Leakage Resilient Secret Sharing and Applications[⋆]

Akshayaram Srinivasan and Prashant Nalini Vasudevan

University of California, Berkeley
{akshayaram,prashvas}@berkeley.edu

Abstract. A secret sharing scheme allows a dealer to share a secret among a set of n parties such that any authorized subset of the parties can recover the secret, while any unauthorized subset 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 every other share. 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 secret sharing schemes are for threshold access structures for very low ($O(1)$) or very high ($n - o(\log n)$) thresholds.

In this work, we give a compiler that takes a secret sharing scheme for any monotone access structure and produces a local leakage resilient secret sharing scheme for the same access structure, with only a constant-factor asymptotic blow-up in the sizes of the shares. Furthermore, the resultant secret sharing 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. Using this secret sharing scheme as the main building block, we obtain the following results:

- **Rate Preserving Non-Malleable Secret Sharing.** We give a compiler that takes any secret sharing scheme for a 4-monotone access structure[1] with rate R and converts it into a non-malleable secret sharing scheme for the same access structure with rate $\Omega(R)$. The previous such non-zero rate construction (Badrinarayanan and Srinivasan, EUROCRYPT '19) achieved a rate of $\Theta(R/t_{\max} \log^2 n)$, where $t_{\max}$ is the maximum size of any minimal set in the access structure. As a special case, for any threshold $t \geq 4$ and an arbitrary $n \geq t$, we get the first constant-rate construction of t-out-of-n non-malleable secret sharing.
- **Leakage-Tolerant Multiparty Computation for General Interaction Patterns.** For any function f, we give a reduction from

[⋆] Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for LongTerm Cybersecurity (CLTC, UC Berkeley).

[1] A 4-monotone access structure has the property that any authorized set has size at least 4.

constructing a leakage-tolerant secure multi-party computation protocol for computing f that obeys any given interaction pattern to constructing a secure (but not necessarily leakage-tolerant) protocol for a related function that obeys the star interaction pattern. Together with the known results for the star interaction pattern, this gives leakage tolerant MPC for any interaction pattern with statistical/computational security. This improves upon the result of (Halevi et al., ITCS 2016), who presented such a reduction in a leak-free environment.

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; while any subset of parties that is not authorized do not learn anything about the secret from their shares. Secret sharing has had widespread applications across cryptography, ranging from secure multiparty computation [GMW87, BGW88, CCD88] and threshold cryptographic systems [DF90, Fra90, DDFY94] to 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. 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 leakage-resilience in secret sharing – we ask that 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.

The Leakage Model. A secret sharing scheme consist of a sharing algorithm, which takes a secret and shares it into a set of shares, and a reconstruction algorithm, which takes some subset of these shares and reconstructs the secret from it. In this work, we do not deal with the leakage from the machines that run these procedures. Instead, the leakage that we care about is that which could happen from the machines that these shares are stored on after they have been generated, and the sharing and reconstruction are assumed to be leak-free.

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, performs an arbitrary (possibly inefficient) computation 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 each of the remaining shares. This form of leakage-resilience for secret sharing was formalized in recent work by Goyal and Kumar [GK18a], and Benhamouda, Degwekar, Ishai and Rabin [BDIR18].

This leakage model may be seen as an adaptation of the "memory attacks" model introduced by Akavia, Goldwasser, and Vaikuntanathan [AGV09] to the context of secret sharing. In this model, the basic axiom is that everything that is stored in the memory is subject to leakage, and the only restriction is that the leakage function must be shrinking. This model was introduced as an alternative to the well-studied "Only Computational Leaks" (OCL) model [MR04] (which we do not consider in this work) in order to capture known real-world attacks that were not captured by the OCL model. A notable example of such an attack is the cold-boot attack by Halderman et al. [HSH$^+$09], which showed measures to leak a significant fraction of the bits of a secret if it was ever stored in a part of memory which could be accessed by an adversary (e.g. DRAM). The definition of leakage-resilience for secret sharing that we work with is intended (as was the memory attacks model) to protect against such attacks on the machines that store the shares after they have been generated.

Goyal and Kumar, and Benhamouda et al, showed constructions of leakage-resilient threshold secret sharing schemes (where subsets above a certain size are authorized) for certain thresholds. They then showed how such schemes could be used to construct 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 of leakage-resilient secret sharing 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[2] (and 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.

We present a construction of leakage-resilient secret sharing that is near-optimal in terms of the above parameters, and show applications of our construction to constructing constant-rate non-malleable secret sharing schemes and leakage-tolerant multi-party computation protocols.

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

1.1 Our Results and Techniques

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

Informal Theorem 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-resilience rate 1. The only constructions of local leakage resilient secret sharing known before our work were for threshold access structures with either very small or very large thresholds. Goyal and Kumar [GK18a] 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 [BS19], 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 $1/4$.

Outline of our Compiler. 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 min-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. The key here is that as long as the leakage from the source w_i is much smaller than its length, it still has enough entropy for the output of the extractor on w_i to be 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 without an adversary being able to tell.

However, in order to make the argument work, we must ensure that the leakage from the source is independent of the seed (which is required for the extractor to work). This is where we will be using the fact that the seed is secret shared using a 2-out-of-n secret sharing scheme. Intuitively, this ensures that a local leakage function has no idea what the seed is, and so cannot leak anything about w_i that depends on the seed. 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[3], 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 seed. 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.

This construction described above has several useful properties. The most significant one is that the transformation is rather simple and only incurs a very small overhead when compared to the original secret sharing scheme. In particular, the rate of the resultant leakage resilient secret sharing has only a small constant factor loss when compared to the initial secret sharing scheme. Also, we can sample the seed s of the extractor once and use it for sharing multiple secrets.[4] The second advantage is that it easily generalizes to all monotone access structures, basically, the only difference is that we use a secret sharing scheme for this access structure to obtain the set of shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$, and the rest of the steps are exactly the same as before. The third advantage is that the resultant secret sharing scheme has optimal leakage-resilience rate, i.e., the ratio between the number of bits of leakage tolerated and size of the share tends to 1 as the amount of leakage that the scheme is designed to handle increases. Finally, if we use the inner product two-source extractor of Chor and Goldreich [CG88] as the underlying extractor and the Shamir secret sharing scheme, then the sharing procedure is a linear function of the secret and a quadratic function of the randomness, and this can be implemented very efficiently.

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

[4] For the security of this modification to go through, we need the adversary to specify all the secrets and leakage functions upfront – it cannot adaptively choose the secrets and leakage functions depending on the previous leakage.

Stronger Leakage-Resilience. We also extend our construction to satisfy a stronger notion of leakage resilience, which we describe next. In the earlier definition of local leakage, the leakage functions that are applied on the shares of honest parties are required to be specified independently of the shares that are completely revealed to the adversary. In our stronger definition, these leakage functions are allowed to depend on some number of the adversary's shares.

In particular, we construct t-out-of-n threshold secret sharing schemes that are resilient to such stronger leakage where the adversary is given $(t-1)$ shares, and the leakage functions applied on the honest party's shares are allowed to depend on $(t-2)$ of these shares. This construction, which is in fact a simple modification of our earlier one, has worse rate, but still has optimal leakage-resilience rate. Referring temporarily to the above as $(t-2, t-1)$-strong local leakage, we have the following.

Informal Theorem 2 *For any $t \leq n$, there is a t-out-of-n threshold secret sharing scheme that is resilient against $(t-2, t-1)$-strong local leakage, has rate $\Omega(1/n)$, and leakage-resilience rate tending to* 1.

It is easy to check that this definition is impossible to achieve for a t-out-of-n threshold secret sharing scheme if we allow the leakage functions to depend on all $(t-1)$ of the adversary's shares, as the leakage function on any honest party's share can use the $(t-1)$ shares along with this share to reconstruct the secret and leak a few bits of the secret. Later in this section, we will describe an application of this strong leakage resilient secret sharing scheme in constructing leakage tolerant MPC for general interaction patterns.

Application 1: Rate-Preserving Non-Malleable Secret Sharing. Non-malleable secret sharing schemes, introduced by Goyal and Kumar [GK18a], are secret sharing schemes where it is not possible to tamper with the shares of a secret s (in certain limited ways) so as to convert them to shares corresponding to a different secret $\widetilde{s}$ that is related to s (such as $s+1$ or s with the first bit flipped). We are interested in security against an adversary that tampers each share independent of the others (called individual tampering). Such an adversary works as follows. Initially, it specifies n "tampering functions" $f_1, \ldots, f_n$ and an authorized set. A secret s is then shared into $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$ and the shares are tampered to get $\widetilde{\mathsf{Sh}}_i \leftarrow f_i(\mathsf{Sh}_i)$. The requirement now is that if the above specified authorized set of parties try to reconstruct the secret using the shares $\{\widetilde{\mathsf{Sh}}_i\}$, the resulting secret $\widetilde{s}$ is either the same as s or something completely independent.

In this setting, Goyal and Kumar presented a construction of a non-malleable t-out-of-n threshold secret sharing scheme, and in a later paper [GK18b] extended this to general access structures. Their constructions, however, had an asymptotic rate of zero.

Badrinarayanan and Srinivasan [BS19] gave a rate-efficient compiler that takes any secret sharing scheme for a 4-monotone[5] access structure and outputs

[5] k-monotone means that all authorized sets in the access structure are of size at least k.

a non-malleable secret sharing scheme for the same access structure. The main tool used in their compiler was a local leakage resilient threshold secret sharing scheme. The loss in the rate of the resulting non-malleable secret sharing scheme depended on the parameters of the underlying local leakage resilient secret sharing. In particular, to have only a constant loss in the rate, it was important to have a local leakage resilient threshold secret sharing scheme that had a constant rate and a constant leakage-resilience rate. We plug in our leakage resilient secret sharing scheme that has both these features with the compiler of Badrinarayanan and Srinivasan to obtain a rate-preserving compiler for non-malleable secret sharing.

Informal Theorem 3 *There is a compiler that, given a secret sharing scheme for a* 4*-monotone access structure* $\mathcal{A}$ *with rate* R*, produces a secret sharing scheme for* $\mathcal{A}$ *that has rate* $\Omega(R)$ *and is non-malleable against individual tampering.*

Application 2: Leakage-Tolerant MPC for General Interaction Patterns. Next, we provide an application of our constructions to secure multiparty computation (MPC), an area where secret sharing is rather pervasive. In particular, we study MPC protocols obeying a specified interaction pattern.

Background. An interaction pattern (introduced by Halevi et al [HIJ$^+$16]) generalizes the communication graph of a standard MPC protocol. It is defined as a directed graph which specifies the sequence of messages that have to be sent during the execution of a MPC protocol – its vertices correspond to the messages, and edges indicate dependencies between messages. We illustrate by example with the ring interaction pattern. Here, the first message is sent by the party P_1 to the party P_2 and depending on this message, P_2 sends a message to P_3 and so on. Finally, the party P_n sends a message to P_1 who computes the output based on this message. The directed graph corresponding to this has $(n + 1)$ nodes, one corresponding to each message and one for the output, and the graph is a single directed path that goes from the first message to the last and then to the output node. To give another example, a standard 2-round MPC protocol with n parties can be represented by an interaction pattern graph with two sets of $\binom{n}{2}$ nodes, representing the messages sent by each party to every other party in the two rounds. The edges then go from the nodes corresponding to first-round messages to second-round messages, according to the protocol.

Given an interaction pattern specified by such a directed graph, the main goal is to understand which functions can be computed securely by a protocol following this pattern. It is known that without any form of correlated randomness setup, even simple functions such as majority cannot be computed with any meaningful form of security for certain interaction patterns [BGI$^+$14]. It is also known from a sequence of works [HLP11, GGG$^+$14, BGI$^+$14] that standard notions of security in MPC that guarantee that only the output is leaked are impossible to achieve for certain interaction patterns. To see this, consider the star interaction pattern [FKN94] where there is a special party called the evaluator and every other party sends a single message to the evaluator who then computes the output. In this interaction pattern, if the evaluator colludes with some subset of the parties, then it is easy to see that the colluding parties can learn the entire residual function resulting from fixing the honest parties' inputs to the function being computed.

In other interaction patterns, the residual function that the colluding parties are able to learn may be different. In general, Halevi et al. [HIJ$^+$16] classify the parties' inputs into *fixed* and *free* – every honest party's input is fixed, and so is a corrupted party's input if there exists a path from a message sent by the corrupted party to the output that passes through at least one honest party's message. The inputs of the remaining corrupted parties are free. To capture the inherent security loss in certain interaction patterns, Halevi et al. allow the adversary to learn the residual function with the above set of fixed inputs, and say a protocol that is compliant with an interaction pattern is secure if it hides everything other than this residual function.

Defining Leakage Tolerance. We extend the above definition of security to also account for possible leakage from the states of honest parties. Specifically, we define the notion of leakage tolerance for an MPC protocol that is compliant with an interaction pattern along the same lines as that of leakage tolerant MPC [GJS11, BCH12]. In the setting of leakage tolerance, as in the standard setting, we consider an adversary who corrupts an arbitrary subset of parties and can see their entire views. But in addition to this, the adversary also obtains bounded leakage on the complete internal state – that includes the correlated randomness, the input, the secret randomness, and the entire view of the protocol – of every honest party. The only process that we assume happens in a leak-free manner is the correlated randomness generation phase which is anyway independent of the actual inputs of the parties. After this leak-free randomness generation, every bit of an honest party's secret state including its input is subject to leakage. Here, the adversary can potentially learn bounded information about the honest party's input since it has access to all of the honest parties' secret state. We would like to guarantee that nothing beyond such bounded information about the inputs and the residual function is actually leaked to the adversary – note that this is the best possible security we can hope for in this setting. Technically, we account for this leakage by allowing the simulator to learn the same amount of information about the honest parties' inputs.

What makes the task of providing such security non-trivial is that, unlike a standard MPC simulator who is allowed to cheat in generating the protocol messages, a simulator in the leakage tolerance setting cannot deviate from the protocol specification. This is because any deviation can be caught by the adversary by leveraging the leakage on the secret state of the honest party. At first sight, the task of designing such a simulator seems impossible as we require the simulator to generate the correct protocol messages based only the output (or more generally, based on the residual function). However, notice that the leakage functions are local to the honest party's view. Hence, the simulator must follow the protocol correctly at the local level but must somehow cheat at the global level, i.e., in generating the joint distribution of the protocol messages. To make this task even more demanding, we do not wish to use any computational assumptions and only make use of information theoretic tools to achieve leakage tolerance.

Our Results. In this setting, we upgrade one of the results of Halevi et al [HIJ$^+$16] to have the additional guarantee of leakage tolerance. They showed that the star interaction pattern described earlier is complete for obtaining MPC for general interaction patterns – given a secure protocol for a function f that is compliant with the star interaction pattern, they showed how to construct a secure protocol for f compliant with any other interaction pattern. In this work, we show that star interaction pattern is complete for obtaining leakage-tolerant MPC for general interaction patterns. Specifically, we obtain the following.

Informal Theorem 4 *There is a compiler that, given a function $f : \{0,1\}^n \to \{0,1\}$, an interaction pattern $\mathcal{I}$, and a secure protocol for f compliant with the star interaction pattern, produces a secure protocol (with a leak-free setup phase producing correlated randomness) for f compliant with $\mathcal{I}$ that is leakage tolerant.*

Using the known protocols for the star interaction pattern [BGI$^+$14, BKR17, GGG$^+$14], we obtain the following corollaries for any interaction pattern $\mathcal{I}$ and function $f : \{0,1\}^n \to \{0,1\}$:

- An $\mathcal{I}$-compliant protocol for f with statistical leakage tolerance against upto $(n-1)$ passive corruptions, with communication exponential in n.
- An efficient $\mathcal{I}$-compliant protocol for $f \in \mathsf{NC}^1$ with statistical leakage tolerance against a constant number of passive corruptions.
- Assuming the existence of one-way functions, and that f is computable by a polynomial-sized circuit, an efficient $\mathcal{I}$-compliant protocol for f with computational leakage tolerance against a constant number of passive corruptions.
- Assuming the existence of indistinguishability obfuscation and one-way functions, and that f is computable by a polynomial-sized circuit, an efficient $\mathcal{I}$-compliant protocol for f with computational leakage tolerance against upto $(n-1)$ passive corruptions.

Our actual construction also covers functions where each party has multiple bits as input and the function can output multiple bits (see Theorem 9). The compiler we use is the same as that of Halevi et al, except for using a leakage-resilient secret sharing scheme where theirs uses additive secret sharing. However, the proof of leakage tolerance is quite involved and, in fact, it turns out that standard local leakage resilience is insufficient for this purpose and we require strong leakage resilience. We now provide some intuition on why this is the case. In the Halevi et al's construction, some set of secrets are shared among all the parties in the correlated randomness generation phase. The messages sent during the execution of the protocol comprise of a subset of a party's shares. So, a party's secret state not only includes its own shares, but also the shares received from the other parties. Thus, the leakage function on an honest party's internal state is not local as it gets to see a subset of the other parties' shares. Thus, we need a secret sharing scheme satisfying the stronger notion of leakage resilience, where the leakage on the honest party's share can potentially depend on the shares of corrupted parties. For this purpose, we make use of the secret sharing scheme described in Informal Theorem 2.

1.2 Related Work

In a concurrent and independent work, Aggarwal et al. [ADN$^+$18] also construct leakage-resilient secret sharing schemes for any access structure from any secret sharing scheme for that access structure. Their transformation incurs a $O(1/n)$-factor loss in the rate and achieves a leakage-resilience rate of $(1-c)$ for a small constant c. In comparison, our transformation has a constant-factor loss in the rate and achieves a leakage-resilience rate of 1. They use their techniques and results to construct non-malleable secret sharing for 3-monotone access structures with an asymptotic rate of 0, and threshold signatures that are resilient to leakage and mauling attacks. In comparison, our compiler for non-malleable secret sharing is rate-preserving, but works only for 4-monotone access structures. Their work also considers the stronger model of concurrent tampering and gives positive results in this model as well.

In another concurrent and independent work, Kumar et al. [KMS18] also consider the problem of obtaining leakage-resilient secret sharing schemes in a stronger leakage model. In particular, they consider a leakage model where every bit of the leakage can depend on an adaptively chosen set of $O(\log n)$ shares. 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 are both $\Theta(1/\text{poly}(n))$. As an application, they construct a leakage-resilient non-malleable secret sharing scheme where the tampering function can obtain bounded, adaptive leakage from each share. In comparison, our strong leakage-resilient secret sharing scheme works against local leakage with a single level of adaptivity, where the leakage on each honest party's share could depend on at most $(t-2)$ shares in a t-out-of-n threshold scheme; our scheme has rate $\Omega(1/n)$ and a leakage-resilience rate of 1.

Apart from these, most closely related to our work are the papers by Goyal and Kumar on non-malleable secret sharing [GK18a, GK18b], Benhamouda et al on leakage-resilient secret sharing and MPC [BDIR18], and Badrinarayanan and Srinivasan on non-malleable secret sharing with non-zero rate [BS19].

Local leakage resilient secret sharing (in the sense in which we use this term) was first studied by Goyal and Kumar [GK18a] and Benhamouda et al [BDIR18] (independently of each other). [GK18a] constructed a local leakage resilient 2-out-of-n threshold secret sharing scheme with rate and leakage-resilience rate both $\Theta(1/n)$. They used this as a building block to construct non-malleable threshold secret sharing schemes secure against individual and joint tampering (where the adversary is allowed to jointly tamper sets of shares). A later paper also by Goyal and Kumar [GK18b] extended this to a compiler that adds non-malleability to a secret sharing scheme for any access structure. The non-malleable schemes resulting from both of these works, however, had rate tending to 0. Badrinarayanan and Srinivasan [BS19] later presented a compiler that converts any rate R secret sharing scheme to a non-malleable one for the same access struture with rate $\Theta(R/t_{\max}\log^2 n)$, where $t_{\max}$ is the maximum size of any minimal set in the access structure. In the process, they constructed local leakage resilient t-out-of-n secret sharing schemes for a constant t that had rate $\Theta(1/\log(n))$ and leakage-resilience rate $\Theta(1/n\log(n))$.

Benhamouda et al [BDIR18] were interested in studying the leakage-resilience of existing secret sharing schemes and MPC protocols. Inspired by the results of Guruswami and Wootters [GW16] that implied the possibility of recovering the secret from single-bit local leakage of Shamir shares over small characteristic fields, they investigated the leakage resilience of Shamir secret sharing over larger characteristic fields. They showed that, for large enough characteristic and large enough number of parties n, this scheme is leakage-resilient (with leakage-resilience rate close to $1/4$) as long as the threshold is large (at least $n-o(\log(n))$). They used this fact to show leakage-resilience of the GMW protocol [GMW87] (using Beaver's triples), and to show an impossibility result for multi-party share conversion.

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$^+$16, DHP11]. More broadly, leakage-resilience of various cryptographic primitives have been quite widely studied – we refer the reader to the survey by Alwen et al [ADW09] and the references therein. The notion of leakage tolerance was introduced by Garg et al [GJS11] and Bitansky et al [BCH12], and has been the subject of many papers since [BCG$^+$11, BGJ$^+$13, BDL14].

Secure multiparty computation with general interaction patterns was first studied by Halevi et al [HIJ$^+$16], who showed a reduction from general interaction patterns to the star pattern (which is what we base our reduction on). For any interaction pattern, they then showed an inefficient information-theoretically secure protocol for general functions, and an efficient one for symmetric functions; they also showed a computationally secure protocol for general functions assuming the existence of indistinguishability obfuscation and one-way functions, and for symmetric functions under an assumption about multilinear maps.

Subsequent Work. Subsequent to our work, Nielsen and Simkin [NS19] showed a lower bound on the share size of leakage resilient secret sharing schemes that satisfies the property that $\hat{t}$ shares completely determine the other $n-\hat{t}$ shares. In particular, they showed that the size of the shares of such schemes for threshold access structures with threshold t must be at least $\ell(n-t)/\hat{t}$ where ℓ is the size of the leakage tolerated. This in particular, shows that Shamir secret sharing cannot be leakage resilient for thresholds $o(n)$ when leaking, say, 1/4-th of the share size. On the other hand, it does not apply to schemes like ours where each share contains some randomness independent of the other shares and is not determined even given all the other shares.

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 $x \xleftarrow{\$} S$ 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. We assume the reader's familiarity with the standard definitions of min-entropy, statistical distance and seeded extractors and for completeness give the definition in the full version.

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

Definition 1 (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 (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. A dealer distributes a secret $m \in \mathcal{M}$ by computing the vector* $\mathsf{Share}(m) = (\mathsf{S}_1, \ldots, \mathsf{S}_n)$*, and privately communicating each share S_i to the party 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 3 ($(\mathcal{A}, n, \epsilon_c, \epsilon_s)$-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 $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$-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* $m \in \mathcal{M}$,
$$\Pr[\mathsf{Rec}(\mathsf{Share}(m)_T) = m] = 1 - \epsilon_c$$
where the probability is over the randomness of the Share *function. We will slightly abuse the notation and denote* Rec *as the reconstruction procedure that takes in $T \in \mathcal{A}$ and* $\mathsf{Share}(m)_T$ *as input and outputs the secret.*
- ***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 distinguisher D with output in $\{0,1\}$, the following holds :*
$$|\Pr[D(\mathsf{Share}(m_0)_U) = 1] - \Pr[D(\mathsf{Share}(m_1)_U) = 1]| \leq \epsilon_s$$

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 1 (Threshold Secret Sharing Scheme). For ease of notation, we will denote a t-out-of-n threshold secret sharing scheme as $(t, n, \epsilon_c, \epsilon_s)$-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 [GK18a].

Definition 4 (Leakage Resilient Secret Sharing [GK18a]). *An $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$ 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 $m_0, m_1 \in \mathcal{M}$:*
$$|f(\mathsf{Share}(m_0)) - f(\mathsf{Share}(m_1))| \leq \epsilon$$

3.1 Local Leakage Resilience

In this subsection, 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 $(\mathcal{S}_1 \times \mathcal{S}_2 \ldots \times \mathcal{S}_n)$ be the domain of shares for some secret sharing scheme, and $\mathcal{A}$ be an access structure. The corresponding local leakage function family is given by $\mathcal{F}_{\mathcal{A},\mu} = \{f_{K,\vec{\tau}} : K \subseteq [n], K \notin \mathcal{A}, \tau_i : \mathcal{S}_i \to \{0,1\}^{\mu}\}$ 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 a secret sharing scheme to be $\lim_{\mu\to\infty} \frac{\mu}{\max_{i\in[n]} \log|\mathcal{S}_i|}$.

Remark 2. We remark that Definition 4 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 1). Hence, in the rest of the paper, we will concentrate on 2-monotone access structures.

Description of the Compiler. We will give a compiler that takes any $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$ 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.

Let $(\mathsf{Share}, \mathsf{Rec})$ be a $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$ 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 $(\eta - \mu, \epsilon)$-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 a uniform seed $s \overset{\$}{\leftarrow} \{0,1\}^d$ and a masking string $r \overset{\$}{\leftarrow} \{0,1\}^\rho$.
    3. For each $i \in [n]$ do:
        (a) Choose $w_i \overset{\$}{\leftarrow} \{0,1\}^\eta$.
        (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 $m$.
```

Fig. 1. Local Leakage-Resilient Secret Sharing

Theorem 5. *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 $\epsilon_c, \epsilon_s, \epsilon \in [0, 1)$, the following exist:*

- *A* $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$ *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 $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$ *secret sharing scheme for* $\mathcal{M}$ *that is* $2(\epsilon_s + n \cdot \epsilon)$*-leakage resilient against* $\mathcal{F}_{\mathcal{A},\mu}$. *It has share size* $(\eta + 2\rho + d)$.

We give the proof of this theorem in the full version of the paper.

Instantiation. Next we demonstrate an instantiation of Theorem 5 with the state-of-the-art explicit construction of strong seeded extractors from the work of Guruswami, Umans and Vadhan [GUV09].

Theorem 6 ([GUV09]). *For any constant* $\alpha > 0$, *and all integers* $n, k > 0$ *there exists a polynomial time computable* (k, ϵ)*-strong seeded extractor* $\mathrm{Ext} : \{0,1\}^{n} \times \{0,1\}^{d} \to \{0,1\}^{m}$ *with* $d = O(\log n + \log(\frac{1}{\epsilon}))$ *and* $m = (1-\alpha)k$.

We now instantiate our scheme with the following building blocks:

- Let $(\mathsf{Share}, \mathsf{Rec})$ be a secret sharing scheme for a 2-monotone access structure $\mathcal{A}$ for sharing m-bit messages with rate R.
- We use the Guruswami, Umans and Vadhan [GUV09] strong seeded extractor (refer Theorem 6). We set $n = 1.01m/R + \log(1/\epsilon) + \mu$ and $d = O(\log n + \log(1/\epsilon))$ and from Theorem 6 and from [DORS08], it follows that Ext is a $(1.01m/R + \log(1/\epsilon), 2\epsilon)$ average-case, strong seeded extractor.

Thus, (using terminology from Figure 1) we get $|\mathsf{share}_i| = |w_i| + |\mathsf{Sh}_i| + |S_i| = n + m/R + (m/R + d) = 3.01m/R + \mu + O(\log m + \log \mu + \log 1/\epsilon)$.

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

For the special case of threshold secret sharing scheme for which we know constructions with rate 1 [Sha79], we obtain the following corollary, where $\mathcal{F}_{(t,n),\mu}$ denotes the local leakage function family corresponding to the t-out-of-n threshold access structure.

Corollary 2. *For any* $n, t, \mu \in \mathbb{N}$ *such that* $t \leq n$, *and* $\varepsilon \in (0,1)$, *there is a t-out-of-n threshold secret sharing scheme that is* $(2n\varepsilon)$*-leakage resilient against* $\mathcal{F}_{(t,n),\mu}$, *and has rate* $\Omega(1)$, *and leakage-resilience rate* 1.

3.2 Strong Local Leakage Resilience

In this subsection, we consider a stronger notion of leakage resilience for secret sharing, in which the leakage on the "honest" shares is allowed to depend arbitrarily on the "corrupted" shares – this is meant to capture a scenario where an adversary first learns the shares of t of the n parties, and then specifies leakage functions that are applied to the remaining $(n-t)$ shares, the outputs of which are then given to the adversary. This corresponds to leakage resilience against the function family described below.

Our motivation for studying this specific strengthening of local leakage resilience is an application to constructing leakage-tolerant MPC protocols where local leakage resilience turns out to be insufficient (see Section 5). For simplicity, we will describe our results (and definitions) in this subsection only for threshold access structures (which suffices for our MPC construciton), but they can be generalized to all access structures in a straightforward manner.

Semi-Local Leakage Function Family. Let $(\mathcal{S}_1 \times \cdots \times \mathcal{S}_n)$ be the domain of shares for some secret sharing scheme, and $t, t' \in [n]$ and μ be natural numbers. A semi-local leakage function family is parametrized by three numbers t (the adaptivity threshold), t' (the corruption threshold), and μ (the amount of leakage), such that $t \leq t'$. The family $\mathcal{H}_{t,t',\mu}$ consists of functions $\{h_{T,T',\vec{\tau}}\}$, where the subsets $T \subseteq T' \subseteq [n]$ are such that $|T| = t$ and $|T'| = t'$; and for $i \in [n] \setminus T'$, the function τ_i takes inputs from $(\mathcal{S}_{i_1} \times \cdots \times \mathcal{S}_{i_t}) \times \mathcal{S}_i$ (where $T = \{i_1, \ldots, i_t\}$), and outputs μ bits. The function $h_{T,T',\vec{\tau}}$, when given input $(\mathsf{share}_1, \ldots, \mathsf{share}_n)$, outputs share_i for each $i \in T'$, and $\tau_i((\mathsf{share}_{i_1}, \ldots, \mathsf{share}_{i_t}), \mathsf{share}_i)$ for $i \notin T'$.

A secret sharing scheme resilient to leakage by such function families is said to be *strongly local leakage resilient.*

Game-based Definition. Strong local leakage resilience of a secret sharing scheme $(\mathsf{LRShare}, \mathsf{LRRec})$ may alternatively, and perhaps more naturally, be defined as the inability of the adversary to guess the bit b correctly in the following game:

1. The adversary selects the sets $T \subseteq T' \subseteq [n]$ such that $|T| = t$ and $|T'| = t'$. It then picks messages $m_0, m_1 \in \mathcal{M}$, and sends all of these to the challenger.
2. The challenger picks a random bit b and computes $(\mathsf{share}_1, \ldots, \mathsf{share}_n) \leftarrow \mathsf{LRShare}(m_b)$. It sends share_T to the adversary.
3. The adversary now chooses a local leakage function $f_{(T' \setminus T),\mu}$ that operates on the $(n-t)$ shares $(\mathsf{share}_i)_{i \notin T}$. It sends this to the challenger.
4. The challenger sends the leakage $f_{(T' \setminus T),\mu}((\mathsf{share}_i)_{i \notin T})$.
5. The adversary outputs a guess b' for b.

We require that $\Pr[b = b'] = 1/2 + \mathrm{negl}(m)$. To see that these two definitions are equivalent, note that the task of the adversary in the game is essentially to specify a function from $\mathcal{H}_{t,t',\mu}$ – any function $h_{T,T',\vec{\tau}}$ in this class is specified by sets $T \subseteq T'$, outputs the shares in T' in the clear and also leaks some information about the honest parties' shares depending on the shares in T. And what the adversary gets from the challenger is precisely the output of this function applied to the shares.

We show that a modification of the construction from Section 3.1 can achieve strong local leakage resilience. This is presented in Figure 2.

Let $(\mathsf{Share}_{(t,n)}, \mathsf{Rec}_{(t,n)})$ represent a t-out-of-n threshold secret sharing scheme for secrets in an unspecified domain; let ρ be the bit-length of each share under this scheme when the secret is from the secret domain $\mathcal{M}$. Let η and d be such that there is a (k, ε)-average-case strong seeded extractor $\mathrm{Ext} : \{0,1\}^\eta \times \{0,1\}^d \to \{0,1\}^\rho$ that outputs ρ bits, where $k = (\eta - \mu)$.

```
LRShare : To share a secret $m \in \mathcal{M}$:
    1. Run $\mathsf{Share}_{(t,n)}(m)$ to obtain the shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$.
    2. Choose a uniform seed $s \xleftarrow{\$} \{0,1\}^d$.
    3. For each $i \in [n]$ do:
        (a) Choose $w_i \xleftarrow{\$} \{0,1\}^\eta$.
        (b) Choose a masking string $r_i \xleftarrow{\$} \{0,1\}^\rho$.
        (c) Set $\mathsf{Sh}'_i = \mathsf{Sh}_i \oplus \mathrm{Ext}(w_i, s) \oplus r_i$.
        (d) Run $\mathsf{Share}_{(t,n)}(r_i)$ to obtain $r_{(i,1)}, \ldots, r_{(i,n)}$.
    4. Run $\mathsf{Share}_{(t,n)}(s)$ to obtain $S_1, \ldots, S_n$.
    5. Output $\mathsf{share}_i$ as $(w_i, \mathsf{Sh}'_i, S_i, (r_{(1,i)}, \ldots, r_{(n,i)}))$.
LRRec : Given any set of $t$ shares $\mathsf{share}_{i_1}, \mathsf{share}_{i_2}, \ldots, \mathsf{share}_{i_t}$, do:
    1. For each $i_j$, parse $\mathsf{share}_{i_j}$ as $(w_{i_j}, S'_{i_j}, S_{i_j}, (r_{(1,i_j)}, \ldots, r_{(n,i_j)}))$.
    2. Run $\mathsf{Rec}_{(t,n)}(S_{i_1}, \ldots, S_{i_t})$ to recover $s$.
    3. For each $i_j$, do:
        (a) Run $\mathsf{Rec}_{(t,n)}(r_{(i_j,i_1)}, \ldots, r_{(i_j,i_t)})$ to recover $r_{i_j}$.
        (b) Recover $\mathsf{Sh}_{i_j}$ by computing $S'_{i_j} \oplus \mathrm{Ext}(w_{i_j}, s) \oplus r_{i_j}$.
    4. Run $\mathsf{Rec}(\mathsf{Sh}_{i_1}, \ldots, \mathsf{Sh}_{i_t})$ to recover the secret $m$.
```

Fig. 2. Strongly Local Leakage-Resilient Secret Sharing

Theorem 7. *Consider any $n, t, \mu \in \mathbb{N}$ such that $t \leq n$ and a secret domain $\mathcal{M}$. Suppose for some $\eta, d, R \in \mathbb{N}$ and $\epsilon \in [0,1)$, the following exist:*

- *A perfect t-out-of-n threshold secret sharing scheme with share size ρ for secrets in $\mathcal{M}$.*
- *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 2, when instantiated with these, is a t-out-of-n threshold secret sharing scheme for $\mathcal{M}$ that is $(2n\varepsilon)$-leakage resilient against $\mathcal{H}_{(t-2),(t-1),\mu}$. It has share size $(\eta + \rho + d + n\rho)$.

Using the same instantiations as in Section 3.1, we get the following.

Corollary 3. *For any $n, t, \mu \in \mathbb{N}$ such that $t \leq n$, and $\varepsilon \in [0, 1]$, there is a t-out-of-n threshold secret sharing scheme that is $(2n\varepsilon)$-leakage resilient against $\mathcal{H}_{(t-2),(t-1),\mu}$, and has rate $\Omega(1/n)$, and leakage-resilience rate* 1.

We prove Theorem 7 along the same lines as Theorem 5, and we give the details in the full version.

4 Rate Preserving Non-Malleable Secret Sharing

In this section, we will use the leakage resilient secret sharing scheme in Section 3 to construct a non-malleable secret sharing scheme. Specifically, we give a compiler that takes any secret sharing scheme for a 4-monotone access structure (see Definition 1) with rate R and converts it into a non-malleable secret sharing scheme for the same access structure with rate $\Omega(R)$.

In the full version, we give some background on non-malleable codes and below we recall the definition of non-malleable secret sharing for a monotone access structure $\mathcal{A}$.

Definition 5 (Non-Malleable Secret Sharing for General Access Structures [GK18b]). *Let* $(\mathsf{Share}, \mathsf{Rec})$ *be a* $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$*-secret sharing scheme for message space* $\mathcal{M}$ *and access structure* $\mathcal{A}$*. Let* $\mathcal{F}$ *be a family of tampering functions. For each* $f \in \mathcal{F}$*,* $m \in \mathcal{M}$ *and authorized set* $T \in \mathcal{A}$*, define the tampered distribution* $\mathsf{Tamper}_m^{f,T}$ *as* $\mathsf{Rec}(f(\mathsf{Share}(m))_T)$ *where the randomness is over the sharing function* Share*. We say that the* $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$*-secret sharing scheme,* $(\mathsf{Share}, \mathsf{Rec})$ *is* ϵ'*-non-malleable w.r.t.* $\mathcal{F}$ *if for each* $f \in \mathcal{F}$ *and any authorized set* $T \in \mathcal{A}$*, there exists a distribution* $D^{f,T}$ *over* $\mathcal{M} \cup \{\mathsf{same}^\star\}$ *such that for any* m*,*

$$|\mathsf{Tamper}_m^{f,T} - \mathrm{copy}(D^{f,T}, m)| \leq \epsilon'$$

where copy *is defined by* $\mathrm{copy}(x, y) = \begin{cases} x & \text{if } x \neq \mathsf{same}^\star \\ y & \text{if } x = \mathsf{same}^\star \end{cases}$*. We call* ϵ' *as the simulation error.*

4.1 Construction

We give a construction of a non-malleable secret sharing scheme for a 4-monotone access structures against the individual tampering function family $\mathcal{F}_{\mathsf{ind}}$ (see below).

Individual Tampering Family $\mathcal{F}_{\mathsf{ind}}$. Let Share be the sharing function of the secret sharing scheme that outputs n-shares in $\mathcal{S}_1 \times \mathcal{S}_2 \ldots \times \mathcal{S}_n$. The function family $\mathcal{F}_{\mathsf{ind}}$ is composed of tuples of functions $(f_1, \ldots, f_n)$ where each $f_i : \mathcal{S}_i \to \mathcal{S}_i$.

Construction. The construction is same as the one given in [BS19] but we instantiate the leakage-resilient secret sharing scheme with the one constructed in the previous section. We now give the description of the building blocks and then give the construction. In the following, we will denote a t-out-of-n monotone access structure as (t, n).

Building Blocks. The construction uses the following building blocks. We instantiate them with concrete schemes later:

- A 3-split-state non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{Enc} : \mathcal{M} \rightarrow \mathcal{L} \times \mathcal{C} \times \mathcal{R}$ and the simulation error of the scheme is ϵ_1. Furthermore, we assume that for any two messages $m, m' \in \mathcal{M}$, $(\mathsf{C}, \mathsf{R}) \approx_{\epsilon_2} (\mathsf{C}', \mathsf{R}')$ where $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(m)$ and $(\mathsf{L}', \mathsf{C}', \mathsf{R}') \leftarrow \mathsf{Enc}(m')$.
- A $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$ (where $\mathcal{A}$ is 4-monotone) secret sharing scheme $(\mathsf{SecShare}_{(\mathcal{A},n)}, \mathsf{SecRec}_{(\mathcal{A},n)})$ with statistical privacy (with error ϵ_s) for message space $\mathcal{L}$. We will assume that the size of each share is m_1.
- A $(3, n, 0, 0)$ secret sharing scheme $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$ that is ϵ_3-leakage resilient against leakage functions $\mathcal{F}_{(3,n),m_1}$ for message space $\mathcal{C}$. We assume that the size of each share is m_2.
- A $(2, n, 0, 0)$ secret sharing scheme $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$ for message space $\mathcal{R}$ that is ϵ_4-leakage resilient against leakage functions $\mathcal{F}_{(2,n),\mu}$ where $\mu = m_1 + m_2$. We assume that the size of each share is m_3.

We give the formal description of the construction in Figure 3 (taken verbatim from [BS19]).

Imported Theorem 8 ([BS19]) *For any arbitrary $n \in \mathbb{N}$ and any 4-monotone access structure $\mathcal{A}$, the construction given in Figure 3 is a $(\mathcal{A}, n, \epsilon_c, \epsilon_s + \epsilon_2)$ secret sharing scheme. Furthermore, it is $(\epsilon_1 + \epsilon_3 + \epsilon_4)$-non-malleable against $\mathcal{F}_{\mathsf{ind}}$.*

We defer the rate analysis to the full version of the paper and only state the corollary below.

Corollary 4. *For any $n \in \mathbb{N}$, $\rho > 0$ and 4-monotone access structure $\mathcal{A}$, if there exists a statistically private (with privacy error ϵ) secret sharing scheme for $\mathcal{A}$ that can share m-bit secrets with rate R, there exists a non-malleable secret sharing scheme for sharing m-bit secrets for the same access structure $\mathcal{A}$ against $\mathcal{F}_{ind}$ with rate $\Omega(R)$ and simulation error $\epsilon + 2^{-\Omega(m/\log^{1+\rho}(m))}$.*

5 Leakage Tolerant MPC for General Interaction Patterns

In this section, we will construct a leakage tolerant secure multiparty computation protocol for any interaction pattern (defined below). We will first recall some basic definitions from [HIJ+16].

Let $(\mathsf{SecShare}_{(\mathcal{A},n)}, \mathsf{SecRec}_{(\mathcal{A},n)})$ be a $(\mathcal{A}, n, \epsilon_c, \epsilon_s)$ (where $\mathcal{A}$ is 4-monotone) secret sharing scheme. Let $(\mathsf{Enc}, \mathsf{Dec})$ be a 3-split state non-malleable code and $(\mathsf{LRShare}_{(t,n)}, \mathsf{LRRec}_{(t,n)})$ be leakage resilient threshold secret sharing schemes with threshold t.

$\mathsf{Share}(m)$: To share a secret $s \in \mathcal{M}$ do:

1. Encode the secret s as $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$.
2. Compute the shares

$$(\mathsf{SL}_1, \ldots, \mathsf{SL}_n) \leftarrow \mathsf{SecShare}_{(\mathcal{A},n)}(\mathsf{L})$$

$$(\mathsf{SC}_1, \ldots, \mathsf{SC}_n) \leftarrow \mathsf{LRShare}_{(3,n)}(\mathsf{C})$$

$$(\mathsf{SR}_1, \ldots, \mathsf{SR}_n) \leftarrow \mathsf{LRShare}_{(2,n)}(\mathsf{R})$$

3. For each $i \in [n]$, set share_i as $(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)$ and output $(\mathsf{share}_1, \ldots, \mathsf{share}_n)$ as the set of shares.

$\mathsf{Rec}(\mathsf{Share}(m)_T)$: Given a set of shares in an authorized set $T' \in \mathcal{A}$, let $T \subseteq T'$ denote a minimal authorized set. To reconstruct the secret from the shares in set T (of size at most t), do:

1. Let the shares corresponding to the set T be $(\mathsf{share}_{i_1}, \ldots, \mathsf{share}_{i_t})$.
2. For each $j \in \{i_1, \ldots, i_t\}$, parse share_j as $(\mathsf{SL}_j, \mathsf{SC}_j, \mathsf{SR}_j)$.
3. Reconstruct

$$\mathsf{L} := \mathsf{SecRec}_{(\mathcal{A},n)}(\mathsf{SL}_{i_1}, \ldots, \mathsf{SL}_{i_t})$$

$$\mathsf{C} := \mathsf{LRRec}_{(3,n)}(\mathsf{SC}_{i_1}, \mathsf{SC}_{i_2}, \mathsf{SC}_{i_3})$$

$$\mathsf{R} := \mathsf{LRRec}_{(2,n)}(\mathsf{SR}_{i_1}, \mathsf{SR}_{i_2})$$

4. Output the secret s as $\mathsf{Dec}(\mathsf{L}, \mathsf{C}, \mathsf{R})$.

Fig. 3. Construction of Non-Malleable Secret Sharing Scheme for 4-monotone access structure taken verbatim from [BS19]

5.1 Basic Definitions

This subsection consists of definitions and some associated exposition, all taken verbatim from [HIJ$^+$16].

We begin by defining the syntax for specifying a communication pattern $\mathcal{I}$ and a protocol Π that complies with it. In all the definitions below, we let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote a fixed set of parties who would participate in the protocol. When we want to stress the difference between a protocol message as an entity by itself (e.g., "the 3rd message of party P_1") and the content of that message in a specific run of the protocol, we sometime refer to the former as a "message slot" and the latter as the "message content." To define an N-message interaction pattern for the parties in $\mathcal{P}$, we assign a unique identifier to each message slot. Without loss of generality, the identifiers are the indices 1 through N. An interaction pattern is then defined via a set of constraints on these message slots, specifying the sender and receiver of each message, as well as the other messages that it depends on. These constraints are specified by a message dependency graph, where the vertices are the message slots and the edges specify the dependencies.

Definition 6 (Interaction pattern [HIJ$^+$16]). *An N-message interaction pattern for the set of parties $\mathcal{P}$ is specified by a message dependency directed acyclic labeled graph,*

$$\mathcal{I} = ([N], D, L : V \to \mathcal{P} \times (\mathcal{P} \cup \mathsf{Out}))$$

The vertices are the message indices $[N]$, each vertex $i \in [N]$ is labeled by a sender-receiver pair $L(i) = (S_i, R_i)$, with $R_i = \mathsf{Out}$ meaning that this message is output by party S_i rather than sent to another party. The directed edges in D specify message dependencies, where an edge $i \to j$ means that message j in the protocol may depend on message i. The message-dependency graph must satisfy two requirements:

- *$\mathcal{I}$ is acyclic. We assume without loss of generality that the message indices are given in topological order, so $i < j$ for every $(i \to j) \in D$.*
- *If message j depends on message i, then the sender of message j is the receiver of message i. That is, for every $(i \to j) \in D$, we have $S_j = R_i$ (where $L(i) = (S_i, R_i)$ and $L(j) = (S_j, R_j)$).*

We assume without loss of generality that each party $P \in \mathcal{P}$ has at most one output, namely at most one $i \in [N]$ such that $L(i) = (P, \mathsf{Out})$. For a message $j \in [N]$, we denote its incoming neighborhood, i.e. all the messages that it depends on, by $\mathsf{DepOn}(j) := \{i : (i \to j) \in D\}$.

An n-party, N-message interaction pattern, is an N-message pattern for $\mathcal{P} = [n]$. We will interchangeably denote the i-th party as either using i or P_i.

A well known example of an interaction pattern is the star pattern which we define below.

Star Interaction Pattern. A $n+1$-party, $n+1$-message interaction pattern is called a star interaction pattern, if for each $i \in [n]$, $L(i) = (P_i, P_{n+1})$, $(i \to n+1) \in D$ and $L(n+1) = (P_{n+1}, \mathsf{Out})$. In other words, for every $i \in [n]$, P_i sends a single message to P_{n+1} who computes the output from all the messages received.

$\mathcal{I}$-compliant MPC. We next define the syntax of an MPC protocol complying with a restricted fixed interaction pattern. Importantly, our model includes general correlated randomness set-up, making protocols with limited interaction much more powerful.

Definition 7 ($\mathcal{I}$ compliant protocol [HIJ$^+$16]). *Let $\mathcal{I} = ([N], D, L)$ be an n-party N-message interaction pattern. An n-party protocol complying with $\mathcal{I}$ is specified by a pair of algorithms $\mathit{\Pi} = (\mathsf{Gen}, \mathsf{Msg})$ of the following syntax:*

- Gen *is a randomized sampling algorithm that outputs an n-tuple of correlated random strings $(r_1, \ldots, r_n)$.*
- Msg *is a deterministic algorithm specifying how each message is computed from the messages on which it depends. Concretely, the input of Msg consists of the index $i \in [N]$ of a vertex in the dependency graph, the randomness r_{S_i} and input x_{S_i} for the sender S_i corresponding to that vertex, and an assignment of message-content to all the messages that message i depends on, $M : \mathsf{DepOn}(i) \to \{0,1\}^*$. The output of* Msg *is an outgoing message in $\{0,1\}^*$, namely the string that the sender S_i should send to the receiver R_i.*

The execution of such a protocol Π with pattern $\mathcal{I}$ proceeds as follows. During an offline set-up phase, before the inputs are known, Gen is used to generate the correlated randomness $(r_1, ..., r_n)$ and distribute r_i to party P_i . In the online phase, on inputs $(x_1, \ldots, x_n)$, the parties repeatedly invoke Msg on vertices (message-slots) in $\mathcal{I}$ to compute the message-content they should send. The execution of Π goes over the message slots in a topological order, where each message is sent after all messages on which it depends have been received. We do not impose any restriction on the order in which messages are sent, other than complying with the depend-on relation as specified by $\mathcal{I}$. Once all messages (including outputs) are computed, the parties have local outputs $(y_1, \ldots, y_n)$, where we use $y_i = \perp$ to indicate that P_i does not have an output.

For a set $T \subset [n]$ of corrupted parties, let view_T denote the entire view of T during the protocol execution. This view includes the inputs x_T , correlated randomness r_T , and messages received by T. (Sent messages and outputs are determined by this information.) The view does not include messages exchanged between honest parties. Security of a protocol with communication pattern $\mathcal{I}$ requires that for any subset of corrupted parties $T \subset \mathcal{P}$, the view view_T reveals as little about the inputs $x_{\overline{T}}$ of honest parties as is possible with the interaction pattern $\mathcal{I}$. We formulate this notion of "as little as possible" via the notion of fixed vs. free inputs: If parties P_i , P_j are corrupted and no path of messages from P_i to P_j passes through any honest party, then the adversary can learn the output of P_j on every possible value of x_i . However, if there is some honest party on some communication path from P_i to P_j , then having to send a message through that party may be used to "fix" the input of P_i that was used to generate that message, so the adversary can only learn the value of the function on that one input.

Definition 8 (Fixed vs. free inputs.). *For an interaction pattern $\mathcal{I}$, parties $P_i, P_j \in \mathcal{P}$ (input and output parties), and a set $T \subset P$ of corrupted parties, we say that P_i has fixed input with respect to $\mathcal{I}, T$ and P_j if either*

- *$P_i \notin T$ (the input party is honest), or*
- *there is a directed path in $\mathcal{I}$ starting with some message sent by P_i, ending with some message received by P_j , and containing at least one message sent by some honest party $P_h \notin T$.*

We say that P_i has free input (with respect to $\mathcal{I}, T, P_j$) if $P_i \in T$ and its input is not fixed. We let $\mathsf{Free}(\mathcal{I}, T, P_j) \subseteq T$ *denote the set of parties with free inputs,* *and* $\mathsf{Fixed}(\mathcal{I}, T, P_j) = P \setminus \mathsf{Free}(\mathcal{I}, T, P_j)$ *is the complement set of parties with fixed input (all with respect to* $\mathcal{I}, T$ *and* P_j*).*

Using the notion of fixed inputs, we can now capture the minimum information available to the adversary by defining a suitable restriction of the function f that the protocol needs to compute.

Definition 9. *For an* n*-party functionality* f*, interaction pattern* $\mathcal{I}$*, corrupted set* $T \subset P$*, input* $x = (x_1, \ldots, x_n)$ *and output party* $P_j \in P$*, the residual function* $f_{\mathcal{I},T,x,P_j}$ *is the function obtained from* f_j *by restricting the input variables indexed by* $F = \mathsf{Fixed}(\mathcal{I}, T, P_j)$ *to their values in* x*. That is, for input variables* $x'_{\overline{F}} = \{x'_i\}_{i \notin F}$ *, we define* $f_{\mathcal{I},T,x,P_j}(x'_{\overline{F}}) = f_j(x'_1, \ldots, x'_n)$*, where* $x'_i = x_i$ *for all* $i \in F$*.*

We formalize our notion of security in the semi-honest model below. To get around general impossibility results for security with polynomial-time simulation [HLP11,GGG+14,BGI+14], we will allow by default simulators to be unbounded (but will also consider bounded simulation variants). We start by considering perfectly/statistically/computationally secure protocols.

Definition 10. *(Security with semi-honest adversaries). Let* f *be a deterministic* n*-party functionality,* $\mathcal{I}$ *be an* n*-party,* N*-message interaction pattern, and* $\Pi = (\mathsf{Gen}, \mathsf{Msg})$ *be an* n*-party protocol complying with* $\mathcal{I}$*. We say that* Π *is a perfectly* T*-secure protocol for* f *in the semi-honest model for a fixed set* $T \subset P$ *of corrupted parties if the following requirements are met:*

- ***Correctness:*** *For every input* $x = (x_1, \ldots, x_n)$*, the outputs at the end of the protocol execution are always equal to* $f(x)$ *(namely, with probability 1 over the randomness of* Gen*).*
- ***Semi-honest security:*** *There is an unbounded simulator* $\mathcal{S}$ *that for any input* x *is given* x_T *and the truth tables of the residual functions* $f_{\mathcal{I},T,x,P_j}$ *for all* $P_j \in T$*, and its output is distributed identically/statistically close/computationally indistinguishable to* $\mathsf{view}_T(x)$*.*

Remark 3 (Efficient Simulation). For the case where we require the simulator to be efficient, we provide the simulator with oracle access to the residual function $f_{\mathcal{I},T,x,P_j}$.

5.2 Definition: Leakage Tolerant MPC for an Interaction Pattern

We now define what it means for an MPC protocol compliant with an interaction pattern $\mathcal{I}$ to be *leakage-tolerant.*

We consider an $(n+1)$-party $\mathcal{P} = \{P_1, \ldots, P_n, P_{n+1}\}$ protocol $\Pi = (\mathsf{Gen}, \mathsf{Msg})$ that is compliant with an interaction pattern $\mathcal{I}$ with a single output party, namely, P_{n+1} (that does not have any inputs)[6] that computes a function f :

[6] The case of multiple output parties reduces to the case of single output party by considering each output party computing a specific function of the other parties input.

$(\{0,1\}^m)^n \to \{0,1\}^*$, where the party P_i gets input $x_i \in \{0,1\}^m$ for each $i \in [n]$. The execution of Π proceeds along an identical fashion as in the standard MPC for general interaction pattern (see Definition 7) and we recall this once again. In the offline phase before the parties get to know their actual inputs, the algorithm Gen is run and this outputs the correlated randomness $(r_1, \ldots, r_{n+1})$ where r_i is given to party P_i. In the online phase, on inputs $(x_1, \ldots, x_n)$, the parties repeatedly invoke Msg on vertices (message-slots) in $\mathcal{I}$ to compute the message-content they should send. The execution of Π goes over the message slots in a topological order, where each message is sent after all messages on which it depends have been received. Once all messages are sent, the output party P_{n+1} computes the output.

Let us say that at the end of a protocol Π, the party P_i's view $view_i$ is from a domain $\mathcal{V}_i$. Recall that view_i includes the correlated randomness output by Gen, party P_i's input x_i as well as the messages that it has received during the execution of the protocol. Let us denote $\Pi(x)$ as the joint distribution of the views of every party during the execution of the protocol. We are interested in adversaries that statically corrupt t $(< n)$ of the parties, obtaining their entire states, and also obtain some leakage on the states of the other uncorrupted parties. More formally, we represent the view of such adversaries as families of functions of the form $\mathcal{G}_{t,\mu} = \{g_{T,\vec{\tau}} : T \subseteq [n], |T| \leq t, \tau_i : \mathcal{V}_i \to \{0,1\}^\mu\}$; where $g_{T,\vec{\tau}}(\Pi(x))$ outputs $view_i$ for every $i \in T$, and $\tau_i(view_i)$ for $i \notin T$, when the protocol Π is run with input x – we refer to such a function as a (T, μ)-leakage function. Informally, we assume that the algorithm Gen runs in a leak-free manner and from then on, the honest party's entire secret state is subject to leakage.

Definition 11 (Leakage Tolerance against Semi-Honest Adversaries). *Let f be a deterministic n-party functionality, $\mathcal{I}$ be an n-party, N-message interaction pattern, and $\Pi = (\mathsf{Gen}, \mathsf{Msg})$ be an n-party protocol complying with $\mathcal{I}$. We say that Π is a (T, μ)*–leakage tolerant protocol *for f in the semi-honest model for a set $T \subseteq \mathcal{P}$ if it satisfies the following properties:*

- ***Correctness:*** *The protocol Π computes $f(x)$ correctly for any input $x = (x_1, \ldots, x_n)$.*
- ***Leakage Tolerance:*** *For any (T, μ)- leakage function $g_{T,\vec{\tau}}$, there is an unbounded simulator $\mathcal{S}$ satisfying the following.*
  - *For any input $x = (x_1, \ldots, x_n)$, the simulator $\mathcal{S}$ is given the inputs of the corrupted parties x_T and the truth tables of the residual functions $f_{\mathcal{I},T,x,P_j}$ for all $P_j \in T$ as input. It is allowed a single query to an oracle $\mathcal{O}[x_{\overline{T}}]$, which takes as input a tuple of functions $(\sigma_i)_{i \in \overline{T}}$, where each function is of the form $\sigma_i : \{0,1\}^m \to \{0,1\}^\mu$, and outputs $(\sigma_i(x_i))_{i \in \overline{T}}$.*
  - *We require that:*

$$g_{T,\vec{\tau}}(\Pi(x)) \approx \mathcal{S}^{\mathcal{O}[x_{\overline{T}}]}(f_{\mathcal{I},T,x,P_j}, x_T)$$

    where $\approx$ might indicate identical/statistically close/computationally indistinguishable.

We say that Π is a (t, μ)-leakage tolerant protocol for f *if it is (T, μ)-leakage tolerant for all $T \subseteq \mathcal{P}$ and $|T| \le t$.*

5.3 Construction

In this subsection, we give a construction of a leakage-tolerant semi-honest MPC for any interaction pattern $\mathcal{I}$. Specifically, we give a reduction from a leakage-tolerant semi-honest MPC for any interaction pattern $\mathcal{I}$ to constructing a (possible leakage intolerant) MPC protocol for the star interaction pattern. The construction we give is the same as the one given in [HIJ+16] with the only change being that we use our strong local leakage-resilient scheme instead of any secret sharing scheme.

Before we describe the construction, we introduce the following notation. For a function $f : (\{0,1\}^m)^n \to \{0,1\}^*$, we denote by $f^{bit} : \{0,1\}^{mn} \to \{0,1\}^*$ the function that takes mn bits as inputs, groups them together in order into n strings of length m each, and applies f on them.

Building Blocks. The construction uses the following building blocks:

- A star compliant, semi-honest protocol $\Pi^* = (\mathsf{Gen}^*, \mathsf{Msg}^*, \mathsf{Eval}^*)$ that securely (either perfect/statistical/computational) computes the function f^{bit}. Here, Msg^* denotes the next message function of the parties $P_1, \ldots, P_{mn}$ and Eval^* is the function computed by the evaluator (or in other words, party P_{mn+1}).
- A $(n+1, n+1, 0, 0)$ threshold secret sharing scheme $(\mathsf{LRShare}, \mathsf{LRRec})$ that is ϵ-strong leakage resilient for some negligible ϵ against the function family $\mathcal{H}_{n-1,n,\mu}$ (where $\mathcal{H}$ function class is defined in Section 3.2).

Construction. Let $f : (\{0,1\}^m)^n \to \{0,1\}^*$ be a n-party functionality that depends on all its inputs and $\mathcal{I}$ be an interaction pattern with a single sink. Let $\mathcal{P} = \{P_1, \ldots, P_{n+1}\}$ be the set of parties with P_{n+1} being the evaluator who does not have any inputs. We give the construction of an $\mathcal{I}$ compliant protocol in Figure 4.

Theorem 9. *If Π^* computes f^{bit} with statistical/computational security and* $(\mathsf{LRShare}, \mathsf{LRRec})$ *is an ϵ-strong leakage resilient secret sharing scheme against $\mathcal{H}_{n-1,n,\mu}$ for some negligible ϵ, then the construction in Figure 4 is a semi-honest, $\mathcal{I}$-compliant protocol for f that is (n, μ)-leakage tolerant with statistical/computational security. Furthermore, if each party uses R bits of correlated randomness and sends M bits in the protocol Π^*, then each party in the protocol in Figure 4 uses $O(m(R + n^2M + n\mu))$ bits of correlated randomness and sends $O((n^2M + n\mu)m)$ bits.*

We give the proof of this theorem in the full version. Using the known protocols for the star interaction pattern from the works of [BGI+14, BKR17, GGG+14], we obtain the following corollary.

Gen : To generate the correlated randomness, do:

1. Run Gen^* to obtain the correlated randomness $(r_1, \ldots, r_{mn+1})$.
2. For each $i \in [mn]$ and $\sigma \in \{0, 1\}$, compute $m_i^\sigma := \mathsf{Msg}^*(\sigma, r_i)$.
3. For each $i \in [mn]$ and $\sigma \in \{0, 1\}$, compute $(m_{i,1}^\sigma, \ldots, m_{i,n+1}^\sigma) \leftarrow \mathsf{LRShare}(m_i^\sigma)$.
4. Choose random permutation strings $b_1, \ldots, b_n \leftarrow \{0, 1\}^m$, one for each party P_i, $i \leq n$.
5. Let $c = b_1 \circ b_2 \circ \ldots \circ b_n$. For each $i \in [mn]$, let c_i denote the i-th bit of c.
6. For each $j \in [n]$, the correlated randomness for party j is $(\{m_{i,j}^{c_i}, m_{i,j}^{1-c_i}\}_{i\in[mn]}, b_j)$. The correlated randomness of the evaluator P_{n+1} is $(r_{mn+1}, \{m_{i,n+1}^{c_i}, m_{i,n+1}^{1-c_i}\}_{i\in[mn]})$

Msg : On input $x_j \in \{0, 1\}^m$ and the correlated randomness, party P_j does the following:

1. Parses the correlated randomness as $(\{M_{i,j}^0, M_{i,j}^1\}_{i\in[mn],\sigma\in\{0,1\}}, b_j)$.
2. Computes $s_j = x_j \oplus b_j$ and sends s_j on every path to the evaluator in $\mathcal{I}$.
3. Then, for every P_k such that some path from P_k to the evaluator goes through P_j, party P_j waits until it receives the string s_k and then sends $\{M^{s_{k,\ell}}_{(k-1)m+\ell,j}\}_{\ell\in[m]} = \{m^{x_{k,\ell}}_{(k-1)m+\ell,j}\}_{\ell\in[m]}$ on every path to the evaluator.
4. For every P_k such that no path from P_k to the evaluator goes through P_j, party P_j sends both shares $\{M^0_{(k-1)m+\ell,j}, M^1_{(k-1)m+\ell,j}\}_{\ell\in[m]}$ on every path to the evaluator.
5. In addition, P_j forwards every message that it receives from other parties on some $\mathcal{I}$-path to the evaluator.

Eval: The evaluator uses its correlated randomness to reconstruct $M^{s_{k,\ell}}_{(k-1)m+\ell}$ for every $k \in [n]$ and $\ell \in [m]$. It then uses the function Eval^* on these reconstructed values to learn the output.

Fig. 4. A $\mathcal{I}$ compliant protocol computing f. The construction is same as the one in [HIJ+16] except that we use our leakage resilient secret sharing.

Corollary 5 ([BGI+14, BKR17, GGG+14]). *Let $\mathcal{I}$ be a n-party interaction pattern with a single sink and let be $f : (\{0, 1\}^m)^n \to \{0, 1\}^*$ be a function which depends on all its inputs. Then,*

- *There is a statistical $\mathcal{I}$-compliant leakage tolerant protocol that securely computes f against upto $n-1$ passive corruptions. The communication complexity is exponential in n, m.*
- *If f is computable by a circuit in NC^1 and $m = O(\log n)$, then there exists an efficient $\mathcal{I}$-compliant leakage tolerant protocol that computes f with statistical security upto a constant number of corruptions. Assuming one-way functions, every f that is computable by polynomial-sized circuits has a computationally secure, efficient, $\mathcal{I}$-compliant leakage tolerant protocol upto a constant number of corruptions.*
- *Assuming indistinguishability obfuscation and one-way functions, every function computable by polynomial-sized circuits has a computationally secure,*

efficient, $\mathcal{I}$-compliant leakage tolerant protocol against upto $n-1$ passive corruptions.

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. Cryptology ePrint Archive, Report 2018/1147, 2018. `https://eprint.iacr.org/2018/1147`.

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

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.

BCG^{+}11. Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 722–739. Springer, Heidelberg, December 2011.

BCH12. Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 266–284. Springer, Heidelberg, March 2012.

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.

BDL14. Nir Bitansky, Dana Dachman-Soled, and Huijia Lin. Leakage-tolerant computation with input-independent preprocessing. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 146–163. Springer, Heidelberg, August 2014.

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.

BGI^{+}14. Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 387–404. Springer, Heidelberg, August 2014.

BGJ$^+$13. Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 316–334. Springer, Heidelberg, August 2013.

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.

BKR17. Fabrice Benhamouda, Hugo Krawczyk, and Tal Rabin. Robust non-interactive multiparty computation against constant-size collusion. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 391–419. Springer, Heidelberg, August 2017.

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

BS19. Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 593–622, 2019.

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.

FKN94. Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th Annual ACM Symposium on Theory of Computing*, pages 554–563. ACM Press, May 1994.

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.

GGG$^+$14. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 578–602, 2014.

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.

GJS11. Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 297–315. Springer, Heidelberg, August 2011.

GK18a. 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.

GK18b. Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 501–530. Springer, Heidelberg, August 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.

GMW17. Divya Gupta, Hemanta K. Maji, and Mingyuan Wang. Constant-rate non-malleable codes in the split-state model. Cryptology ePrint Archive, Report 2017/1048, 2017. `http://eprint.iacr.org/2017/1048`.

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

GW16. Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 216–226. ACM Press, June 2016.

HIJ+16. Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns. In Madhu Sudan, editor, *ITCS 2016: 7th Conference on Innovations in Theoretical Computer Science*, pages 157–168. Association for Computing Machinery, January 2016.

HLP11. Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 132–150. Springer, Heidelberg, August 2011.

HSH+09. J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 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.

KOS18. Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, pages 589–617, 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.

NS19. Jesper Buus Nielsen and Mark Simkin. Lower bounds for leakage-resilient secret sharing. *IACR Cryptology ePrint Archive*, 2019:181, 2019.

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.