Nearly Optimal Robust Secret Sharing against Rushing Adversaries

Pasin Manurangsi[*] Google Research
Akshayaram Srinivasan[†] UC Berkeley
Prashant Nalini Vasudevan[‡] UC Berkeley

October 1, 2019

Abstract

Robust secret sharing is a strengthening of standard secret sharing that allows the shared secret to be recovered even if some of the shares being used in the reconstruction have been adversarially modified. In this work, we study the setting where out of all the n shares, the adversary is allowed to adaptively corrupt and modify t shares, where $n = 2t + 1$.[1] Further, we deal with *rushing* adversaries, meaning that the adversary is allowed to see the honest parties' shares before modifying its own shares.

It is known that when $n = 2t + 1$, to share a secret of length m bits and recover it with error less than $2^{-\lambda}$, shares of size at least $m + \lambda$ bits are needed. Recently, Bishop, Pastro, Rajaraman, and Wichs (EUROCRYPT 2016) constructed a robust secret sharing scheme with shares of size $m + O(\lambda \cdot \text{polylog}(n, m, \lambda))$ bits that is secure in this setting against non-rushing adversaries. Later, Fehr and Yuan (EUROCRYPT 2019) constructed a scheme that is secure against rushing adversaries, but has shares of size $m + O(\lambda \cdot n^{\varepsilon} \cdot \text{polylog}(n, m, \lambda))$ bits for an arbitrary constant $\varepsilon > 0$. They also showed a variant of their construction with share size $m + O(\lambda \cdot \text{polylog}(n, m, \lambda))$ bits, but with super-polynomial reconstruction time.

We present a robust secret sharing scheme that is secure against rushing adversaries, has shares of size $m+O(\lambda \log n(\log n+\log m))$ bits, and has polynomial-time sharing and reconstruction. Central to our construction is a polynomial-time algorithm for a problem on semi-random graphs that arises naturally in the paradigm of local authentication of shares used by us and in the aforementioned work.

1 Introduction

Secret sharing, first studied by Shamir [Sha79] and Blakley [Bla79], is a fundamental cryptographic primitive that allows a secret to be shared among several parties in such a way that certain authorized subsets of parties can reconstruct the secret, while unauthorized subsets learn no information about the secret from their shares. Secret sharing has widespread applications across cryptography, ranging from secure multiparty computation [GMW87,BGW88,CCD88] to threshold cryptographic systems [DF90, Fra90, DDFY94].

[*]Email: pasin@google.com.
[†]Email: akshayaram@berkeley.edu.
[‡]Email: prashvas@berkeley.edu.

[1]Note that if the adversary is allowed to modify any more shares, then correct reconstruction would be impossible.

Typically, threshold secret sharing schemes[2] are required to satisfy two properties: *correctness*, which says that more than a certain number of parties can use their shares to reconstruct the secret, and *privacy*, which says that if there are fewer than this number of parties, then their shares together reveal nothing about the secret. A number of strengthenings of secret sharing have also been studied in the past owing to various applications, such as verifiable secret sharing [CGMA85], robust secret sharing [RB89], leakage-resilient secret sharing [BDIR18, GK18], etc. In this work, we focus on robust secret sharing.

Robust Secret Sharing. In robust secret sharing [RB89], in addition to the standard correctness and privacy properties, we require the following robustness property: even if some of the shares are adversarially modified, there is a reconstruction procedure that can recover the original secret given all the shares (among which it does not know which have been modified). In this sense, robust secret sharing is to standard threshold secret sharing as decoding from errors is to decoding from erasures in coding theory.

To be more specific, suppose a secret of length m bits is to be shared among n parties with threshold t – meaning the adversary is allowed to (adaptively) corrupt up to t of the parties. The properties we ask of a robust secret sharing scheme are:

- Correctness – given $(t+1)$ shares, it is possible to reconstruct the secret,
- Privacy – given t shares, the secret is hidden, and,
- Robustness – even if the adversary arbitrarily modifies up to t shares belonging to the parties it corrupts, the secret should be recoverable given all n shares.

If $t < n/3$, it may be seen that Shamir secret sharing [Sha79] with threshold t satisfies the robustness requirement, owing to the error correcting properties of the Reed-Solomon code. On the other hand, if $t \geqslant n/2$, robustness is impossible as the adversary could modify a majority of the shares. In addition, it is known that for $n/3 \leqslant t < n/2$, it is not possible to achieve perfect robustness, and any construction will necessarily have a small probability of failure of reconstruction [Cev11], which we will call the *robustness error*. Further, any robust secret sharing scheme for $n = 2t+1$ that has robustness error at most $2^{-\lambda}$ has shares of length at least $(m+\lambda)$ bits [CDV94].

In this work, we are interested in schemes that are robust in this extreme case of $n = 2t+1$. And the quantity we are most interested in is the size of the shares as a function of the secret length, the robustness error and the number of parties.

Prior Work. There has been significant past work [RB89, CDV94, CDF01, CDF$^+$08, CFOR12, Che15, CDD$^+$15, BP16, BPRW16, HO18, FY19] in studying and constructing robust secret sharing schemes, both in the setting where $n = 2t+1$, and where $t < (1-\delta)n/2$ for some constant $\delta > 0$. We discuss here only the former line of work, which is what leads up to our own, and refer the reader to the paper by Bishop et al. [BPRW16] for discussions of the rest.

The first construction of robust secret sharing was by Rabin and Ben-Or [RB89], and had a share size of $m + \tilde{O}(n\lambda)$ bits.[3] This was done by giving each party a set of $(n-1)$ keys of a message

[2]Throughout this work, we will be concerned only with threshold secret sharing, and thus we leave out this specification hereafter.

[3]Throughout the introduction, we use $\tilde{O}$ to hide polylogarithmic factors in λ,n, and m.

authentication code (MAC) that it could use to authenticate the shares held by each other party, and a set of $(n-1)$ MAC tags that other parties could use to authenticate its share. This was improved by Cramer, Damgard and Fehr [CDF01] to $m+\tilde{O}(n+\lambda)$ but with inefficient reconstruction, and later Cevallos et al. [CFOR12] achieved the same overhead with efficient reconstruction. The approach of the latter was to reduce the size of the authentication keys used in the MAC at the expense of a more complicated reconstruction procedure.

Recently, Bishop et al. [BPRW16] improved the overhead in the share size to $m+\tilde{O}(\lambda)$. The central idea in their work is to not authenticate each share to every other party, but instead, for each party to have d (roughly $O(\log n)$) authentication keys/tags corresponding to d other randomly chosen parties. They further showed that even though some of the keys/tags themselves could be adversarially modified, enough information can be recovered after such corruptions to reconstruct the secret. However, it was pointed out later by Fehr and Yuan [FY19] that the proof of robustness of this scheme relies on the adversary being *non-rushing.*

Rushing Adversaries. A rushing adversary, in our context, is one that decides how to change the shares of the parties it has corrupted after seeing the honest parties' shares. In the case of interactive reconstruction (which is what is used in our construction and in prior work), in each round the parties corrupted by the adversary may wait till they see all the honest parties' messages and then decide what to send. Robustness against such adversaries becomes relevant, for instance, if the parties are conducting the reconstruction amongst themselves as in multiparty computation protocols.

As mentioned above, Fehr and Yuan [FY19] noted that the proof of robustness of Bishop et al. [BPRW16] does not work if the adversary is rushing, though it is not known whether their construction is actually non-robust in this case. Fehr and Yuan then presented a construction of robust secret sharing, using the local authentication approach of Bishop et al., that was robust against rushing adversaries, but with a share size of $m+\tilde{O}(\lambda\cdot n^{\varepsilon})$ for an arbitrary constant $\varepsilon>0$. They also showed how to improve this to $m+\tilde{O}(\lambda)$ if the reconstruction procedure was allowed to run in super-polynomial time.

Our Results. We construct robust secret sharing against rushing adversaries in the setting of $n=2t+1$, with a share size of $m+O(\lambda\cdot\log n(\log n+\log m))$ for secrets of size m and robustness error $2^{-\lambda}$. (Note that the overhead here is only polylogarithmically larger than in the lower bound of $m+\lambda$ for share size shown in [CDV94].) Our reconstruction procedure is interactive, with two rounds of interaction, and both sharing and reconstruction are polynomial-time. Our approach is similar to those of Bishop et al. and Fehr and Yuan, though our construction is simpler than either and does not use some of the sophisticated tools used there.

1.1 Technical Overview

In this subsection, we give a high-level overview of our construction and the techniques we use. Recall that we wish to share secrets of length m among n parties, with a threshold of t (the adversary is allowed to corrupt t parties), with a robustness error smaller than $2^{-\lambda}$.

1.1.1 Sharing and Reconstruction

We follow the local authentication paradigm used in [BPRW16,FY19]. In this approach, each party is given authentication information about the shares of a small set of parties, which we will call its "watchlist." We will set the size of the watchlist to be roughly $O(\log n)$, thus this contributes only a polylog(n) factor to the overhead in the size of each share. This is to be contrasted with Rabin and Ben-Or's approach [RB89] where each party stores authentication information about every other party, thus leading to a linear blow-up in the overhead. We now give some more details on how we use the local authentication paradigm.

Sharing. In the sharing phase, we first compute a set of Shamir shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$ for the given secret. Then, for each $i \in [n]$, we pick a random multiset S^i of size d (where d is roughly $O(\log n)$) from $[n] \setminus \{i\}$. S^i will be the set of parties in the watchlist of party i. For every $j \in S^i$, we pick a random MAC key $k_{i \to j}$ and compute the tag $\sigma_{i \to j}$ of the j^{th} Shamir share Sh_j using $k_{i \to j}$. The tuple $(k_{i \to j}, \sigma_{i \to j})$ constitutes authentication information for Sh_j. The share corresponding to the i^{th} party includes the Shamir share Sh_i, the watchlist S^i, and the authentication information $\{k_{i \to j}, \sigma_{i \to j}\}_{j \in S^i}$ of the parties in its watchlist.

There is, however, a concern about privacy as we are storing both the key $k_{i \to j}$ and the tag $\sigma_{i \to j}$ together and this might leak some information about the share Sh_j that is being authenticated. In order to deal with this issue, we use a tool called private (randomized) MAC introduced in [BPRW16]. This private MAC has the property that for any key $k_{i \to j}$, the pair $(k_{i \to j}, \sigma_{i \to j})$ does not reveal any information about Sh_j. This allows us to argue that even when the key is stored together with the tag, the privacy is still preserved.

Reconstruction. Recontruction is performed by a two-round interactive protocol that proceeds as follows.

- In the first round, the i-th party broadcasts its Shamir share Sh_i. The honest parties will broadcast the correct shares, whereas for the adversarial parties, the broadcasted shares could either be the original share or some modified (even empty) share. At this point, we may partition the set of parties into three sets – the set H of honest parties, the set P of "passive" corrupted parties i that are corrupted but broadcast the correct share Sh_i, and the set A of "active" corrupt parties that broadcast a modified share.

  At the end of the first round, all the parties can determine if the shares of the parties in its watchlist have been modified or not by checking if the corresponding MAC tag verifies under the respective key. Specifically, for every $j \in S^i$, such that the $\sigma_{i \to j}$ verifies, party i labels j as "good." Similarly, if the tag does not verify, it labels j as "bad." Thus, at the end of the first round, the parties can obtain the labels for each $j \in S^i$. Note that the honest parties will always label a party $j \in H \cup P$ as "good" and with overwhelming probability, will label a party in A as "bad." Furthermore, at the end of the first round, the adversarial parties do not learn any information about the watchlist of the honest parties. This will be crucially used to argue robustness.

- In the second round, each party i broadcasts S^i along with the labels it computed as above for each $j \in S^i$. Again, the honest parties will broadcast the correct information whereas the adversarial parties, including the parties in P, can broadcast incorrect information. In particular, an adversarial party might modify its watchlist, and also incorrectly accuse some honest party as being "bad" or label a party in A as "good."

The action of any adverary in this protocol effectively induces a (labelled directed) graph on the vertex set $V = [n]$ (with a vertex representing each party) that is generated by the following process:

- The adversary partitions V into sets H, P, and A, where $|H| \geqslant t+1$.
- For every $i \in H$, we choose a random multiset S^i of size d from $[n] \setminus \{i\}$. For each $j \in S^i$, we add an edge (i, j) to the graph. We label the edge (i, j) to be "good" if $j \in H \cup P$, and "bad" if $j \in A$.
- The outgoing edges from $P \cup A$ and their labels are generated adversarially after seeing the edges and labels from H.

(In the above process, we sample the watchlists S^i of honest parties *after* the adversary has partitioned the set into H, P, A. We note that this is fine, in spite of the fact that the watchlists are actually sampled during the sharing, since the adversary does not learn any information about the watchlist of honest parties at the end of the first round of the reconstruction protocol.)

Suppose we have an algorithm that on input the above graph, outputs the set $S = H \cup P$. In this case, we are done since we can use the shares of these parties to reconstruct the correct secret by the correctness of Shamir sharing. We give an algorithm that finds an S that has a large intersection with $H \cup P$ and a small intersection with A. With such an S, we can use the error correction properties of Shamir secret sharing (a.k.a. Reed-Solomon codes) to recover the correct secret.

We next briefly describe how the above graph algorithm works.

1.1.2 Vertex Identification Algorithm

Our vertex identification algorithm crucially uses the connection between the problem at hand and the *independent set* problem. Recall that a subset of vertices in an *undirected graph* is said to be an independent set if there is no edge between any pair of vertices in the set. While our graph G is a directed graph, there is a natural way to view it as an undirected graph: by simply keeping each edge with a "bad" label as an undirected edge and discarding all edges with "good" labels. It is not hard to show that a maximum independent set in this undirected version of G would give the desired set S.

Unfortunately, computing the maximum independent set is NP-hard in the worst case [Kar72]. On the other hand, our graph is not a worst case graph since all edges from H are random, although the edges from $A \cup P$ are worst case (i.e., adversarially generated). Finding independent sets in such "semi-random" graphs has long been a topic of study in literature, starting with the work of Feige and Kilian [FK01] (see also [FK00, CSV17, MMT18]). Unfortunately, these works do not apply to our scenario because of the following two reasons. First, the guarantees from this line of work do not suffice for us; a typical guarantee there is that an independent set found has a large size relative to the maximum independent set, whereas we need the fact that the independent set has a large intersection with $H \cup P$ and a small intersection with A. Second, the distribution of our graph is unlike those considered in [FK01, FK00, CSV17, MMT18]. Specifically, the distributions considered in literature are often the following: pick a set I of vertices (i.e., the "planted independent set") and add random edges between I and the remaining vertices. Then, the adversary is allowed to add arbitrary edges that are not within I. However, this is not the case for us since the edges from P to H are not random.

Despite the challenges mentioned in the previous paragraph, several things go in our favor. First, our directed graph G actually contains more information than its undirected variant considered in the previous paragraph. For instance, if we have two vertices u_1, u_2 each having a directed edge pointing to v but with different labels, then we know that either u_1 or u_2 must be corrupted. Such information is not included when we just consider finding an independent set in the trivial undirected version of G. This motivates us to look instead to what we call the *conflict graph* G^{conf}, where we add an edge between every pair of vertices u_1, u_2 that label a common neighbor differently. Clearly, H remains an independent set in G^{conf}. Moreover, from the definition of G^{conf}, any independent set I of G^{conf} has a "consistent opinion" on all vertices in the following sense: every vertex v is labelled with the same label by all its in-neighbors that lie in I. This leads us to the overall structure of our algorithm: (1) find a large independent set I in G^{conf} and (2) output the set of all vertices labelled "good" by (at least one vertex in) I.

Of course, we have not yet specified how we find a large independent set I in G^{conf}. This is indeed where a second advantage of our scenario comes in: we are guaranteed to have an independent set H of size more than half of the graph, unlike previous works that place weaker assumptions on the size of the "planted" independent set. It turns out that this $1/2$ threshold makes the problem "easier." For instance, Nemhauser and Trotter [NT74] show that any extremal solution to the linear program (LP) relaxation of the independent set problem is half integral (i.e., every variable is assigned either 0, 1/2, or 1), which means that at least one vertex in H is assigned 1 in the solution. In our proof, we use a more specific structural lemma from [ACF$^+$04] (Lemma 3.10) together with the expansion property of the random part of our graph (Lemma 3.11) to argue that, if we let I be the set of all vertices assigned 1 by the LP solution, then it contains sufficiently many vertices from H. This in turn implies that I labels most of the vertices in $H \cup P$ as "good" and most of those in A as "bad." A more quantitative version of this argument shows that the output set satisfies the desired properties.

1.2 Comparison with Prior Work

As mentioned earlier, our construction follows the paradigm of local authentication of shares introduced by Bishop et al. [BPRW16] and used also by Fehr and Yuan [FY19]. There are a number of similarities and differences between how we proceed in this paradigm and how these papers do, and we briefly explain these below.

Bishop et al. [BPRW16]. The authors here use local authentication of shares to reduce the robust reconstruction to a graph theoretic problem called graph bisection, which when solved gives a set S of trustworthy parties whose shares are used to reconstruct the secret.

Their reduction also involves partitioning the corrupted parties into a set P of "passive" corruptions and a set A of "active" corruptions according to whether a certain part of the shares are reported correctly during reconstruction. But their notion of passive corruption is stronger than the one we use here – they also require that parties in P never falsely label a party in H as being "bad." This required them to store the authentication information in a distributed manner, using a primitive they call robust distributed storage. Additionally, they had to authenticate not only the Shamir shares (as we do), but also the MAC keys themselves. As pointed out by Fehr and Yuan, authenticating the keys in this way eventually causes their proof of robustness to fail against rushing adversaries. Finally, in the end, the set S that is the solution to the graph bisection problem is the set $H \cup P$.

We use a weaker notion of passive corruptions, where parties in P are allowed to label parties in H as "bad." This allows us leave out the distributed storage and the authentication of the MAC keys, enabling proofs of security against rushing adversaries. And solving the graph problem that we reduce to does not require recovering the entire set $H \cup P$, but only a set S that has a large intersection with $H \cup P$ and a small intersection with A. Such a set is easier to find, which is what lets us relax the definition of passive corruptions, and is still sufficient to recover the shared secret due to the error correction properties of Shamir sharing.

Fehr and Yuan [FY19]. The approach of Fehr and Yuan [FY19] is more similar to ours. They also partition the parties into sets H, P and A, and their definitions for these sets are the same as ours. However, they still use robust distributed storage to store authentication information. Our construction is simpler, using only private MACs, which both of these papers also use.

Their approach is also to recover a set S of vertices that has a large intersection with $H \cup P$ and a small intersection with A, and then use the error correction properties of Shamir sharing to recover the secret. Further, like our approach, they also do this by reducing robust reconstruction to the vertex identification problem in the model of random graphs that comes up in our work.

But their algorithm to solve this problem over n vertices with out-degree d only works when the number of passive parties ($|P|$) is at most roughly $n \cdot (\log d / \log n)^2$. And so when they find that $|P|$ is less than this and their algorithm fails, they fall back to list-decoding on all the shares together to recover a list of possible sharings and then iterate over this list to find the actual secret. The size of this list is roughly $(\log n / \log d)^{\tilde{O}((\log n / \log d)^2)}$, leading to the restriction of $d \geqslant n^{\varepsilon}$ for some constant ε for the list-decoding to run in polynomial-time.

Our algorithm, on the other hand, solves the same problem without any such restriction on $|P|$. To be more precise, it first solves the problem when $|P|$ is at most roughly $0.84 \cdot n$, and then observes that if $|P|$ is more than this, then $S = [n]$ is already a solution. As $|P|$ can be efficiently estimated from the graph, this solves the problem. This releases us from their restriction on d, which we can set to be $O(\log n)$, leading to our shares being significantly smaller.

Outline. In Section 2, we define robust secret sharing and private MACs, and state some known facts and theorems that will be useful later. In Section 3, we present our vertex identification algorithm that will be used in our reconstruction procedure. For readers who are only interested in our cryptographic constructions, we suggest to look at Theorem 3.1 in Section 3 and skip to Section 4, where we present our robust secret sharing scheme.

2 Preliminaries

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

Multisets. Let S be a multiset and for any element a, we define the multiplicity $m_a(S)$ to be the number of times a occurs in the multiset S. For any two multisets S_1, S_2, we define $S_1 \| S_2$ to be the multiset such that for any element a, $m_a(S_1 \| S_2) = m_a(S_1) + m_a(S_2)$.

We assume the reader's familiarity with the definition of statistical distance.

2.1 Private MAC

In this subsection, we recall the definition of a private message authentication code (MAC) used by Bishop et al [BPRW16] and Fehr and Yuan [FY19], but using different terminology. A private MAC for message space $\{0,1\}^\eta$ for some $\eta \in \mathbb{N}$ consists of the following algorithms, all of them running in time poly(η).

- KeyGen : A randomized algorithm that outputs a key k.
- $\mathsf{Tag}(k,(m,r))$: A deterministic algorithm that takes a key k, a "message tuple" $(m,r) \in \{0,1\}^\eta \times \{0,1\}^\kappa$ for some $\kappa \in \mathbb{N}$ (called the randomness length), and outputs a tag σ.
- $\mathsf{Verify}(k,(m,r),\sigma)$: A deterministic algorithm that takes a key k, a message tuple (m,r), and a tag σ, and outputs 1 or 0.

Definition 2.1 *For an $\eta, \ell \in \mathbb{N}$ and $\varepsilon \in [0,1]$, a triple of algorithms* $(\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$ *is an (ℓ,ε)-private MAC for a message space $\{0,1\}^\eta$ if the following properties are satisfied for some $\kappa \in \mathbb{N}$.*

- ***Correctness.*** *For every message tuple $(m,r) \in \{0,1\}^\eta \times \{0,1\}^\kappa$, with $k \leftarrow \mathsf{KeyGen}$, and $\sigma \leftarrow \mathsf{Tag}(k,(m,r))$,*
$$\Pr[\mathsf{Verify}(k,(m,r),\sigma)=1]=1$$
- ***Unforgeability.*** *For any message tuple $(m,r) \in \{0,1\}^\eta \times \{0,1\}^\kappa$, and any adversary $\mathcal{A}$, with $k \leftarrow \mathsf{KeyGen}$, $\sigma \leftarrow \mathsf{Tag}(k,(m,r))$, and $(m',r',\sigma') \leftarrow \mathcal{A}(m,r,\sigma)$,*
$$\Pr[(m,r) \neq (m',r') \wedge \mathsf{Verify}(k,(m',r'),\sigma')=1] \leqslant \varepsilon.$$
- ***Privacy.*** *For every $m_0, m_1 \in \{0,1\}^\eta$, any arbitrary set of ℓ keys $\{k_1,\ldots,k_\ell\}$, and any adversary $\mathcal{A}$, with $k \leftarrow \mathsf{KeyGen}$, $r \leftarrow \{0,1\}^\kappa$, and $\sigma_i^b \leftarrow \mathsf{Tag}(k_i,(m_b,r))$ for $i \in [\ell]$ and $b \in \{0,1\}$,*
$$\Pr[\mathcal{A}(\sigma_1^0,\sigma_2^0,\ldots,\sigma_\ell^0)=1] = \Pr[\mathcal{A}(\sigma_1^1,\sigma_2^1,\ldots,\sigma_\ell^1)=1]$$
- ***Uniformity.*** *There is an $s \in \mathbb{N}$ such that for every $(m,r) \in \{0,1\}^\eta \times \{0,1\}^\kappa$, with $k \leftarrow$ KeyGen, the distribution of $\sigma \leftarrow \mathsf{Tag}(k,(m,r))$ is uniform over $\{0,1\}^s$.*

The following theorem follows from the construction of a private MAC presented in [BPRW16], using GF$[2^\lambda]$ as the field there.

Theorem 2.2 ([BPRW16]) *For any $\eta, \ell \in \mathbb{N}$ and $\varepsilon \in [0,1]$, there exists an (ℓ,ε)-private MAC for message space $\{0,1\}^\eta$, with randomness length $\ell\lambda$, key length 2λ and tag length λ, where $\lambda = \lceil \log((\eta+\ell)/\varepsilon) \rceil$.*

2.2 Secret Sharing Scheme

We start with the definition of the sharing function and then give the definition of a threshold secret sharing scheme.

Definition 2.3 (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 2.4 ($(t, n, \varepsilon_c, \varepsilon_s)$-Secret Sharing Scheme) *Let $\mathcal{M}$ be a finite set of secrets, where $|\mathcal{M}| \geqslant 2$. 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 $(t, n, \varepsilon_c, \varepsilon_s)$-secret sharing scheme if the following two properties hold :*

- ***Correctness:*** *The secret can be reconstructed by any t-out-of-n parties. That is, for any set $T \subseteq [n]$ such that $|T| \geqslant t$, there exists a deterministic, interactive reconstruction protocol* Rec *between the parties in T with the input of $i \in T$ being $\mathsf{Share}(m)_i$ such that for every $m \in \mathcal{M}$,*

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

  where the probability is over the randomness of the Share *function. We will slightly abuse the notation and denote* Rec *as the reconstruction protocol that takes in T and $\mathsf{Share}(m)_T$ where T is of size at least t and outputs the secret.*

- ***Statistical Privacy:*** *Any collusion of less than t parties should have "almost" no information about the underlying secret. More formally, for any unauthorized set $U \subseteq [n]$ such that $|U| < t$, and for every pair of secrets $m_0, m_1 \in M$, for any distinguisher D with output in $\{0, 1\}$, the following holds :*

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

We define the rate of the secret sharing scheme as

$$\lim_{|m| \to \infty} \frac{|m|}{\max_{i \in [n]} |\mathsf{Share}(m)_i|}$$

Remark The above definition of privacy considers a weaker notion where the unauthorized set U is specified upfront. We can also consider a stronger variant where the adversary adaptively specifies this set U one party at a time, seeing the share of each party as it is specified. We note that for the case of perfect privacy (i.e., $\varepsilon_s = 0$), the above two variants are equivalent.

2.3 Robust Secret Sharing

We now give the definition of robust secret sharing scheme.

Definition 2.5 *An $(t, n, \varepsilon_c, \varepsilon_s)$ secret sharing scheme* $(\mathsf{Share}, \mathsf{Rec})$ *for message space $\mathcal{M}$ is said to be (δ, κ, τ)-*robust *if for every interactive adversary $\mathcal{A}$ and message $m \in \mathcal{M}$,*

$$\Pr[\mathsf{Expt}_{\mathcal{A},m,\kappa,\tau} = 1] \leqslant \delta$$

where $\mathsf{Expt}_{\mathcal{A},m,\kappa,\tau}$ is defined below.

- $(\mathsf{share}_1, \dots, \mathsf{share}_n) \leftarrow \mathsf{Share}(m)$.
- *$\mathcal{A}$ outputs a set $\Gamma \subseteq [n]$ such that $|\Gamma| = \kappa$.*
- *Set $T = \emptyset$. Repeat until $|T| = \tau$:*
  - *$\mathcal{A}$ chooses $i \in \Gamma \setminus T$.*
  - *Update $T = T \cup \{i\}$ and give share_i to $\mathcal{A}$.*
- *Run the reconstruction protocol among the parties in Γ with every party $i \in \Gamma \setminus T$ behaving honestly using its share share_i and the adversary $\mathcal{A}$ taking control the parties in T. $\mathcal{A}$ is allowed to behave maliciously (possibly using a different share) and can deviate arbitrarily from the specification of the reconstruction protocol. Here, we assume that in every round of the reconstruction protocol, $\mathcal{A}$ can send its outgoing messages after seeing all its incoming messages from the honest parties (a.k.a. rushing adversary). For every $i \in \Gamma \setminus T$, let m'_i be the output of the i-th party at the end of the reconstruction algorithm.*
- *Output 1 if and only if there exists an $i \in \Gamma \setminus T$ such that $m \neq m'_i$.*

We call $\log(1/\varepsilon_s)$ as the privacy parameter and $\log(1/\delta)$ as the robustness parameter.

Fact 2.6 *Fix $t \in \mathbb{N}$, $n, \kappa \geqslant t$. There is a robust reconstruction protocol such that Shamir secret sharing is a $(t, n, 0, 0)$ secret sharing scheme that is $(0, \kappa, \lfloor(\kappa - t)/2\rfloor)$-robust.*

3 Vertex Identification Algorithm

In this section, we give a polynomial time graph algorithm that we call the vertex identification algorithm, which will be used as a building block in the construction of robust secret sharing. We start with the description of the model of semi-random graphs the algorithm works for.

Model. Our (directed) graph[4] G where $|V| = 2t + 1$ and a labeling $L : E_G \to \{\text{good}, \text{bad}\}$ is generated as follows:

- First, the adversary partitions V into three parts H, P, A such that $|H| \geqslant t + 1$.
- For every $u \in H$ and every $i \in \{1, \dots, d\}$, select a vertex $v_i \in (V \setminus \{u\})$ uniformly at random and add an edge (u, v_i) to the graph. The label for $L(u, v_i)$ of the edge is "good" if v_i lies in $H \cup P$, and is "bad" if $v_i \in A$.

[4] We note that our graph allows multi-edges and self-loops.

- The outgoing edges of $A \cup P$ and their labels are generated by the adversary after seeing the edges and the labels from H.

For notational convenience, we will always think of each vertex $v \in V$ as having a self-loop with label $L(v,v) = \text{good}$. We use n to denote the number of total vertices, i.e. $n = 2t+1$. Our main theorem in this section is the following:

Theorem 3.1 *There is a polynomial-time (deterministic) algorithm* VertexID *that, given G, L generated as above with $d \geqslant C \log n$, outputs a set $S \subseteq V$ that satisfies*

$$|S \cap (H \cup P)| \geqslant (t+1) + 2 \cdot |S \cap A| \tag{3.1}$$

with probability $1 - O(e^{-\beta d})$, where $C > 1$ and $\beta > 0$ are some constants.

The Algorithm. Our algorithm relies on the linear programming (LP) relaxation of the Independent Set problem. The LP relaxation for an undirected graph $F = (V, E_F)$ can be stated as:

$$\max \sum_{v \in V_F} x_v \tag{3.2}$$

$$\text{subject to } 0 \leqslant x_v \leqslant 1 \qquad \forall v \in V \tag{3.3}$$

$$x_u + x_v \leqslant 1 \qquad \forall \{u,v\} \in E_F \tag{3.4}$$

We refer to the above relaxation as LP-IS of F.

As the reader might have noticed, the Independent Set (IS) problem is defined on undirected graphs, whereas our input graph G is a directed graph. To turn this into an instance of Independent Set, we create what we will call the *conflict graph* of G, L:

Definition 3.2 *Given a directed graph $G = (V, E_G)$ and a labeling $L : V \to \{good, bad\}$, their* conflict graph *is denoted*[5] *by $G^{conf} = (V, E^{conf})$. This is an undirected graph on the same vertex set of G, and there is an edge between two vertices $u, v \in V$ iff there exists a common out-neighbor w of u, v such that $L(u,w) \neq L(v,w)$.*

Recall that we always add a self-loop with "good" label to every vertex in G, which means that $\{u,v\}$ will always be an edge in the conflict graph if $(u,v) \in E_G$ and $L(u,v) = \text{bad}$.

There are a couple (straightforward) observations that will be useful to keep in mind. The first one is that H is an independent set in this conflict graph:

Observation 3.3 *H is an independent set in the conflict graph of G, L.*

The second is that, for any independent set I of the conflict graph, its vertices never label a vertex inconsistently. This following from the definition of the conflict graph, as such an inconsistency would create an edge in the graph.

Observation 3.4 *Let I be any independent set of a conflict graph of G, L. Then, for any $v \in V$, it must fall into one of the following three categories:*

[5]Of course, the conflict graph depends on the labeling. However, we choose not to have L in the notation to avoid cumbersomeness.

- *v has no in-neighbor (w.r.t G) in I.*
- *v has at least one in-neighbor (w.r.t G) in I and each of its neighbors labels it with good.*
- *v has at least one in-neighbor (w.r.t G) in I and each of its neighbors labels it with bad.*

With these ready, we can now describe our algorithm, which we call Conflict-LP:

```
Conflict-LP($G = (V, E_G), L$)

0. For each vertex $v \in V$, let $b_v$ denote the number of vertices labelled bad by it. If the
   median of $b_v$'s is at most $0.16d$, then output $V$ and terminates.

1. Construct the conflict graph $G^{\mathrm{conf}}$ of $G, L$ (as in Definition 3.2).

2. Solve for an extreme point optimum solution $\{x^*_v\}_{v \in V}$ of LP-IS of $G^{\mathrm{conf}}$.
   Let $I$ denote the set of all vertices $u \in V$ such that $x^*_u = 1$.

3. Let $S$ be the set of all vertices $v \in V$ that has at least one in-neighbor (w.r.t. $G$) in $I$
   and is labelled good by the neighbor(s). Output $S$.
```

It is obvious to see that the algorithm runs in polynomial time.

Correctness Intuition. Before we proceed with the formal proof of correctness, let us briefly give an informal intuition behind the proof. First, Step 0 simply helps us deal with the "trivial" case where $|A|$ is less than[6] say $0.3t$. In this case, we can simply output the whole vertex set V; this is indeed what Step 0 does. Thus, from this point onward, we may assume that $|A| \geqslant 0.3t$.

Next, a priori, it is not even clear that I must be non-empty. To see this, let us first recall a classic result of Nemhauser and Trotter that an extreme point solution of LP-IS is always *half-integral*, meaning that $x_v \in \{0, 1/2, 1\}$ for every vertex v.

Theorem 3.5 ([NT74]) *In any extreme point solution of* LP-IS*, $x_v \in \{0, 1/2, 1\}$ for all $v \in V$.*

Now, if I were empty, then we would have $x_v \in \{0, 1/2\}$, which would imply that the optimum of LP-IS is at most $|V|/2 = t + 0.5$. This would be a contradiction to Observation 3.3.

Next, let us consider the set $T = \{v \mid x^*_v = 0\}$. It is well-known that $|I| \geqslant |T|$; this can be easily seen because otherwise we can instead assign 1 to T and 0 to I and obtain a valid LP solution with larger objective value. (In fact, we will use a stronger property between the two sets below.)

For simplicity of exposition, let us assume for now that I only contains honest players, i.e., $I \subseteq H$, and that T only contains active adversary, i.e., $T \subseteq A$. Observe that, due to condition (3.4) of the relaxation, we must have $N_{G^{\mathrm{conf}}}(I) \subseteq T$. Notice also that, since every honest player labels its out-neighbor in A as "bad", we must have $N_{G^{\mathrm{conf}}}(I) \supseteq (N^{out}_G(I) \cap A)$. From this and from the bound $|I| \geqslant |T|$ in the previous paragraph, it must be that $|I| \geqslant |N^{out}_G(I) \cap A|$. However, recall that the edges from I to A are ("essentially") random of degree $\Omega(d) \geqslant 100 \log t$. Such a

[6]The constant 0.3 here can be replaced with any constant less than 1/3. We only use 0.3 to avoid introducing an additional parameter.

"non-expansion" condition can only hold when $N_G^{out}(I) \cap A$ already contains all but $o(1)$ fraction of A. In other words, we have $|I| \geqslant |N_G^{out}(I) \cap A| \geqslant (1 - o(1))|A| \geqslant (0.3 - o(1))t$, where the last inequality comes from our assumption that $|A| \geqslant 0.3t$.

Now, note that we never output vertices from $N_G^{out}(I) \cap A$ because they are already labelled as bad by at least one vertex in I. This means that $S \cap A$ is small (i.e. $o(t)$). On the other hand, we always output all vertices in $N_G^{out}(I) \cap (H \cap P)$ because they are labelled good by at least one vertex in I; since we conclude that $|I| \geqslant (0.1 - o(1))t$ in the previous paragraph, it follows from vertex expansion of random graphs that $|N_G^{out}(I) \cap (H \cap P)| \geqslant |H \cup P| - o(t) \geqslant (t+1) + |P| - o(t)$. By a more careful calculation of the terms $o(t)$, it is then possible to show that (3.1) holds.

To turn the above intuition into a formal proof, we not only have to make the calculations more precise, but we also need to deal with the case where $I \nsubseteq H$ (or $T \nsubseteq A$). Nevertheless, we can still show, using a more general structural result (see Lemma 3.10), that $I \cap H$ still satisfies "non-expansion". This allows the proof to go through in a similar manner.

3.1 Proof of Correctness of the Algorithm

We now give a formal proof of correctness of our algorithm. We will need several additional notations:

- Once again, let $I = \{v \in V \mid x_v^* = 1\}$ and $T = \{v \in V \mid x_v^* = 0\}$. Moreover , let $R = \{v \in V \mid x_v^* = 1/2\}$.
- Let I_H, T_H and R_H denote $I \cap H, T \cap H$ and $R \cap H$ respectively. Similarly, let $\overline{I_H}, \overline{T_H}$ and $\overline{R_H}$ denote $I \setminus H$, $T \setminus H$ and $R \setminus H$ respectively.

We will prove our main theorem for the constants $C = 10^{10}$ and $\beta = 10^{-10}$. It is henceforth assumed that $d \geqslant C \log t$, and this will not be explicitly stated. We remark that we make no attempt in optimizing these constants and it is likely that they can be reduced substantially.

Step 0: Dealing with the trivial case. As stated earlier, Step 0 in our algorithm helps us take care of the "trivial" case where A is already small. In particular, we can show that, if $|A| \leqslant 0.3t$, then the algorithm w.h.p. simply outputs V, which is a correct output in this case. Moreover, it is not hard to see that, when A is larger than $t/3$ and V is the wrong answer, then we do not terminate in this step and proceed to the remaining part of the algorithm. This is encapsulated in the following lemma.

Lemma 3.6 *When $|A| \leqslant 0.3t$, our algorithm outputs V and terminates at Step 0 with probability $1 - O(e^{-\beta d})$. On the other hand, when $|A| > t/3$, our algorithm terminates at Step 0 with probability only $O(e^{-\beta d})$.*

It turns out that the above lemma follows easily from the concentration of the number of bad labels given by each honest vertex. This concentration is stated and proved below.

Observation 3.7 *With probability $1 - O(e^{-\beta d})$, for all vertices $u \in H$, we have $(\mu - 0.001)d \leqslant |N_G^{out}(u) \cap A| \leqslant (\mu + 0.001)d$ where $\mu = \frac{|A|}{2t}$.*

Proof We will only prove that $|N_G^{out}(u) \cap A| \geqslant (\mu - 0.001)d$ for every $u \in H$ with high probability. The upper bound can be shown analogously. Note that our desired bound is obvious when $\mu \leqslant 0.001$. Hence, we may assume that $\mu \geqslant 0.001$, or equivalently $|A| \geqslant 0.002t$.

Let us fix $u \in H$ and a vertex $v \in A$. The probability that v belongs to $N_G^{out}(u)$ is exactly $\frac{d}{2t}$. Moreover, from how the graph is generated, the events $v \in N_G^{out}(u)$ for different v's are independent. Hence, Chernoff bounds implies that

$$\begin{aligned}\Pr\left[|N_G^{out}(u) \cap A| < 0.999 \cdot \left(\frac{d}{2t} \cdot |A|\right)\right] &\leqslant \exp\left(-\frac{10^{-6}}{2} \cdot \frac{d}{2t} \cdot |A|\right) \\ &\leqslant \exp\left(-\frac{10^{-6}}{2} \cdot \frac{d}{2t} \cdot 0.002t\right) \\ &\leqslant t^{-2} \cdot e^{-\beta d},\end{aligned}$$

where the second and third inequalities follow from $|A| \geqslant 0.002t$ and $d \geqslant 10^{10} \log t$ respectively. Furthermore, observe that $0.999 \cdot \left(\frac{d}{2t} \cdot |A|\right) = 0.999\mu d \geqslant (\mu - 0.001)d$. Hence, we have $\Pr\left[|N_G^{out}(u) \cap A| < (\mu - 0.001)d\right] \leqslant t^{-2} \cdot e^{-\beta d}$. Using union bound over all $u \in H$ concludes our proof.

■

Now that we have proved the concentration, we can prove Lemma 3.6 simply as follows.

Proof of Lemma 3.6: Suppose that $|A| \leqslant 0.3t$. Then, from Observation 3.7, w.p. $1 - O(e^{-\beta d})$ we have $|N_G^{out}(u) \cap A| \leqslant \left(\frac{|A|}{2t} + 0.001\right) d < 0.16d$ for all $u \in H$. Notice that $N_G^{out}(u) \cap A$ is exactly the set of vertices for which u labels bad. As a result, for these $|H| \geqslant t + 1$ vertices, they label bad to at most $0.16d$ vertices. This means that the condition in Step 0 is satisfied and the algorithm outputs V.

On the other hand, if $|A| > t/3$, then Observation 3.7 gives the bound $|N_G^{out}(u) \cap A| \geqslant \left(\frac{|A|}{2t} - 0.001\right) d > 0.16d$ with probability $1 - O(e^{\beta d})$. When this event occurs, each $u \in H$ labels more than $0.16d$ vertices as bad. Thus, the condition in Step 0 is not satisfied in this case.

■

Step I: Non-Expansion of I_H to A. The first step of the remaining part of the proof is to show that the set I_H does not (vertex-)expand in A (w.r.t out-edges in G), as stated below. We remark here that it also implicitly implies that $I_H \neq \emptyset$.

Lemma 3.8 $|I_H| > |N_G^{out}(I_H) \cap A|$.

To prove non-expansion of I_H, we will resort to a structural result regarding an extreme point LP solution. It is easiest to state in terms of *crown* as defined below [CFJ04, ACF+04]:

Definition 3.9 *For any undirected graph $F = (V, E_F)$ and disjoint subsets $I, T \subseteq F$, (I, T) is said to be a* crown *of F if (i) $T = N(I)$ and (ii) there is a matching between T and I such that all vertices in T are matched.*

Lemma 3.10 ([AFLS07]) *For any undirected graph $F = (V, E_F)$, let $\{x_v^*\}_{v \in V}$ be an extreme point solution, and let $I = \{v \in V \mid x_v^* = 1\}$ and $T = \{v \in V \mid x_v^* = 0\}$. Then, (I, T) forms a crown (w.r.t F).*

Notice that if (I, T) is a crown, then it must be that $|I| \geqslant |T|$ due to (ii). Hence, the above result is stronger than the one we used in the informal exposition.

Proof of Lemma 3.8: First, we claim that

$$|R_H| \leqslant |\overline{R_H}|. \tag{3.5}$$

This is because otherwise we can instead set $x_v^* = 1$ for $v \in R_H$ and $x_v^* = 0$ for $v \in \overline{R_H}$, which would give an LP solution with higher value. (Note that this is a valid LP solution because R_H is an independent set from Observation 3.3, and all neighbors of I lie in T.)

From Lemma 3.10, (I, T) forms a crown. Consider a matching from T to I such that all vertices in T are matched (which is guaranteed to exist from the definition of a crown). Notice that there is no edge from T_H to I_H in the graph G^{conf}; hence, all vertices in T_H must be matched to vertices in $\overline{I_H}$. In other words, we have

$$|T_H| \leqslant |\overline{I_H}|. \tag{3.6}$$

From (3.5) and (3.6), we have

$$|I_H| = |H| - |T_H| - |R_H| > t - |\overline{R_H}| - |\overline{I_H}| = |\overline{T_H}|.$$

Finally, observe that $(N_G^{out}(I_H) \cap A) \subseteq (N_{G^{\text{conf}}}(I_H) \cap A) \subseteq \overline{T_H}$, which yields the desired bound. ■

Step II: Expansion of Subsets of H in G. Similar to the outline, the second step of the proof is to observe that most subsets $X \subseteq H$ expands very well into A (or V), with respect to out-edges in G. The reason is simply that the graph from H to these sets are (essentially) random bipartite graphs of out-degree $\Omega(d)$. Due to technical reasons, we will also state the vertex expansion properties in terms of the in-degree graphs to V. The formal statement is as follows.

Lemma 3.11 *Suppose that $|A| \geqslant 0.3t$. Then, with probability $1 - O(e^{-\beta d})$, the following holds:*

1. *For any set $X \subseteq H$ such that $|X| \leqslant 0.1t$, we have*

$$|A \cap N_G^{out}(X)| \geqslant |X|. \tag{3.7}$$

2. *For any set $W \subseteq V$, we have*

$$|H \cap N_G^{in}(W)| \geqslant \min\{|H| - 0.1t, 10|W|\}. \tag{3.8}$$

We remark here that the above lemma is the main place we use $|A| \geqslant \Omega(t)$ as guaranteed from Step 0; otherwise the inequality (3.7) may not be true (some vertex in H might not even have an outgoing edge to A at all if A is too small).

The proof is via a standard approach to prove vertex/edge expansion of graph: we bound the probability that each neighbor is a subset of a too-small set and then use union bound in the end.

Proof

1. Let us consider any sets $X \subseteq H$ and $Y \subseteq A$. For any vertex $u \in X$, the probability that $(N_G^{out}(u) \cap A) \subseteq Y$ is exactly $\left(\frac{|H|-1+|Y|}{2t}\right)^d$. Since the events $N_G^{out}(u) \subseteq Y$ are independent for all $u \in X$, we have

$$\Pr[(N_G^{out}(X) \cap A) \subseteq Y] = \left(\frac{|H|-1+|Y|}{2t}\right)^{d|X|} \leqslant \left(1 - \frac{|A \setminus Y|}{2t}\right)^{d|X|}.$$

   Hence, the undesired event happens with probability at most

$$\begin{aligned}\sum_{\substack{X \subseteq H, Y \subseteq A \\ |X| \leqslant 0.1t, |Y|=|X|-1}} \Pr[(N_G^{out}(X) \cap A) \subseteq Y] &\leqslant \sum_{\substack{X \subseteq H, Y \subseteq A \\ |X| \leqslant 0.1t, |Y|=|X|-1}} \left(1 - \frac{|A \setminus Y|}{2t}\right)^{d|X|} \\ &\leqslant \sum_{\substack{X \subseteq H, Y \subseteq A \\ |X| \leqslant 0.1t, |Y|=|X|-1}} (0.55)^{d|X|} \\ &= \sum_{i=1}^{0.1t} \binom{|H|}{i} (0.55)^{d|X|} \\ &\leqslant \sum_{i=1}^{0.1t} (2t+1)^i \, (0.55)^{di} \\ &\leqslant \sum_{i=1}^{0.1t} t^{-2} \cdot e^{-\beta d} \\ &= O(e^{-\beta d}),\end{aligned}$$

   where the second inequality follows from $|A| \geqslant 0.3t$ and $|Y| \leqslant 0.1t$ and we use our choice of $d \geqslant 10^{10} \log t$ in the last inequality.

2. Consider any set $W \subseteq V$ and $X \subseteq H$. We will bound the probability that $(H \cap N_G^{in}(W))$ is a subset of X. Recall from our definition that every vertex has a self-loop. Hence, if $(W \cap H) \nsubseteq X$, it is immediate that $\Pr[(H \cap N_G^{in}(W)) \subseteq X] = 0$.

   Now, for the case $(W \cap H) \subseteq X$, we can bound $\Pr[(H \cap N_G^{in}(W)) \subseteq X]$ as follows. First, notice that each vertex $u \in (H \setminus X)$, u does *not* belongs to $N_G^{in}(W)$ (or equivalently $N_G^{out}(u) \cap W = \emptyset$) with probability exactly $\left(1 - \frac{|W|}{2t}\right)^d$. Since the events $u \notin N_G^{in}(W)$ are independent for all $u \in (H \setminus X)$, we have

$$\Pr[(H \cap N_G^{in}(W)) \subseteq X] = \left(1 - \frac{|W|}{2t}\right)^{d \cdot |H \setminus X|} \leqslant e^{-\frac{d|W| \cdot |H \setminus X|}{2t}} = e^{-\frac{d|W| \cdot (|H| - |X|)}{2t}}.$$

   For convenience, let $\mu(|W|)$ denote $\lceil \min\{|H| - 0.1t, 10|W|\} \rceil - 1$. From union bound and the previous inequality, the probability of the undesired event is at most

$$\begin{aligned}\sum_{\substack{W\subseteq V, X\subseteq H\\ |X|=\mu(|W|)}} \Pr[(H\cap N_G^{in}(W))\subseteq X] &\leqslant \sum_{\substack{W\subseteq V, X\subseteq H\\ |X|=\mu(|W|)}} e^{-\frac{d\cdot|W|\cdot(|H|-|X|)}{2t}}\\ &\leqslant \sum_{\substack{W\subseteq V, X\subseteq H\\ |X|=\mu(|W|)}} e^{-0.05d|W|}\\ &= \sum_{i=1}^{2t+1}\binom{2t+1}{i}\binom{|H|}{\mu(i)}e^{-0.05di}\\ &\leqslant \sum_{i=1}^{2t+1}(2t+1)^{11i}e^{-0.05di}\\ &\leqslant \sum_{i=1}^{2t+1} t^{-2}\cdot e^{-\beta d}\\ &= O(e^{-\beta d})\end{aligned}$$

where the second inequality is due to $\mu(|W|)\leqslant |H|-0.1t$ and the last inequality follows from $d\geqslant 10^{10}\log t$. This completes our proof.

■

We can deduce from the above lemma the following corollary, which will be more convenient to use in the main proof.

Corollary 3.12 *Suppose that $|A|\geqslant 0.3t$. Then, with probability $1-O(e^{-\beta d})$, for any set $X\subseteq H$, at least one of the following must hold: (i) $|X|\leqslant |N_G^{out}(X)\cap A|$ or (ii) $|N_G^{out}(X)\cap(H\cup P)|\geqslant (t+1)+2\cdot|A\setminus N_G^{out}(X)|$.*

Proof Suppose for the sake of contradiction that there exists $X\subseteq H$ that violates both inequalities. Since X violates (i) and from the first item (i.e. (3.7)) of Lemma 3.11, we must have $|X|>0.1t$, which means

$$|H\setminus X|<|H|-0.1t. \tag{3.9}$$

Let $Y=(A\setminus N_G^{out}(X))$ and $k=|Y|$. From the violation of (i), we have

$$|X|\geqslant |N_G^{out}(X)\cap A|+1=|A|-k+1=(2t+1-|H|-|P|)-k+1$$

From the above, we have

$$|H\setminus X|=|H|-|X|\leqslant |P|+k+2(|H|-(t+1)).$$

For convenience, we let $\Gamma=|H|-(t+1)$. We may write the above inequality as

$$|H\setminus X|\leqslant |P|+k+2\Gamma \tag{3.10}$$

Now, from (3.8) in Lemma 3.11 with $W=Y$, we have

$$|H\setminus X|\geqslant |N_G^{in}(Y)\cap H|\geqslant \min\{|H|-0.1t, 10k\}.$$

From (3.9), it cannot be that $|H \setminus X| \geqslant |H| - 0.1t$. As a result, we have

$$|H \setminus X| \geqslant 10k. \tag{3.11}$$

Similarly, let $Z = (H \cup P) \setminus N_G^{out}(X)$. From the violation of (ii), we have

$$|Z| = |H| + |P| - |N_G^{out}(X) \cap (H \cup P)| \geqslant |H| + |P| - t - 2k = |P| - 2k + 1 + \Gamma. \tag{3.12}$$

Moreover, from (3.8) in Lemma 3.11 with $W = Z$, we have

$$|H \setminus X| \geqslant |N_G^{in}(Z) \cap H| \geqslant \min\{|H| - 0.1t, 10|Z|\}.$$

Once again, (3.9) implies that $|H \setminus X|$ cannot be at least $|H| - 0.1t$. Hence, we have

$$|H \setminus X| \geqslant 10|Z| \overset{(3.12)}{\geqslant} 10(|P| - 2k + 1 + \Gamma). \tag{3.13}$$

By combining (3.10), (3.11) and (3.13), we arrive at

$$\begin{aligned} |P| + k + 2\Gamma \overset{(3.10)}{\geqslant} |H \setminus X| &= 0.8|H \setminus X| + 0.2|H \setminus X| \\ &\overset{(3.11),(3.13)}{\geqslant} 8k + 2(|P| - 2k + 1 + \Gamma) \\ &= 2|P| + 4k + 1 + 2\Gamma, \end{aligned}$$

a contradiction.

■

Step III: Putting things together. With the above lemmas ready, we now prove our main theorem by simply plugging them together.

Proof of Theorem 3.1: From Lemma 3.6, if $|A| < 0.3t$, then we output the entire vertex set V and terminates with probability $1 - O(e^{-\beta d})$; this is a correct output. Moreover, Lemma 3.6 also ensures that we w.p. $1 - O(e^{\beta d})$ do not terminate here with an incorrect output.

We may now assume for the rest of the proof that $|A| \geqslant 0.3t$. From Lemma 3.8, we have $|I_H| > |N_G^{out}(I_H) \cap A|$. As a result, from Corollary 3.12, we with probability $1 - O(e^{-\beta d})$ must have

$$|N_G^{out}(I_H) \cap (H \cup P)| \geqslant (t + 1) + 2 \cdot |A \setminus N_G^{out}(I_H)|. \tag{3.14}$$

Now, observe that, every vertex in $N_G^{out}(I_H) \cap (H \cup P)$ is labelled good by at least one vertex in $I_H \subseteq I$; hence, they will be included in S. In other words, $S \cap (H \cup P) \supseteq N_G^{out}(I_H) \cap (H \cup P)$.

On the other hand, all vertices in $N_G^{out}(I_H) \cap A$ are labelled bad by at least one vertex in $I_H \subseteq I$; hence, they will not be included in S. In other words, we have $S \cap A \subseteq (A \setminus N_G^{out}(I_H))$.

As a result, we arrive at

$$|S \cap (H \cup P)| \geqslant |N_G^{out}(I_H) \cap (H \cup P)| \overset{(3.14)}{\geqslant} (t + 1) + 2 \cdot |A \setminus N_G^{out}(I_H)| \geqslant (t + 1) + 2 \cdot |S \cap A|,$$

which concludes our proof. ■

4 Construction of Robust Secret Sharing

Let λ be the robustness parameter. In this section, we give a construction of robust secret sharing for messages of length m with share size $m + O(\lambda \log n(\log n + \log m))$. In Section 4.1, we first give a construction of a basic robust secret sharing scheme with share size $m + O(\lambda^2 + \lambda(\log n + \log m) + \log^2 n + \log n \log m)$. In the Section 4.2, we use parallel repetition (as was done by Bishop et al [BPRW16]) to improve the share size of our construction to $m + O(\lambda \log n(\log n + \log m))$.

4.1 Basic Scheme

The construction is described in Figure 1 and we show the following theorem.[7]

Theorem 4.1 *For some $t, d, \lambda, \rho \in \mathbb{N}$, $\varepsilon_2, \varepsilon_3 \in [0,1]$, $n = 2t+1$, and message space $\mathcal{M}$, assume we have the following:*

- *A $(t+1, n, 0, 0)$ secret sharing scheme* $(\mathsf{Share}, \mathsf{Rec})$ *for $\mathcal{M}$ that is $(0, s, \lfloor (s-(t+1))/2 \rfloor)$-robust for any $s \geqslant t+1$. Further, the shares are strings in $\{0,1\}^\rho$.*
- *A $(2d, \varepsilon_2)$-secure private MAC scheme* $(\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$ *for message space $\{0,1\}^\rho$ with randomness length ν, key length κ and tag length λ.*
- *The vertex identification algorithm* $\mathsf{VertexID}$ *from Theorem 3.1, when run on graphs with out-degree d, has error probability at most ε_3.*

Then, the construction in Figure 1 is a $(t+1, n, ne^{-d/3}, 0)$ secret sharing scheme for $\mathcal{M}$ that is $(nd\varepsilon_2 + \varepsilon_3 + ne^{-d/3}, n, t)$-robust. The size of each share is $(\rho + \nu + d\lceil \log n \rceil + d(\kappa + \lambda))$ bits.

Proof of Theorem 4.1: The share size may be verified by inspection. We first show the correctness and privacy properties of our construction and finally show its robustness.

Correctness. Note that in an honest execution of the sharing and the reconstruction algorithm, in the absence of any corruptions, every edge in the graph G will be labeled as good. Thus, the algorithm $\mathsf{VertexID}$ will output $V = [n]$. It now follows from the perfect correctness of Rec that the secret output by $\mathsf{BasRobRec}$ will be equal to m^* with probability 1. We will now bound the probability that m^* is not equal to m.

We estimate the probability that the chosen multisets $\{S^i\}_{i\in[n]}$ has the property such that for $v \in [n]$, the multiplicity of v in $S^1 \| \ldots \| S^n$ is at most $2d$. Let us fix a party $v \in [n]$, and call the event that its multiplicity is more than $2d$ as Bad_v. For any $i \neq v$, v might get selected (possibly multiple times) in the multiset S^i. Thus, there are totally $d(n-1)$ random draws where v might get selected. For any $i \in [d(n-1)]$, let X_i be the indicator random variable which is 1 if and only if v is selected in the i-th draw. Now, for any $i \in [d(n-1)]$,

$$\Pr[X_i = 1] = \frac{1}{n-1} \tag{4.1}$$

Then, Bad_v occurs if $\sum_i X_i > 2d$. The variables $\{X_i\}_i$ are independent and hence from Chernoff bounds,

$$\Pr[\mathsf{Bad}_v] = \Pr[\sum_i X_i > 2d] \leqslant e^{-d/3} \tag{4.2}$$

[7]Recall the definition of multiplicity of a multiset and $\|$ from Section 2.

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

1. For each $i \in [n]$, select a multiset $S^i \subseteq [n] \setminus \{i\}$ of size d uniformly at random with replacement.
2. If there is an $i' \in [n]$ such that the multiplicity of i' in $(S^1 \| S^2 \| \ldots \| S^n)$ is greater than $2d$, select $m^* \leftarrow \mathcal{M}$. Else, set $m^* = m$.
3. Run $\mathsf{Share}(m^*)$ to obtain the shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$.
4. For each $i \in [n]$, choose $r_i \leftarrow \{0,1\}^\nu$.
5. For each $i \in [n]$ and each $j \in [d]$ do:
   (a) Let v^i_j be the j-th element in the ordered multiset S^i.
   (b) Choose a private MAC key $k^j_{i \to v^i_j} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
   (c) Compute $\sigma^j_{i \to v^i_j} \leftarrow \mathsf{Tag}(k^j_{i \to v^i_j}, (\mathsf{Sh}_{v^i_j}, r_{v^i_j}))$.
6. Set $\mathsf{share}_i = (\mathsf{Sh}_i, r_i, S^i, \{k^j_{i \to v^i_j}, \sigma^j_{i \to v^i_j}\}_{j \in [d]})$.

$\mathsf{BasRobRec}$: The reconstruction algorithm proceeds in two rounds.

- **Round-1:**
  1. For each $i \in [n]$, party i broadcasts (Sh_i, r_i) to every other party and initializes an empty list N_i.
  2. For every $j \in [d]$, party i does:
     (a) Let v^i_j be the j-th element in the ordered multiset S^i.
     (b) Check if $\mathsf{Verify}(k^j_{i \to v^i_j}, (\mathsf{Sh}_{v^i_j}, r_{v^i_j}), \sigma^j_{i \to v^i_j}) = 1$.
     (c) If the verification passes, party i adds $((i, v^i_j), \text{good})$ to N_i. Else, it adds $((i, v^i_j), \text{bad})$ to N_i.
- **Round-2:**
  1. For each $i \in [n]$, party i broadcasts N_i to every other party and initializes an empty graph G and an empty labeling L.
  2. For every $i \in [n]$ and every entry $((i,v), \mathsf{lab}_{i,v}) \in N_i$, the parties add the edge (i,v) in G and set $L(i,v) = \mathsf{lab}_{i,v}$.
  3. The parties (locally) run the algorithm $\mathsf{VertexID}$ on G and L to obtain the vertex set S.
  4. The parties then (locally) run $\mathsf{Rec}(\{\mathsf{Sh}_i\}_{i \in S})$ to obtain the secret m.

Figure 1: Basic Construction of Robust Secret Sharing (using terminology from Theorem 4.1)

Let Bad be the event that there exists at least one $v \in [n]$ such that Bad_v happens. By union bound,

$$\Pr[\mathsf{Bad}] \leqslant \sum_{v} \Pr[\mathsf{Bad}_v] = ne^{-d/3} \tag{4.3}$$

Thus, with probability at least $1 - ne^{-d/3}$, the chosen multisets $\{S^i\}_{i\in[n]}$ satisfies the property the multiplicity of every $v \in [n]$ in $S^1\|\ldots\|S^n$ is at most $2d$. Notice that when this happens, $m^* = m$. Thus, the correctness error is at most $ne^{-d/3}$. We will call $\{S^i\}_{i\in[n]}$ that satisfies the above property to be *bounded.*

Privacy. To show perfect privacy, we need to argue that for every set U of size at most t, for every pair of secrets $m_0, m_1 \in M$, the distributions of $\mathsf{BasRobShare}(m_0)_U$ and $\mathsf{BasRobShare}(m_1)_U$ are identical.[8] It is sufficient to show that this is in the case when $\{S^i\}$ is bounded, and when it is not bounded, by design the output of $\mathsf{BasRobShare}$ is independent of the message being shared, and so the shares of U are identical.

When $\{S^i\}$ is bounded, we show this by the following hybrid argument. Fix any choice of the multisets $\{S^i\}_{i\in[n]}$ such that it is bounded. For every $i \in [n]$, we define T^i to be the sequence of $(v, j) \in [n] \times [d]$ such that the j-th entry of the multiset S^v is equal to i. From the choice of the fixed $\{S^i\}_{i\in[n]}$, each $|T^i| \leqslant 2d$. We first fix all the MAC keys chosen during the share phase, and make the following argument for any such set of keys. We now argue that $\mathsf{BasRobShare}(m_0)_U$ is identical to $\mathsf{BasRobShare}(m_1)_U$, going through the following hybrid distributions over the shares of U:

- Hyb_1 : This is the same as $\mathsf{BasRobShare}(m_0)_U$.
- Hyb_2 : In this hybrid, during the share phase, for every $i \notin U$, we generate $\sigma^j_{v\to i}$ (for any $j \in [d]$ and $v \in [n]$) as $\mathsf{Tag}(k^j_{v\to i}, (0^\rho, r_i))$. We finally output the shares corresponding to U. Note that Hyb_1 is identical to Hyb_2 from the perfect privacy property of the private MAC scheme, which is $(2d, \varepsilon_2)$-secure, since $|T^i| \leqslant 2d$ for every $i \in [n]$.
- Hyb_3 : In this hybrid, during the share phase, instead of running $\mathsf{Share}(m^*)$ to get the Sh_i's, we run $\mathsf{Share}(0)$.[9] We run the rest of the sharing normally and output the shares corresponding to U. Hyb_3 is identical to Hyb_2 by the perfect privacy of the secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$.

Note that via a similar argument we can show that $\mathsf{BasRobShare}(m_1)_U$ is also identical to Hyb_3, and thus to $\mathsf{BasRobShare}(m_0)_U$, proving perfect privacy.

Robustness. We now argue the robustness of our construction. Consider an interactive adversary $\mathcal{A}$ that adaptively corrupts a set T of parties. The adversary is given $\{\mathsf{share}_i\}_{i\in T}$. Let H be the set of honest parties. Consider the interactive reconstruction algorithm.

- In the first round of the reconstruction algorithm, the party i broadcasts (Sh_i, r_i) to every other party. Now, every party $i \in T$ might broadcast the correct (Sh_i, r_i) or a modified (Sh'_i, r'_i). Based on this, we partition T into two sets P and A. P consists of the parties that send

[8]From Remark 2.2, we also satisfy the stronger notion of adaptive privacy.

[9]0 denoting some universally fixed element in $\mathcal{M}$

the unmodified (Sh_i, r_i) whereas the parties in A modify the shares and send $(\mathsf{Sh}'_i, r'_i) \neq (\mathsf{Sh}_i, r_i)$. Note that at the end of the first round, the adversarial parties learn no information about the multisets S^i of the honest parties and conditioned on the information available to the adversary at the end of the first round, the multisets S^i are still random.

- At the end of the first round, the parties will verify the tags of the MAC. Every $i \in H$ will generate the set N_i as follows:
  - Let $S^i = (v_1^i, \dots, v_d^i)$.
  - For every $j \in [d]$ such that $v_j^i \in H \cup P$, party i adds $((i, v_j^i), \text{good})$ to N_i.
  - For every $j \in [d]$ such that $v_j^i \in A$, party i adds $((i, v_j^i), \text{bad})$ to N_i except with probability at most ε_2. The ε_2 error probability follows directly from the ε_2-unforgeability of the private MAC.

  By standard union bound, the probability that there exists an $i \in H$ such that for some $v \in A$, party i adds $((i, v), \text{good})$ to N_i is at most $nd\varepsilon_2$ since each multiset S^i has size d.

- Conditioned on the above event not happening, the graph $G = (V, E)$ with $V = [n]$ and the edge labeling L is effectively generated as follows.
  1. The adversary partitions V into H, P, A where $|H| \geqslant t + 1$.
  2. For every $u \in H$, choose a multiset S^u uniformly at random from $[n] \setminus \{u\}$ with replacement and let $S^u = (v_1^u, \dots, v_d^u)$. For every $j \in [d]$, add an edge (u, v_j^u) and set $L(u, v_j^u) = \text{good}$ if and only if $v_j^u \in H \cup P$. This is identically distributed to the distribution where we choose S^u uniformly at random during the sharing phase since at the end of the first round, the adversary learns no information about the multisets of the honest parties.
  3. The outgoing edges and their labels of $A \cup P$ can be generated adversarially after looking at the outgoing edges and the labels of the vertices in H.

  This is exactly same as the graph generation procedure given in Section 3.

- It now follows from the correctness of the $\mathsf{VertexID}$ algorithm that its output S when run on this graph satisfies the property that $|S \cap (H \cup P)| \geqslant (t + 1) + 2 \cdot |S \cap A|$ except with probability ε_3. The fact that $\mathsf{Rec}(\{\mathsf{Sh}_j\}_{j \in S}) = m^*$ follows from robustness of secret sharing (Fact 2.6).

- Finally, as in the correctness argument, m^* is equal to the actual secret m except with probability $ne^{-d/3}$. Thus, by the union bound, the probability of error of the whole reconstruction procedure is at most $(nd\varepsilon_2 + \varepsilon_3 + ne^{-d/3})$.

This completes the proof of the theorem.

■

4.1.1 Instantiation

We now provide the following instantiation of the building blocks of our robust secret sharing scheme. Let us fix the robustness parameter λ, and the length m of the secret to be shared (if $m < \lceil \log n \rceil$, replace it with $\lceil \log n \rceil$ in the following).

- We set $d = 10\lambda/\beta + 3C \log n/\beta$, where C (> 1) and β $(\in (0,1))$ are constants from Theorem 3.1.
- We instantiate the secret sharing with Shamir secret sharing over GF$[2^m]$. This gives $\rho = m$.
- We instantiate the private MAC with the (ℓ, ε_2)-secure MAC for message space $\{0,1\}^\rho$ from Theorem 2.2, with $\ell = 2d$ and $\varepsilon_2 = 2^{-\lambda}/(2nd)$. The randomness, tag and key lengths are, respectively, $2d(\lambda + \log(2nd(\rho + 2d)))$, $\lambda + \log(2nd(\rho + 2d))$, and $2(\lambda + \log(2nd(\rho + 2d)))$.
- The $\mathsf{VertexID}$ algorithm has error probability $\varepsilon_3 < 2^{-10\lambda}$.

The robustness error is $nd\varepsilon_2 + \varepsilon_3 + ne^{-d/3} < 2^{-\lambda}/2 + 2^{-10\lambda} + n2^{-3\lambda - \log n} \leqslant 2^{-\lambda}$. The correctness error is $ne^{-d/3} \leqslant 2^{-\lambda}$. The size of each share is $\rho + 2d(\lambda + \log(2nd(\rho + 2d))) + d \log n + 3d(\lambda + \log(2nd(\rho+2d))) = m + 5d\lambda + O(d \log(2nd(m+2d))) = m + O(\lambda^2) + O(\lambda \log n) + O((\lambda + \log n)(\log n + \log \lambda + \log m)) = m + O(\lambda^2 + \lambda(\log n + \log m) + \log^2 n + \log n \log m)$.

Corollary 4.2 *For any $\lambda, t, m, n \in \mathbb{N}$ with $n = 2t+1$, there exists a $(t+1, n, 2^{-\lambda}, 0)$-secret sharing scheme that is $(2^{-\lambda}, n, t)$-robust and, for secrets of length m bits, has shares of size $m + O(\lambda^2 + \lambda(\log n + \log m) + \log^2 n + \log n \log m)$ bits.*

4.2 Improved Parameters via Parallel Repetition

In this subsection, we improve the share size of our basic construction to $m + O(\lambda \log n(\log n + \log m))$ to achieve robustness error of $2^{-\lambda}$ via parallel repetition. This is similar to the ideas explained in [BPRW16]. Before we describe the construction, we start with some notation.

Notation. We split $\mathsf{BasRobRec}$ into two steps. The first step $\mathsf{BasRobRec}_1$ is an interactive protocol comprising of the first two rounds of $\mathsf{BasRobRec}$ and the output of the protocol is the set S which is the output of $\mathsf{VertexID}$ algorithm on the constructed graph G and the labeling L. The second step consists of running Rec on $\{\mathsf{Sh}_i\}_{i \in S}$ and outputting the message.

Construction. The construction of robust secret sharing $(\mathsf{RobShare}, \mathsf{RobRec})$ with improved parameters is described in Figure 2.

Theorem 4.3 *For any $\lambda, t, m, n \in \mathbb{N}$, with $n = 2t+1$, the construction in Figure 2 is a $(t+1, n, 0, 0)$ secret sharing scheme (with expected polynomial time sharing algortithm) for secrets of length m that is $(e^{-\lambda/24}, n, t)$-robust. The size of each share is $m + O(\lambda \log n(\log n + \log m))$.*

Proof We first show that the sharing algorithm is expected polynomial time. Notice that in any trial q, the probability that the multiplicity of an i' in $S^{q,1} \| S^{q,2} \| \ldots \| S^{q,n}$ is greater than $2d$, is bounded by $e^{-d/3}$ (from the correctness argument in Theorem 4.1). Thus, by standard union bound, the probability that there exists some i' with multiplicity more than $2d$ in $S^{q,1} \| S^{q,2} \| \ldots \| S^{q,n}$

Set $\lambda' = 2$, and $d = 10\lambda'/\beta + 3C \log n/\beta$, where C and β are the constants from Theorem 3.1. In the following, $(\mathsf{Share}, \mathsf{Rec})$ represents Shamir secret sharing over $\mathrm{GF}[2^m]$. The private MAC $(\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$ used for the message space $\{0,1\}^m$ is the (ℓ, ε)-secure MAC from Theorem 2.2 with $\ell = 2d$ and $\varepsilon = 2^{-\lambda'}/2nd$. The protocol $\mathsf{BasRobRec}$ is from the construction in Figure 1.

$\mathsf{RobShare}(m)$: To share a secret $m \in \{0,1\}^m$:

1. Run $\mathsf{Share}(m)$ to obtain the shares $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$.
2. For each q in 1 to λ do:
    - (a) For each $i \in [n]$, select a multiset $S^{q,i} \subseteq [n] \setminus \{i\}$ of size d uniformly at random with replacement.
    - (b) If there is an $i' \in [n]$ such that the multiplicity of i' in $S^{q,1} \| S^{q,2} \| \ldots \| S^{q,n}$ is greater than $2d$, then go back to step (2a).
3. For each $q \in [\lambda]$, $i \in [n]$ and each $j \in [d]$ do:
    - (a) For each $i \in [n]$, choose $r_i^q \leftarrow \{0,1\}^\nu$.
    - (b) Let $v_j^{q,i}$ be the j-th element in the ordered multiset $S^{q,i}$.
    - (c) Choose a private MAC key $k^{q,j}_{i \to v_j^{q,i}} \leftarrow \mathsf{KeyGen}$.
    - (d) Compute $\sigma^{q,j}_{i \to v_j^{q,i}} \leftarrow \mathsf{Tag}(k^{q,j}_{i \to v_j^{q,i}}, (\mathsf{Sh}_{v_j^{q,i}}, r^q_{v_j^{q,i}}))$.
4. Set $\mathsf{share}_i = \left(\mathsf{Sh}_i, \{r_i^q\}_{q \in [k]}, \{S^{q,i}\}_{q \in [k]}, \{k^{q,j}_{i \to v_j^{q,i}}, \sigma^{q,j}_{i \to v_j^{q,i}}\}_{j \in [d], q \in [k]} \right)$.

RobRec :

1. For each q in 1 to λ do in parallel:
    - (a) Run $\mathsf{BasRobRec}_1 \left(\left\{ \mathsf{Sh}_i, r_i^q, S^{q,i}, \{k^{q,j}_{i \to v_j^{q,i}}, \sigma^{q,j}_{i \to v_j^{q,i}}\}_{j \in [d]} \right\}_{i \in [n]} \right)$ to obtain the set Γ_q.
    - (b) Set $m_q := \mathsf{Rec}(\{\mathsf{Sh}_i\}_{i \in \Gamma_q})$.
2. If there is a majority value m in the sequence $(m_1, \ldots, m_\lambda)$ then output m. Else, output $\perp$.

Figure 2: Improved Construction of Robust Secret Sharing

is bounded by $ne^{-d/3} \leqslant 2^{-\lambda'} = 1/4$. Thus, in expectation, for each q, we find the required $S^{q,1} \| S^{q,2} \| \ldots \| S^{q,n}$ in a constant number of trials. We can also make this running time strictly polynomial at the expense of $2^{-\lambda}$ additional correctness and robustness error by limiting the number of trials to be $O(\lambda)$ and setting all the shares to be empty if these trials are not sufficient.

Privacy. The privacy of the construction is argued similarly to the basic construction. Fix any choice of the multisets $\{S^{q,i}\}_{i\in[n]}$ output by $\mathsf{RobShare}$. For every $q \in [\lambda]$, $i \in [n]$, we define $T^{q,i}$ to be the sequence of $(v, j) \in [n] \times [d]$ such that the j-th entry of the multiset $S^{q,v}$ is equal to i. Note that each $|T^{q,i}| \leqslant 2d$. We first fix all the MAC keys chosen during the share phase, and make the following argument for any such set of keys. We now argue that $\mathsf{RobShare}(m_0)_U$ is identical to $\mathsf{RobShare}(m_1)_U$.

- Hyb_1 : This is identical to $\mathsf{RobShare}(m_0)_U$.
- $\mathsf{Hyb}_{2,q}$: Let $(\mathsf{Sh}'_1, \ldots, \mathsf{Sh}'_n)$ be such that $\mathsf{Sh}'_U = \mathsf{Sh}_U$ and for any $i \notin U$, $\mathsf{Sh}'_i = 0$. For $q \in [\lambda]$, in this hybrid, during the share phase, for $q' < q$, we generate $\sigma^{q',j}_{v\to i}$ (for any $j \in [d]$ and $v \in [n]$) as $\mathsf{Tag}(k^{q',j}_{v\to i}, (\mathsf{Sh}'_i, r^q_i))$. For $q' \geqslant q$, the iterations are run normally as specified in $\mathsf{RobShare}$. We finally output the shares corresponding to U from all the iterations. Note that Hyb_1 is identical to $\mathsf{Hyb}_{2,1}$ and for any q, $\mathsf{Hyb}_{2,q}$ is identical to $\mathsf{Hyb}_{2,q+1}$ from the privacy property of the private MAC scheme, which is $(2d, \varepsilon_2)$-secure, since $|T^{q,i}| \leqslant 2d$ for every $i \in [n]$.
- Hyb_3 : In this hybrid, during the share phase, we generate Sh_U as a secret sharing of message 0 instead of m_0, and do the rest according to $\mathsf{Hyb}_{2,\lambda}$. We finally output the shares corresponding to U. Hyb_3 is identical to $\mathsf{Hyb}_{2,\lambda}$ by the privacy of the secret sharing scheme.

Similarly, we can also show that $\mathsf{RobShare}(m_1)_U$ is identical to Hyb_3, thus proving the privacy of the construction.

Robustness. Let us fix any choice of $(\mathsf{Sh}_1, \ldots .\mathsf{Sh}_n)$ that is in the support of $\mathsf{Share}(m)$ as well as $(r^q_1, \ldots, r^q_n)_{q\in[\lambda]}$. We will argue robustness conditioned on this fixing. Consider an interactive adversary $\mathcal{A}$ that adaptively corrupts a set T of parties. The adversary is given $\{\mathsf{share}_i\}_{i\in T}$. Let H be the set of honest parties. Consider the interactive reconstruction algorithm given in Figure 2. For every $q \in [\lambda]$, we will show that except with probability $2^{-\lambda'}$, m_q obtained during the reconstruction is equal to m.

- We first consider an alternate sharing algorithm where we remove the criterion that there exists no $i' \in [n]$ such that multiplicity of i' in $S^{q,1}\|S^{q,2}\|\ldots\|S^{q,n}$ is greater than $2d$. In particular, we also allow sets that do not satisfy this property. We note that the probability that the sets do not satisfy this property is at most $ne^{-d/3}$ and hence, the distribution of the shares in the real sharing and this alternate sharing is $ne^{-d/3}$ far apart in statistical distance.
- In the first round of the reconstruction algorithm, the party i broadcasts (Sh_i, r^q_i) to every other party for $q \in [\lambda]$. Now, every party $i \in T$ might broadcast the correct (Sh_i, r^q_i) or a modified (Sh'_i, r'^q_i). Based on this, we partition T into two sets P and A. P consists of the parties that send the unmodified (Sh_i, r^q_i) whereas the parties in A modify the shares and send $(\mathsf{Sh}'_i, r'^q_i) \neq (\mathsf{Sh}_i, r^q_i)$. Note that at the end of the first round, the adversarial parties learn no information about the multisets $S^{q,i}$ of the honest parties and conditioned on the information available to the adversary at the end of the first round, the multisets $S^{q,i}$ for every $i \in H$ are still random.
- At the end of the first round, the parties will verify the tags of the MAC. Every $i \in H$ will generate the set N^q_i as follows:

  - Let $S^{q,i} = (v_1^{q,i}, \dots, v_d^{q,i})$.
  - For every $j \in [d]$ such that $v_j^{q,i} \in H \cup P$, party i adds $((i, v_j^{q,i}), \text{good})$ to N_i^q.
  - For every $j \in [d]$ such that $v_j^{q,i} \in A$, party i adds $((i, v_j^{q,i}), \text{bad})$ to N_i^q except with probability at most ε. The ε error probability follows directly from the ε-unforgeability of the private MAC.

  By standard union bound, the probability that there exists an $i \in H$ such that for some $v \in A$, party i adds $((i, v), \text{good})$ to N_i is at most $nd\varepsilon$ since each multiset S^i has size d.

- Conditioned on the above event not happening, the graph $G = (V, E)$ with $V = [n]$ and the edge labeling L is effectively generated as follows.

  1. The adversary partitions V into H, P, A where $|H| \geqslant t + 1$.
  2. For every $u \in H$, choose a multiset $S^{q,u}$ uniformly at random from $[n] \setminus \{u\}$ with replacement and let $S^{q,u} = (v_1^{q,u}, \dots, v_d^{q,u})$. For every $j \in [d]$, add an edge $(u, v_j^{q,u})$ and set $L(u, v_j^{q,u}) = \text{good}$ if and only if $v_j^{q,u} \in H \cup P$. This is identically distributed to the distribution where we choose $S^{q,u}$ uniformly at random during the sharing phase since at the end of the first round, the adversary learns no information about the multisets of the honest parties.
  3. The outgoing edges and their labels of $A \cup P$ can be generated adversarially after looking at the outgoing edges and the labels of the vertices in H.

  This is exactly same as the graph generation procedure given in Section 3.

- It now follows from the correctness of the $\mathsf{VertexID}$ algorithm (Theorem 3.1) that its output Γ_q when run on this graph satisfies the property that $|\Gamma_q \cap (H \cup P)| \geqslant (t+1) + 2 \cdot |\Gamma_q \cap A|$ except with probability $2^{-10\lambda'}$. The fact that $\mathsf{Rec}(\{\mathsf{Sh}_j\}_{j \in \Gamma_q}) = m$ in this case follows from robustness of secret sharing (Fact 2.6 since the error is 0 and it is correct on every choice of $(\mathsf{Sh}_1, \dots, \mathsf{Sh}_n)$ in the support).

- Thus, the probability of error of the whole reconstruction procedure is at most $(nd\varepsilon + 2^{-10\lambda'} + ne^{-d/3}) \leqslant 2^{-\lambda'}$.

Thus, in each of the λ iterations, except with probability $2^{-\lambda'} = 1/4$, we recover the correct message m, whether there is an adversary corrupting t parties or not. And further, the iterations are independent of each other. From standard Chernoff bounds, it follows that except with probability $\exp(-\lambda/24)$, majority of the values in the sequence $(m_1, \dots, m_\lambda)$ will be equal to the message m.

Share size. Note that we have set $\lambda' = O(1)$ and $d = O(\log n)$. The share size is seen to be: $m + \lambda\left[2d(\lambda' + \log(2nd(m+2d))) + d\log n + 3d(\lambda' + \log(2nd(m+2d)))\right] \leqslant m + O(\lambda \log n(\log n + \log m))$.

■

Acknowledgment

Pasin would like to thank Theo McKenzie for useful discussions on the independent set problem on semi-random graphs.

References

[ACF$^+$04] Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christopher T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA, January 10, 2004*, pages 62–69, 2004.

[AFLS07] Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters. Crown structures for vertex cover kernelization. *Theory Comput. Syst.*, 41(3):411–430, 2007.

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

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

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

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

[BP16] Allison Bishop and Valerio Pastro. Robust secret sharing schemes against local adversaries. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 327–356. Springer, Heidelberg, March 2016.

[BPRW16] Allison Bishop, Valerio Pastro, Rajmohan Rajaraman, and Daniel Wichs. Essentially optimal robust secret sharing with maximal corruptions. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 58–86. Springer, Heidelberg, May 2016.

[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.

[CDD$^+$15] Ronald Cramer, Ivan Bjerre Damgard, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 313–336, 2015.

[CDF01] Ronald Cramer, Ivan Damgard, and Serge Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 503–523. Springer, Heidelberg, August 2001.

[CDF$^+$08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488. Springer, Heidelberg, April 2008.

[CDV94] Marco Carpentieri, Alfredo De Santis, and Ugo Vaccaro. Size of shares and probability of cheating in threshold schemes. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 118–125. Springer, Heidelberg, May 1994.

[Cev11] Alfonso Cevallos. Reducing the share size in robust secret sharing. Mathematisch Instituut, Universiteit Leiden, 2011.

[CFJ04] Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Graph-Theoretic Concepts in Computer Science, 30th International Workshop,WG 2004, Bad Honnef, Germany, June 21-23, 2004, Revised Papers*, pages 257–269, 2004.

[CFOR12] Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 195–208. Springer, Heidelberg, April 2012.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395. IEEE Computer Society Press, October 1985.

[Che15] Mahdi Cheraghchi. Nearly optimal robust secret sharing. Cryptology ePrint Archive, Report 2015/951, 2015. `http://eprint.iacr.org/2015/951`.

[CSV17] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 47–60, 2017.

[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.

[FK00] Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, 16(2):195–208, 2000.

[FK01] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *J. Comput. Syst. Sci.*, 63(4):639–671, 2001.

[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.

[FY19] Serge Fehr and Chen Yuan. Towards optimal robust secret sharing with security against a rushing adversary. In Vincent Rijmen and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, Lecture Notes in Computer Science, pages 472–499. Springer, Heidelberg, May 2019.

[GK18] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 685–698. ACM Press, June 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.

[HO18] Brett Hemenway and Rafail Ostrovsky. Efficient robust secret sharing from expander graphs. *Cryptography and Communications*, 10(1):79–99, 2018.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972.

[MMT18] Theo McKenzie, Hermish Mehta, and Luca Trevisan. A new algorithm for the robust semi-random independent set problem. *CoRR*, abs/1808.03633, 2018.

[NT74] George L. Nemhauser and Leslie E. Trotter. Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6(1):48–61, 1974.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st Annual ACM Symposium on Theory of Computing*, pages 73–85. ACM Press, May 1989.

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