	10496	≤ 4726	$\leq 45.0\%$	82
	204	52480	≤ 25718	$\leq 49.0\%$	410

Table 1: Settings of leakage fraction of the secret shares. μ is the number of bits of leakage the scheme is resilient to, and f is μ as a percentage of the share size.

We evaluated the runtime for our leakage resilient secret sharing scheme for various settings of (n, t) and leakage fraction of the secret shares. Table 1 listed different settings of the secret share leakage, where η is the number of field elements in the inner product function (the minimum number of which is 3), $|\mathsf{share}|$ is the size of a single secret share in bits, μ is the maximal allowed number of leaked bits of every secret share, and f is the maximal allowed leakage fraction of the secret shares (that is, $\frac{\mu}{|\mathsf{share}|}$). For the purpose of illustration we only listed the settings for $n = 2$ and $n = 100$ in the table. The threshold t does not affect the values, so we omit that in the table.

Concretely, we can compute the size of a secret share $|\mathsf{share}| = 256\eta + 256$, which grows linearly in η. The maximal number of leaked bits is $\mu = 128\eta - 3 \cdot \log(2^{82} \cdot n) - 128$, which also grows linearly in η for a fixed n. The maximal allowed leakage fraction is

$$f = \frac{128\eta - 3 \cdot \log(2^{82} \cdot n) - 128}{256\eta + 256} = \frac{\mu}{|\mathsf{share}|} = \frac{1}{2} - \frac{3 \cdot \log(2^{82} \cdot n) + 256}{256\eta + 256},$$

which grows hyperbolically in η for a fixed n and approaches $\frac{1}{2}$ when η goes to infinity.

In our experiments, we will set the maximal allowed leakage fraction f to be the following values: 0.1%, 1%, 10%, 20%, 30%, 40%, 45%, 49%, and set η correspondingly in different settings of (n,t). We will actually control the parameter η in the experiments.

Storage Overhead As mentioned above, the size of a secret share of our scheme is $|\mathsf{share}| = 256\eta + 256$ bits. A secret share of Shamir secret sharing scheme is a field element of 128 bits. Hence the storage overhead of secret shares of our scheme compared with Shamir is

$$\text{Storage Overhead} = \frac{256\eta + 256}{128} = 2\eta + 2.$$

The concrete storage overhead in different settings are also listed in Table 1.

4.2 Computational Overhead

In this section we evaluate the performance of our leakage resilient secret sharing scheme by analyzing the multiplicative computational overhead compared to the Shamir secret sharing scheme in different settings of (n,t,f). We will show how the computational overhead changes with each parameter.

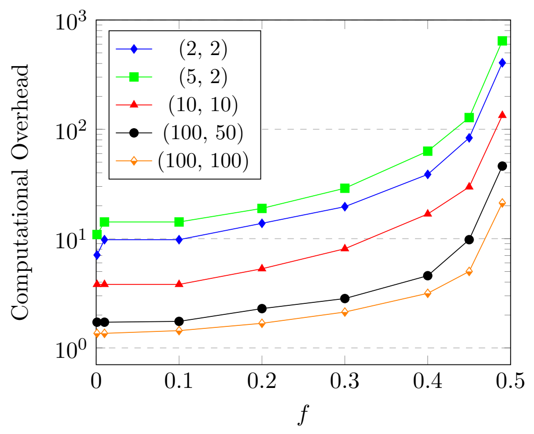
4.2.1 Overhead Growth with f, η, and μ

(n,t)	Shamir	$f=0.1\%$	$f=1\%$	$f=10\%$	$f=20\%$	$f=30\%$	$f=40\%$	$f=45\%$	$f=49\%$
(2, 2)	4.16 μs	7.08	9.78	9.78	13.8	19.6	38.7	83.5	406
(5, 2)	4.74 μs	10.9	14.2	14.2	18.9	28.9	63.1	128	644
(5, 3)	8.04 μs	6.52	9.27	9.27	13.4	19.7	41.7	81.0	414
(10, 2)	6.70 μs	18.8	18.8	18.8	26.8	40.7	82.7	172	822
(10, 5)	19.5 μs	7.51	7.51	7.51	9.55	17.1	33.3	61.6	300
(10, 10)	39.2 μs	3.81	3.81	3.81	4.89	8.08	16.8	29.7	134
(100, 2)	41.4 μs	23.6	23.6	26.1	38.2	74.1	138	292	1319
(100, 50)	1.13 ms	1.72	1.72	1.75	2.29	2.83	4.58	9.78	46.1
(100, 100)	2.27 ms	1.36	1.36	1.44	1.68	2.13	3.16	5.01	21.2

Table 2: Runtime of Shamir secret sharing scheme and computational overhead of our scheme when (n,t) are fixed and f is increasing.

Table 2 shows the runtime of Shamir secret sharing for generating shares of a single secret in different settings of (n,t). For each setting of (n,t), we evaluate the computational overhead of our scheme in generating secret shares compared with Shamir, and show overhead growth with f. The reported time is an average of 10,000 secret share generation.

Figure 2 plotted the computational overhead growth with f in different settings of (n,t). As shown in the figure, the overhead grows rapidly with f especially when $f \geq 40\%$ in all settings. We plotted another graph showing the overhead growth with η and it is roughly linear. From Section 4.1 we know that μ grows linear in η, hence the overhead also grows linear in μ. Since f grows hyperbolically in η, the overhead also grows hyperbolically in f.



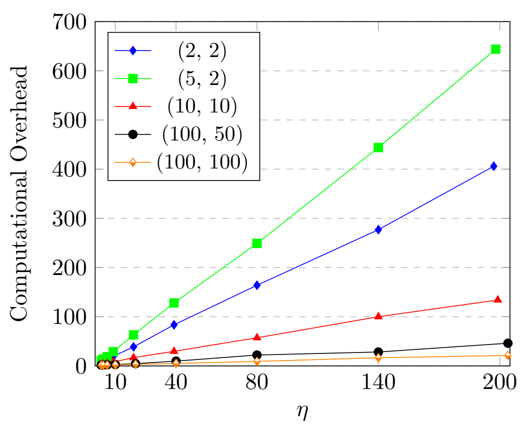


Figure 2: Computational overhead growth with f and η in different settings of (n, t). The figure on the left has logarithmic scale on y-axis.

(n, f)	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$	$t = 9$	$t = 10$
(10, 0.1%)	16.8	10.1	8.61	7.51	5.76	4.62	4.47	4.31	3.81
(10, 20%)	26.8	15.5	12.2	9.55	8.16	7.39	6.93	6.20	5.30
(10, 49%)	822	553	395	300	241	209	188	160	134

(n, f)	$t = 2$	$t = 3$	$t = 5$	$t = 7$	$t = 10$	$t = 20$	$t = 30$	$t = 50$	$t = 70$	$t = 100$
(100, 0.1%)	23.6	15.3	8.55	6.31	4.55	3.07	2.10	1.72	1.66	1.36
(100, 20%)	38.2	26.1	16.5	10.2	7.36	5.10	3.01	2.29	1.79	1.68
(100, 49%)	1319	828	524	341	205	117	77.9	46.1	34.0	21.2

Table 3: Computational overhead of our scheme when (n, f) are fixed and t is increasing.

4.2.2 Overhead Growth with t

Table 3 shows the computational overhead of our scheme in generating secret shares compared with Shamir secret sharing scheme. For different settings of (n, f), we show the overhead growth with the threshold t.

As plotted in Figure 3, the overhead decreases rapidly when t increases in all settings. In fact, we can compute the asymptotic computational overhead of our scheme compared with Shamir by counting the total number of multiplications in both schemes. Let $T(n, t)$ be the runtime of t-out-of-n Shamir secret sharing scheme. Then we have

$$\begin{aligned}\text{Computational Overhead} =& \frac{T(n,t) + (\eta+1)\cdot T(n,2) + \eta\cdot n}{T(n,t)} = 1 + \frac{(\eta+1)\cdot T(n,2) + \eta\cdot n}{T(n,t)} \\ =& 1 + \frac{(\eta+1)\cdot 2n + \eta\cdot n}{n\cdot t} = 1 + \frac{3\eta+2}{t}.\end{aligned}$$

Therefore the overhead approaches 1 when t goes to infinity in any setting of (n, f).

4.2.3 Overhead Growth with n

Table 4 shows the computational overhead of our scheme when f and the fraction of threshold t/n are fixed and the number of parties n is increasing. As plotted in Figure 4, the overhead

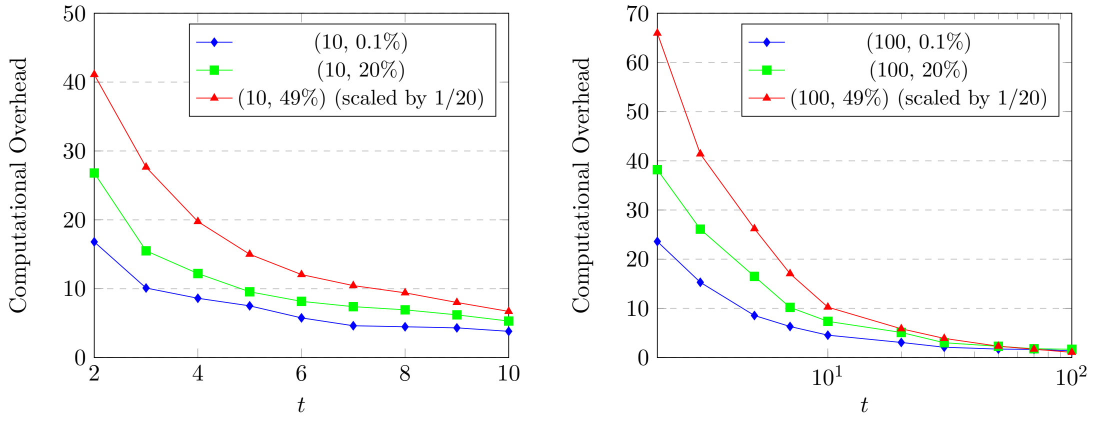


Figure 3: Computational overhead growth with t in different settings of (n, f). The figure on the right has logarithmic scale on x-axis.

$(t/n, f)$	$n = 20$	$n = 30$	$n = 40$	$n = 60$	$n = 100$	$n = 200$	$n = 500$	$n = 1000$
(1/10, 0.1%)	18.8	13.8	9.52	7.85	5.12	3.42	2.02	1.46
(1/10, 20%)	37.0	21.5	16.8	12.4	6.71	4.56	2.14	1.72
(1/10, 49%)	997	662	548	394	202	87.4	24.3	10.1

$(t/n, f)$	$n = 4$	$n = 6$	$n = 8$	$n = 12$	$n = 20$	$n = 40$	$n = 100$	$n = 200$
(1/2, 0.1%)	10.1	7.90	6.06	4.34	3.46	2.91	1.72	1.38
(1/2, 20%)	18.3	13.1	11.6	9.01	6.49	4.45	2.29	1.71
(1/2, 49%)	577	444	373	297	183	117	46.1	23.7

$(t/n, f)$	$n = 2$	$n = 3$	$n = 4$	$n = 6$	$n = 10$	$n = 20$	$n = 50$	$n = 100$
(1, 0.1%)	7.08	6.31	5.52	4.31	3.81	2.45	1.77	1.36
(1, 20%)	13.8	10.7	8.91	7.28	4.89	3.86	2.44	1.68
(1, 49%)	406	327	275	214	134	92.9	51.8	21.2

Table 4: Computational overhead of our scheme when $(t/n, f)$ are fixed and n is increasing.

decreases rapidly as n increases in all settings. Recall that the computational overhead of our scheme approaches 1 when t goes to infinity and the overhead is asymptotically independent of n. When t/n is fixed and n increases, t also increases, hence the overhead approaches 1 as n goes to infinity.

References

[ADN$^+$18] Divesh Aggarwal, Ivan Damgard, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, Jo ao Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret-sharing schemes for general access structures. Manuscript, accessed via personal communication, 2018.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Survey: Leakage resilience and the bounded retrieval model. In Kaoru Kurosawa, editor, *Information Theoretic Security, 4th International Conference, ICITS 2009, Shizuoka, Japan, December 3-6, 2009. Re-*

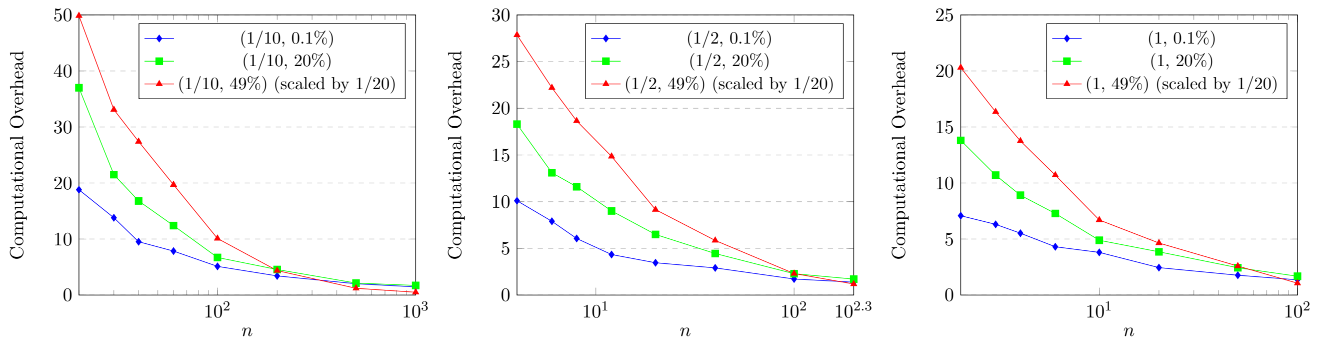


Figure 4: Computational overhead growth with n in different settings of $(t/n, f)$. All the figures have logarithmic scale on x-axis.

vised Selected Papers, volume 5973 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[AG] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, Heidelberg, March 2009.

[BDIR18] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 531–561. Springer, Heidelberg, August 2018.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, pages 11–46, 2011.

[BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1235–1254. ACM Press, May 2012.

[BGK14] Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. *Distributed Computing*, 27(3):147–164, 2014.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.

[BS18] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. Manuscript, accessed via personal communication, 2018.

[CCD88] David Chaum, Claude Crepeau, and Ivan Damgaard. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th Annual ACM Symposium on Theory of Computing*, pages 522–533. ACM Press, May 1994.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, Heidelberg, August 1990.

[DHP11] Ivan Damgard, Carmit Hazay, and Arpita Patra. Leakage resilient secure two-party computation. Cryptology ePrint Archive, Report 2011/256, 2011. `http://eprint.iacr.org/2011/256`.

[DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38:97–139, 2008.

[DP07] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *48th Annual Symposium on Foundations of Computer Science*, pages 227–237. IEEE Computer Society Press, October 2007.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302. IEEE Computer Society Press, October 2008.

[Fra90] Yair Frankel. A practical protocol for large group oriented networks. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 56–61. Springer, Heidelberg, April 1990.

[FRR$^+$10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, Heidelberg, May / June 2010.

[GIM$^+$16] Vipul Goyal, Yuval Ishai, Hemanta K. Maji, Amit Sahai, and Alexander A. Sherstov. Bounded-communication leakage resilience via parity-resilient circuits. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 1–10. IEEE Computer Society Press, October 2016.

[GK18] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 685–698, 2018.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), 2009.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, Heidelberg, August 2003.

[KGG$^+$18] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *CoRR*, abs/1801.01203, 2018.

[KMS18] Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing. *IACR Cryptology ePrint Archive*, 2018:1138, 2018.

[LSG$^+$18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *CoRR*, abs/1801.01207, 2018.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, Heidelberg, February 2004.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, Heidelberg, August 2009.

[Rot12] Guy N. Rothblum. How to compute under $\mathcal{AC}^0$ leakage without secure hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569. Springer, Heidelberg, August 2012.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[SV18] Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. *IACR Cryptology ePrint Archive*, 2018:1154, 2018.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.