Essentially Optimal Robust Secret Sharing with Maximal Corruptions

Allison Bishop[1], Valerio Pastro[1],
Rajmohan Rajaraman[2], and Daniel Wichs[2]

[1] Columbia University
[2] Northeastern University

Abstract. In a t-out-of-n *robust secret sharing scheme*, a secret message is shared among n parties who can reconstruct the message by combining their shares. An adversary can adaptively corrupt up to t of the parties, get their shares, and modify them arbitrarily. The scheme should satisfy *privacy*, meaning that the adversary cannot learn anything about the shared message, and *robustness*, meaning that the adversary cannot cause the reconstruction procedure to output an incorrect message. Such schemes are only possible in the case of an honest majority, and here we focus on unconditional security in the maximal corruption setting where $n = 2t + 1$.

In this scenario, to share an m-bit message with a reconstruction failure probability of at most 2^{-k}, a known lower-bound shows that the share size must be at least $m+k$ bits. On the other hand, all prior constructions have share size that scales linearly with the number of parties n, and the prior state-of-the-art scheme due to Cevallos et al. (EUROCRYPT '12) achieves $m + \widetilde{O}(k + n)$.

In this work, we construct the first robust secret sharing scheme in the maximal corruption setting with $n = 2t + 1$, that avoids the linear dependence between share size and the number of parties n. In particular, we get a share size of only $m + \widetilde{O}(k)$ bits. Our scheme is computationally efficient and relies on approximation algorithms for the minimum graph bisection problem.

1 Introduction

Secret sharing, originally introduced by Shamir [Sha79] and Blakely [Bla79], is a central cryptographic primitive at the heart of a wide variety of applications, including secure multiparty computation, secure storage, secure message transmission, and threshold cryptography. The functionality of secret sharing allows a dealer to split a secret message into shares that are then distributed to n parties. Any authorized subset of parties can reconstruct the secret reliably from their shares, while unauthorized subsets of parties cannot learn anything about the secret from their joint shares. In particular, a t-out-of-n *threshold* secret sharing scheme requires that any t shares reveal no information about the secret, while any subset of $t + 1$ shares can be used to reconstruct the secret.

Many works (e.g. [RB89,CSV93,BS97,CDF01,CFOR12,LP14,CDD$^+$15,Che15]) consider a stronger notion of secret sharing called *robust secret sharing* (RSS). Robustness requires that, even if an adversary can replace shares of corrupted parties by maliciously chosen values, the parties can still reconstruct the true secret. (with high probability). secure storage and message transmission. In particular, we consider a computationally unbounded adversary who maliciously (and adaptively) corrupts t out of n of the parties and learns their shares. After corrupting the t parties, the adversary can adaptively modify their shares and replace them with arbitrary values. The reconstruction algorithm is given the shares of all n parties and we require that it recovers the original secret. Note that robustness requires the reconstruction to work given all n shares of which t contain "errors" while threshold reconstruction is given $t + 1$ correct shares, meaning that $n - t - 1$ shares are "erasures". When $n = 2t + 1$, robustness is therefore a strictly stronger requirement than threshold reconstruction (but in other settings this is not the case).

Known Lower Bounds. It is known that robust secret sharing can only be achieved with an honest majority, meaning $t < n/2$. Moreover, for t in the range $n/3 \leq t < n/2$, we cannot achieve perfect robustness, meaning that we must allow at least a small (negligible) failure probability for reconstruction [Cev11]. Furthermore, in the maximal corruption setting with $n = 2t + 1$ parties, any robust secret sharing scheme for m-bit messages with failure probability 2^{-k} must have a share size that exceeds $m + k$ bits [CSV93,LP14].

Prior Constructions. On the positive side, several prior works show how to construct robust sharing schemes in the maximal corruption setting with $n = 2t + 1$ parties. The first such scheme was described in the work of Rabin and Ben-Or [RB89] with a share size of $m + \widetilde{O}(nk)$ bits. Cramer, Damgård and Fehr [CDF01] showed how to improve this to $m + \widetilde{O}(k + n)$ bits, using what later became known as algebraic-manipulation detection (AMD) codes [CDF$^+$08], but at the cost of having an inefficient reconstruction procedure. Cevallos et al. [CFOR12] then presented an efficient scheme with share size $m + \widetilde{O}(k + n)$.

Other Related Work. Two recent works [CDD$^+$15,Che15] study robust secret sharing in the setting where the number of corruptions is below the maximal threshold by some constant fraction; i.e., $t = (1/2 - \delta)n$ for some constant $\delta > 0$. In this setting, robustness does not necessarily imply threshold reconstructability from $t + 1$ correct shares (but only from $(1/2 + \delta)n$ correct shares). This is often called a *ramp* setting, where there is a gap between the privacy threshold t and the reconstructability threshold. The above works show that it is possible to achieve robustness in this setting with share size of only $O(1)$ bits, when n is sufficiently large in relation to k, m. The work of [Che15] also considers a setting that separately requires robustness *and* threshold reconstruction from $t + 1$ correct shares, and gives a scheme with share size $m + \widetilde{O}(k)$. However, we notice that security in this setting could always be achieved with a *generic* *construction*, by adding a standard (non-robust) threshold secret sharing scheme on top of a robust scheme – when given exactly $t+1$ shares, the reconstruction uses the threshold scheme, and otherwise it uses the robust scheme. This adds m additional bits to the share size of the robust scheme and, in particular, when n is sufficiently large, the share size would be $m + O(1)$ bits.[3]

The main technique used to get robustness in [CDD+15,Che15] is to compose list-decodable codes with a special privacy property together with AMD codes from [CDF+08]. Unfortunately, this technique appears to crucially fail in the setting of $n = 2t+1$ parties. In this setting, the parameters of the list-decodable codes either force a large alphabet or an exponential list size. The latter in turn forces us to use an AMD code with large overhead. In either case the overhead on the share size appears to be at least $O(n)$ bits.

Another related work of [LP14] considers a relaxation of robustness to a setting of *local adversaries*, where each corrupted party's modified share can only depend on its own received share, but not on the shares received by the other corrupted parties. They construct a robust scheme in this setting for $n = 2t+1$ parties with share size of $m+\widetilde{O}(k)$ bits. Unfortunately, the construction is clearly insecure in the traditional robustness setting where a monolithic adversary gets to see the shares received by *all* of the corrupted parties before deciding how to modify each such share.

Finally, the work of [JS13] proposes a robust secret sharing scheme with t corruptions out of $n \geq 2t+2$. Unfortunately, we found a flaw in the security proof and an attack showing that the proposed scheme is insecure (which we communicated to the authors).

In summary, despite the intense study of robust secret sharing since the late 80s and early 90s, in the maximal corruption setting with $n = 2t+1$ parties there is a large gap between the lower bound of $m + k$ bits and the best previously known upper bound of $m+\widetilde{O}(n+k)$ bits on the share size. In particular, prior to this work, it was not known if the linear dependence between the share size and n is necessary in this setting, or whether there exist (even inefficient) schemes that beat this bound.

Our Result. We present an efficient robust secret sharing scheme in the maximal corruption setting with $n = 2t + 1$ parties, where the share size of only $m + \widetilde{O}(k)$ bits (see Section 6 for detailed parameters including poly-logarithmic factors). This is the first such scheme which removes the linear dependence between the share size and the number of parties n. A comparison between our work and previous results is given in table 1.

[3] Yet another intermediate variant is to separately require robustness with $t = (1/2 - \delta)n$ corruptions and ramp reconstruction from $t + \rho n = (1/2 - \delta + \rho)n$ correct shares for some constants $\delta, \rho > 0$. This could always be achieved by adding a good (non-robust) ramp secret sharing scheme on top of a robust scheme while maintaining the $O(1)$ share size when n is sufficiently large.

Setting: $t = (1/2 - \Omega(1))n$			
Reconstruction	**Construction**	**Share Size**	**Lower bound**
Robust Only	[CDD+15,Che15]	$O(1)$ †	
Robust + Threshold	[Che15] Generic construction	$(1 + o(1))m + O(k)$ $m + O(1)$ †	m

Setting: $n = 2t + 1$			
Reconstruction	**Construction**	**Share Size**	**Lower bound**
Robust ⇒ Threshold	[RB89] [CDF01,CDF+08,CFOR12] Our work	$m + \widetilde{O}(kn)$ $m + \widetilde{O}(k + n)$ $m + \widetilde{O}(k)$	$m + k$

Table 1. Comparison of robust secret sharing schemes and lower bounds with n parties, t corruptions, m-bit message, and 2^{-k} reconstruction error. *Robust* reconstruction is given n shares with t errors, while *threshold* reconstruction is given just $t + 1$ correct shares ($n - t - 1$ erasures). The $\widetilde{O}(\cdot)$ notation hides factors that are poly-logarithmic in k, n and m. This is justified if we think of n, m as some arbitrarily large polynomials in the security parameter k. The † requires that n is sufficiently large in relation to m, k.

1.1 Our Techniques

Using MACs. We begin with the same high-level idea as the schemes of [RB89,CFOR12], which use information-theoretic message authentication codes (MACs) to help the reconstruction procedure identify illegitimate shares. The basic premise is to start with a standard (non-robust) t-out-of-n scheme, such as Shamir's scheme, and have parties authenticate each others' Shamir shares using MACs. Intuitively, this should make it more difficult for an adversary to present compelling false shares for corrupted parties as it would have to forge the MACs under unknown keys held by the honest parties.

The original implementation of this idea by Rabin and Ben-Or [RB89] required each party to authenticate the share of every other party with a MAC having unforgeability security 2^{-k} and the reconstruction procedure simply checked that the majority of the tags verified. Therefore, the keys and tags added an extra $\widetilde{O}(nk)$ overhead to the share of each party. The work of Cevallos et al. [CFOR12] showed that one can also make this idea work using a MAC with a weaker unforgeability security of only $\frac{1}{\Omega(n)}$, by relying on a more complex reconstruction procedure. This reduced the overhead to $\widetilde{O}(k + n)$ bits.

Random Authentication Graph. Our core insight is to have each party only authenticate a relatively small but randomly chosen subset of other parties' shares. This will result in a much smaller overhead in the share size, essentially independent of the number of parties n.

More precisely, for each party we choose a random subset of $d = \widetilde{O}(k)$ other parties whose shares to authenticate. We can think of this as a random "authentication graph" $G = ([n], E)$ with out-degree d, having directed edges $(i, j) \in E$

if party i authenticates party j. This graph is stored in a distributed manner where each party i is responsible for storing the information about its d outgoing edges. It is important that this graph is not known to the attacker when choosing which parties to corrupt. In fact, as the attacker adaptively corrupts parties, he should not learn anything about the outgoing edges of uncorrupted parties.[4]

Requirements and Inefficient Reconstruction. As a first step, let's start by considering an inefficient reconstruction procedure, as this will already highlight several challenges. The reconstruction procedure does not get to see the original graph G but a possibly modified graph $G' = ([n], E')$ where the corrupted parties can modify their set of outgoing edges. However, the edges that originate from uncorrupted parties are the same in G and G'. The reconstruction procedure labels each edge $e \in E'$ as either good or bad depending on whether the MAC corresponding to that edge verifies.

Let's denote the subset of uncorrupted *honest* parties by $H \subseteq [n]$. Let's also distinguish between corrupted parties where the adversary does not modify the share, which we call *passive corruptions* and denote by $P \subseteq [n]$, and the rest which we call *active corruptions* and denote by $A \subseteq [n]$. Assume that we can ensure that the following requirements are met:

(I) All edges between honest/passive parties, $(i, j) \in E' \ : \ i, j \in H \cup P$, are labeled good.

(II) All edges from honest to active parties, $(i, j) \in E' \ : \ i \in H, j \in A$ are labeled bad.

In this case, the reconstruction procedure can (inefficiently) identify the set $H \cup P$ by simply finding the *maximum self-consistent set* of vertices $C \subseteq [n]$, i.e. the largest subset of vertices such that all of the tags corresponding to edges $(i, j) \in E'$ with $i, j \in C$ are labeled good. We show that $C = H \cup P$ is the unique maximum self-consistent set with overwhelming probability (see Section 7). Once we identify the set $H \cup P$ we can simply reconstruct the secret message from the Shamir shares of the parties in $H \cup P$ since these have not been modified.

Implementation: Private MAC and Robust Storage of Tags. Let's now see how to implement the authentication process to satisfy requirements I and II defined above. A naive implementation of this idea would be for each party i to have a MAC key key_i for a d-time MAC (i.e., given the authentication tags of d messages one cannot forge the tag of a new message) and, for each edge $(i, j) \in E$, to create a tag $\sigma_{i \to j} = \mathsf{MAC}_{\mathsf{key}_i}(\widetilde{s}_j)$ where $\widetilde{s}_j$ is the Shamir share of

[4] If the graph were chosen at random but known to the attacker in advance, then the attacker could always choose some honest party i and corrupt a set of t parties none of which are being authenticated by i. Then the $t + 1$ shares corresponding to the t corrupted parties along with honest party i would be consistent and the reconstruction would not be able to distinguish it from the set of $t+1$ honest parties. However, with an unknown graph, there is a high probability that every honest party i authenticates many corrupted parties.

party j. The tags $\sigma_{i \to j}$ would then be stored with party j. In particular, the full share of party i would be

$$s_i = (\widetilde{s}_i, E_i, \mathsf{key}_i, \{\sigma_{j \to i}\}_{(j,i) \in E})$$

where $E_i = \{j \in [n] \ : \ (i,j) \in E\}$ are the outgoing edges for party i.

Unfortunately, there are several problems with this. Firstly, if the adversary corrupts party i, it might modify the values key_i, E_i in the share of party i but keep the Shamir share $\widetilde{s}_i$ intact. This will keep the edges going from honest parties to party i labeled good but some of the edges going from party i to honest parties might now be labeled bad. Therefore we cannot define such party as either passive (this would violate requirement I) or active (this would violate requirement II). Indeed, this would break our reconstruction procedure.

To fix this, when party i authenticates another party j, we compute $\sigma_{i \to j} = \mathsf{MAC}_{\mathsf{key}_i}((\widetilde{s}_j, E_j, \mathsf{key}_j))$ where we authenticate the values E_j, key_j along with the Shamir share $\widetilde{s}_j$. This prevents party j from being able to modify these components without being detected. Therefore we can define a party as active if any of the components $\widetilde{s}_j, E_j, \mathsf{key}_j$ are modified and passive otherwise.

Unfortunately, there is still a problem. An adversary corrupting party j might keep the components $\widetilde{s}_j, E_j, \mathsf{key}_j$ intact but modify some subset of the tags $\sigma_{i \to j}$. This will make some of edges going from honest parties to party j become bad while others remain good, which violates the requirements.

To fix this, we don't store tags $\sigma_{i \to j}$ with party j but rather we store all the tags in a distributed manner among the n parties in a way that guarantees recoverability even if t parties are corrupted. However, we do not provide any privacy guarantees for these tags and the adversary may be able to learn all of them in full. We call this *robust distributed storage* (without privacy), and show that we can use it to store the tags without additional asymptotic overhead. The fact that the tags are not stored privately requires us to use a special type of *private (randomized) MAC* where the tags $\sigma_{i \to j}$ do not reveal anything about the authenticated messages even given the secret key key_i. With this implementation, we can guarantee that requirements I, II are satisfied.

Efficient Reconstruction using Graph Bisection. To get an efficient reconstruction procedure, we need to solve the following *graph identification problem.* An adversary partitions vertices $V = [n]$ into three sets H, P, A corresponding to honest, active and passive parties respectively. We know that the out-going edges from H are chosen randomly and that the edges are labeled as either good or bad subject to requirements I, II above. The goal is to identify $H \cup P$. We know that, with overwhelming probability, $H \cup P$ is the unique maximum self-consistent set having no bad edges between its vertices, but its not clear how to identify it efficiently.

Let's consider two cases of the above problem depending on whether the size of the passive set P is $|P| \geq \varepsilon n$ or $|P| < \varepsilon n$ for some $\varepsilon = 1/\Theta(\log n)$. If P is larger than εn, then we can distinguish between vertices in A and $H \cup P$ by essentially counting the number of incoming bad edges of each vertex. In particular, the vertices in $H \cup P$ only have incoming bad edges from the set A of size $|A| = (1/2-\varepsilon)n$ while the vertices in A which have incoming bad edged from H of size $|H| = n/2$. Therefore, at least on average, the vertices in A will have more incoming bad edges than those in $H \cup P$. This turns out to be sufficient to then identify all of $H \cup P$.

On the other hand, if $|P| < \varepsilon n$ then the graph has a "bisection" consisting of H and $A \cup P$ (whose sizes only differ by 1 vertex) with only approximately εnd good edges crossing from H to $A \cup P$, corresponding to the edges from H to P. We then rely on the existence of efficient approximation algorithms for the *minimum graph bisection problem*. This is a classic NP-hard optimization problem [GJS76,FK02], and the best known polynomial-time algorithm is an $O(\log n)$-approximation algorithm due to [Räc08]. In particular, this allows us to bisect the graph it into two components X_0, X_1 with only very few edges crossing from X_0 to X_1. This must mean that one of X_0 or X_1 contains the vast majority of the vertices in H as otherwise, if the H vertices were split more evenly, there would be many more edges crossing. Having such components X_0, X_1 turns out to be sufficient to then identify all of $H \cup P$.

There are many details to fill in for the above high-level description, but one major issue is that we only have efficient approximations for the graph bisection problem in *undirected* graphs. However, in the above scenario, we are only guaranteed that there are few good edges from H to $A \cup P$ but there may be many good edges in the reverse direction. To solve this problem, we need to make sure that our graph problem satisfies one additional requirement (in addition to requirements I, II above):

(III) All edges from active to honest parties, $(i, j) \in E' \,:\, i \in A, j \in H$ are labeled bad.

To ensure that this holds, we need to modify the scheme so that, for any edge $(i, j) \in E$ corresponding to party i using its key to authenticate the share of party j with a tag $\sigma_{i \to j}$, we also add a "reverse-authentication" tag $\sigma_{i \leftarrow j}$ where we authenticate the share of party i under the key of party j. This ensures that edges from active parties to honest parties are labeled bad. Therefore, when P is small, there are very few good edges between H and $A \cup P$ in either direction and we can use an algorithm for the undirected version of the graph bisection problem.

Parallel Repetition and Parameters. A naive instantiation of the above scheme would require a share size of $m+\widetilde{O}(k^2)$ since we need $O(k)$ tags per party and each tag needs to have length $O(k)$. To reduce the share size further, we first instantiate our scheme with much smaller parameters which only provide weak security and ensure that the correct message is recovered with probability $3/4$. We then use $O(k)$ parallel copies of this scheme to amplify security, where the reconstruction outputs the majority value. One subtlety is that all of the copies need to use the same underlying Shamir shares since we don't want a multiplicative blowup in the message size m. We show that this does not hurt security. Altogether, this results in a share size of only $m + \widetilde{O}(k)$.

2 Notation and Preliminaries

For $n \in \mathbb{N}$, we let $[n] := \{1, \ldots, n\}$. If X is a distribution or a random variable, we let $x \leftarrow X$ denote the process of sampling a value x according to the distribution X. If A is a set, we let $a \leftarrow A$ denote the process of sampling a uniformly at random from A. If f is a randomized algorithm, we let $f(x; r)$ denote the output of f on input x with randomness r. We let $f(x)$ be a random variable for $f(x; r)$ with random r.

Sub-Vector Notation. For a vector $\boldsymbol{s} = (s_1, \ldots, s_n)$ and a set $I \subseteq [n]$, we let $\boldsymbol{s}_I$ denote the vector consisting only of values in indices $i \in I$; we will represent this as $\boldsymbol{s}_I = (s'_1, \ldots, s'_n)$ with $s'_i = s_i$ for $i \in I$ and $s'_i = \perp$ for $i \notin I$.

Graph Notation. For a (directed) graph $G = (V, E)$, and sets $X, Y \subseteq V$, define $E_{X \to Y}$ as the set of edges from X to Y; i.e. $E_{X \to Y} = \{(v_1, v_2) \in E \mid v_1 \in X, v_2 \in Y\}$.

2.1 Hash Functions, Polynomial Evaluation

Definition 1 (Universal Hashing). *Let $\mathcal{H} = \{H_k : \mathcal{U} \to \mathcal{V}\}_{k \in \mathcal{K}}$ be family of hash functions. We say that $\mathcal{H}$ is ε-universal if for all $x, x' \in \mathcal{U}$ with $x \neq x'$ we have $\Pr_{k \leftarrow \mathcal{K}}[H_k(x) = H_k(x')] \leq \varepsilon$.*

Polynomial Evaluation. Let $\mathbb{F}$ be a finite field. Define the *polynomial evaluation* function $\mathsf{PEval} : \mathbb{F}^d \times \mathbb{F} \to \mathbb{F}$ as $\mathsf{PEval}(\boldsymbol{a}, x) = \sum_{i=1}^{d} a_i x^i$. See the full version [BPRW15] for the properties of the polynomial evaluation we rely on.

2.2 Graph Bisection

Let $G = (V, E)$ be an undirected graph. Let (V_1, V_2) be a partition of its edges. The *cross edges* of (V_1, V_2) are the edges in $E_{V_1 \to V_2}$. Given an undirected graph $G = (V, E)$ with an even number of vertices $|V| = n = 2t$ a *graph bisection* for G is a partition (V_1, V_2) of V such that $|V_1| = t = |V_2|$. We also extend the notion of a graph bisection to graphs with an odd number of vertices $|V| = n = 2t + 1$ by defining a bisection to be a partition with $|V_1| = t, |V_2| = t + 1$.

Definition 2 (Approximate Graph Bisection Algorithm). *Let $G = (V, E)$ be an undirected graph with n vertices. Assume that G has a graph bisection V_1, V_2 with $|E_{V_1 \to V_2}| = m$ cross edges. An algorithm* Bisect *that takes as input G and outputs a bisection U_1, U_2 with at most $|E_{U_1 \to U_2}| \leq \delta m$ cross edges is called a* δ-approximate graph bisection algorithm.

We remark that standard definitions of graphs bisection only consider the case where $n = 2t$ is even. However, given any δ-approximate graph bisection algorithm that works in the even case, we can generically adapt it to also work in the odd case $n = 2t+1$. In particular, given a graph $G = (V, E)$ with $|V| = 2t+1$

vertices, we can construct a graph $G' = (V \cup \{\perp\}, E)$ with an added dummy vertex $\perp$ that has no outgoing or incoming edges. We then run the δ-approximate graph bisection algorithm that works for an even number of vertices on G' to get a bisection U'_1, U'_2 where, without loss of generality, we assume that U'_1 contains the vertex $\perp$. By simply taking U_1 to be U'_1 with $\perp$ removed and $U_2 = U'_2$ we get a δ-approximate bisection for the graph G.

The work of [FK02] gave an efficient $O(\log^{1.5} n)$-approximate graph bisection algorithm, which was then improved to $O(\log n)$ by [Räc08] (Section 3, "Min Bisection").

3 Definition of Robust Secret Sharing

Throughout the rest of the paper, we use the following notation:

- t denotes the number of players that are arbitrarily corrupt.
- $n = 2t + 1$ denotes the number of players in the scheme.
- $\mathcal{M}$ is the message space.

Definition 3 (Robust Secret Sharing). *A* t-out-of-n, δ-robust secret sharing scheme *over a message space* $\mathcal{M}$ *and share space* $\mathcal{S}$ *is a tuple* $(\mathsf{Share}, \mathsf{Rec})$ *of algorithms that run as follows:*

$\mathsf{Share}(\mathsf{msg}) \to (s_1, \ldots, s_n)$**:** *This is a randomized algorithm that takes as input a message* $\mathsf{msg} \in \mathcal{M}$ *and outputs a sequence of shares* $s_1, \ldots, s_n \in \mathcal{S}$*.*

$\mathsf{Rec}(s_1, \ldots, s_n) \to \mathsf{msg}'$**:** *This is a deterministic algorithm that takes as input* n *shares* $(s_1, \ldots, s_n)$ *with* $s_i \in \mathcal{S} \cup \perp$ *and outputs a message* $\mathsf{msg}' \in \mathcal{M}$*.*

We require perfect correctness, meaning that for all $\mathsf{msg} \in \mathcal{M}$*:* $\Pr[\mathsf{Rec}(\mathsf{Share}(\mathsf{msg})) = \mathsf{msg}] = 1$*. Moreover, the following properties hold:*

Perfect Privacy: *Any* t *out of* n *shares of a secret give no information on the secret itself. More formally, for any* $\mathsf{msg}, \mathsf{msg}' \in \mathcal{M}$*, any* $I \subseteq [n]$ *of size* $|I| = t$*, the distributions* $\mathsf{Share}(\mathsf{msg})_I$ *and* $\mathsf{Share}(\mathsf{msg}')_I$ *are identical.*

Perfect Threshold Reconstruction (with erasures): *The original secret can be reconstructed from any* $t+1$ *correct shares. More formally, for any* $\mathsf{msg} \in \mathcal{M}$ *and any* $I \subseteq [n]$ *with* $|I| = t+1$ *we have* $\Pr[\mathsf{Rec}(\mathsf{Share}(\mathsf{msg})_I) = \mathsf{msg}] = 1$*.*

Adaptive δ-Robustness: *An adversary that adaptively modifies up to* t *shares can cause the wrong secret to be recovered with probability at most* δ*. More formally, we define the experiment* ***Exp***$(\mathsf{msg}, \mathsf{Adv})$ *with some secret* $\mathsf{msg} \in \mathcal{M}$ *and interactive adversary* Adv*.*

Exp$(\mathsf{msg}, \mathsf{Adv})$**:** *is defined as follows:*

E.1. *Sample* $\boldsymbol{s} = (s_1, \ldots, s_n) \leftarrow \mathsf{Share}(\mathsf{msg})$*.*

E.2. *Set* $I := \varnothing$*. Repeat the following while* $|I| \leq t$*.*

- Adv *chooses* $i \in [n] \setminus I$*.*
- *Update* $I := I \cup \{i\}$ *and give* s_i *to* Adv*.*

E.3. Adv *outputs modified shares* $s'_i \ : \ i \in I$ *and we define* $s'_i := s_i$ *for* $i \notin I$*.*

E.4. Compute $\mathsf{msg}' = \mathsf{Rec}(s'_1, \ldots, s'_n)$.

E.5. If $\mathsf{msg}' \neq \mathsf{msg}$ *output 1 else 0.*

We require that for any (unbounded) adversary Adv *and any* $\mathsf{msg} \in \mathcal{M}$ *we have*

$$\Pr[\boldsymbol{Exp}(\mathsf{msg}, \mathsf{Adv}) = 1] \leq \delta.$$

Remarks. We note that since privacy and threshold reconstruction are required to hold perfectly (rather than statistically) there is no difference between defining non-adaptive and adaptive variants. In other words, we could also define adaptive privacy where the adversary gets to choose which shares to see adaptively, but this is already implied by our non-adaptive definition of perfect privacy. We also note that when $n = 2t + 1$ then robustness implies a *statistically* secure threshold reconstruction with erasures. However, since we can even achieve *perfect* threshold reconstruction, we define it as a separate property.

Definition 4 (Non-Robust Secret Sharing). *We will say that a scheme is a* non-robust *t-out-of-n secret sharing scheme, if it satisfies the above definition with $\delta = 1$.*

Using Shamir secret sharing, we get a non-robust t-out-of-n secret sharing for any $t < n$ where the share size is the same as the message size.

4 The Building Blocks

In this section we introduce the building blocks of our robust secret sharing scheme: *Robust Distributed Storage*, *Private MACs*, and the *Graph Identification* problem.

4.1 Robust Distributed Storage

A robust distributed storage scheme allows us to store a public value among n parties, t of which may be corrupted. There is no secrecy requirement on the shared value. However, we require robustness: if the adversary adaptively corrupts t of the parties and modifies their shares, the reconstruction procedure should recover the correct value with overwhelming probability. We can think of this primitive as a relaxation of an error-correcting code where shares correspond to codeword symbols. The main difference is that the encoding procedure can be randomized and the adversary only gets to see a set of t (adaptively) corrupted positions of the codeword before deciding on the errors in those positions. These restrictions allow us to achieve better parameters than what is possible with standard error-correcting codes.

Definition 5. *A* t-out-of-n, δ-robust distributed storage *over a message space $\mathcal{M}$ is a tuple of algorithms* $(\mathsf{Share}, \mathsf{Rec})$ *having the same syntax as robust secret sharing, and satisfying the δ-robustness property. However, it need not satisfy the privacy or perfect reconstruction with erasures properties.*

We would like to construct such schemes for $n = 2t + 1$ and for a message of size m so that the share of each party is only $O(m/n)$ bits. These parameters are already beyond the reach of error-correcting codes for worst-case errors. We construct a simple robust distributed storage scheme by combining list-decoding and universal hashing.

List Decoding. In list-decoding, the requirement to decode to a unique codeword is relaxed, and it is only required to obtain a polynomially sized list of potential candidates that is guaranteed to include the correct codeword. We can simply use Reed-Solomon codes and the list-decoding algorithm provided by Sudan [Sud97] (better parameters are known but this suffices for our needs):

Theorem 1 ([Sud97]). *A Reed-Solomon code formed by evaluating a degree d polynomial on n points can be efficiently list-decoded to recover from $e < n - \sqrt{2dn}$ errors with a list of size $L \leq \sqrt{2n/d}$.*

Setting $d = \lfloor n/8 \rfloor$, we can then therefore recover from t out of $n = 2t + 1$ errors and obtain a list of size $L \leq \sqrt{2n/d} = O(1)$.

Construction of Robust Distributed Storage. Let t be some parameter, let $n = 2t + 1$, and let $\mathbb{F}$ be a field of size $|\mathbb{F}| = 2^u$ with $|\mathbb{F}| > n$. Let $\mathcal{H} = \{H_k : \mathbb{F}^{d+1} \to \mathbb{F}\}_{k \in \mathbb{F}}$ be an ε-universal hash function. For concreteness, we can use the polynomial evaluation hash $H_k(\boldsymbol{a}) = \mathsf{PEval}(\boldsymbol{a}, k)$, which achieves $\varepsilon = (d + 1)/2^u$ (see 'XOR-universality' in the full version [BPRW15]). We use list-decoding for the Reed Solomon code with degree $d = \lfloor n/8 \rfloor = \Omega(n)$ which allows us to recover from t out of n errors with a list size $L = O(1)$. We construct a δ-robust t-out-of-n distributed storage scheme with message space $\mathcal{M} = \mathbb{F}^{d+1}$, meaning that the messages have bit-size $m = u(d + 1) = \Omega(un)$, share size $3u = O(u)$, and robustness $\delta = nL\varepsilon = O(n^2)/2^u$. The scheme works as follows:

- $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(\mathsf{msg})$. Encode $\mathsf{msg} \in \mathbb{F}^{d+1}$ using the Reed-Solomon code by interpreting it as a degree d polynomial and evaluating it on n points to get the Reed-Solomon codeword $(\hat{s}_1, \ldots, \hat{s}_n) \in \mathbb{F}^n$. Choose random values $k_1, \ldots, k_n \leftarrow \mathbb{F}$ and define the shares $s_i = (\hat{s}_i, k_i, H_{k_i}(\mathsf{msg})) \in \mathbb{F}^3$.
- $\mathsf{msg}' \leftarrow \mathsf{Rec}(s'_1, \ldots, s'_n)$. Parse $s'_i = (\hat{s}'_i, k'_i, y'_i)$. Use list-decoding on the modified codeword $(\hat{s}'_1, \ldots, \hat{s}'_n) \in \mathbb{F}^n$ to recover a list of $L = O(1)$ possible candidates $\mathsf{msg}^{(1)}, \ldots, \mathsf{msg}^{(L)} \in \mathbb{F}^{d+1}$ for the message. Output the first value $\mathsf{msg}^{(j)}$ that agrees with the majority of the hashes:
$$|\{i \in [n] \ : \ H_{k'_i}(\mathsf{msg}^{(j)}) = y'_i\}| \geq t + 1.$$

Theorem 2. *For any $n = 2t+1$ and any $u \geq \log n$, the above scheme is a t-out-of-n, δ-robust distributed storage scheme for messages of length $m = \lfloor n/8 \rfloor u = \Omega(nu)$ with shares of length $3u = O(u)$ and robustness $\delta = O(n^2)/2^u$.*

The proof is given in the full version [BPRW15].

4.2 Private Labeled MAC

As a tool in our construction of robust secret sharing schemes, we will use a new notion of an information-theoretic message-authentication code (MAC) that has additional privacy guarantees.

The message authentication code $\sigma = \mathsf{MAC}_{\mathsf{key}}(\mathsf{lab}, \mathsf{msg}, r)$ takes as input a *label* lab, a *message* msg, and some additional *randomness* r. The randomness is there to ensure privacy for the message msg even given key, σ.

Definition 6 (Private Labeled MAC). *An* (ℓ, ε) private MAC *is a family of functions* $\{\mathsf{MAC}_{\mathsf{key}} : \mathcal{L} \times \mathcal{M} \times \mathcal{R} \to \mathcal{T}\}_{\mathsf{key} \in \mathcal{K}}$ *with key-space* $\mathcal{K}$*, message space* $\mathcal{M}$*, label space* $\mathcal{L}$*, randomness space* $\mathcal{R}$*, and tag space* $\mathcal{T}$*. It has the following properties:*

Authentication: *For any* ℓ *values* $(\mathsf{lab}_i, \mathsf{msg}_i, r_i, \sigma_i) \in \mathcal{L} \times \mathcal{M} \times \mathcal{R} \times \mathcal{T} \; : \; i = 1, \ldots, \ell$ *such that the labels* lab_i *are distinct, and for any* $(\mathsf{lab}', \mathsf{msg}', r', \sigma') \in \mathcal{L} \times \mathcal{M} \times \mathcal{R} \times \mathcal{T}$ *such that* $(\mathsf{lab}', \mathsf{msg}', r') \notin \{(\mathsf{lab}_i, \mathsf{msg}_i, r_i)\}_{i \in [\ell]}$ *we have:*

$$\Pr_{\mathsf{key} \leftarrow \mathcal{K}}[\mathsf{MAC}_{\mathsf{key}}(\mathsf{lab}', \mathsf{msg}', r') = \sigma' \mid \{\mathsf{MAC}_{\mathsf{key}}(\mathsf{lab}_i, \mathsf{msg}_i, r_i) = \sigma_i\}_{i \in [\ell]}] \leq \varepsilon.$$

This implies that even after seeing the authentication tags σ_i for ℓ tuples $(\mathsf{lab}_i, \mathsf{msg}_i, r_i)$ with distinct labels lab_i, an adversary cannot come up with a valid tag σ' for any new tuple $(\mathsf{lab}', \mathsf{msg}', r')$.

Privacy Over Randomness: *For any* ℓ *distinct labels* $\mathsf{lab}_1, \ldots, \mathsf{lab}_\ell$*, any keys* $\mathsf{key}_1, \ldots, \mathsf{key}_\ell \in \mathcal{K}$*, and any* $\mathsf{msg} \in \mathcal{M}$*, the* ℓ *values* $\sigma_1 = \mathsf{MAC}_{\mathsf{key}_1}(\mathsf{lab}_1, \mathsf{msg}, r)$*,* $\ldots, \sigma_\ell = \mathsf{MAC}_{\mathsf{key}_\ell}(\mathsf{lab}_\ell, \mathsf{msg}, r)$ *are uniformly random and independent in* $\mathcal{T}$ *over the choice of* $r \leftarrow \mathcal{R}$*.*

This says that the tags σ_i do not reveal any information about the message msg, or even about the labels lab_i and the keys key_i, as long as the randomness r is unknown.

Privacy Over Keys: *Let* $(\mathsf{lab}_i, \mathsf{msg}_i, r_i) \in \mathcal{L} \times \mathcal{M} \times \mathcal{R} \; : \; i = 1, \ldots, \ell$ *be* ℓ *values such that the labels* lab_i *are distinct. Then the* ℓ *values* $\sigma_1 = \mathsf{MAC}_{\mathsf{key}}(\mathsf{lab}_1, \mathsf{msg}_1, r_1)$*,* $\ldots, \sigma_\ell = \mathsf{MAC}_{\mathsf{key}}(\mathsf{lab}_\ell, \mathsf{msg}_\ell, r_\ell)$ *are uniformly random and independent in* $\mathcal{T}$ *over a random* $\mathsf{key} \leftarrow \mathcal{K}$*.*

This says that the tags σ_i do not reveal any information about the values $(\mathsf{lab}_i, \mathsf{msg}_i, r_i)$ as long as key is unknown.

Construction. Let $\mathbb{F}$ and $\mathbb{F}'$ be finite fields such that $|\mathbb{F}'| \geq |\mathcal{L}|$ and $|\mathbb{F}| \geq |\mathbb{F}'| \cdot |\mathcal{L}|$. We assume that we can identify the elements of $\mathcal{L}$ as either a subset of $\mathbb{F}'$ or $\mathbb{F}$ and we can also efficiently identify tuples in $\mathbb{F}' \times \mathcal{L}$ as a subset of $\mathbb{F}$. Let $\mathcal{M} = \mathbb{F}^m$, $\mathcal{R} = \mathbb{F}^\ell$, $\mathcal{K} = \mathbb{F}^{\ell+1} \times (\mathbb{F}')^{\ell+1}, \mathcal{T} = \mathbb{F}$. Define $\mathsf{MAC}_{\mathsf{key}}(\mathsf{lab}, \mathsf{msg}, r)$ as follows:

- Parse $\mathsf{key} = (\mathsf{key}_1, \mathsf{key}_2)$ where $\mathsf{key}_1 \in (\mathbb{F}')^{\ell+1}, \mathsf{key}_2 \in \mathbb{F}^{\ell+1}$.
- Compute $\mathsf{key}_1^{\mathsf{lab}} := \mathsf{PEval}(\mathsf{key}_1, \mathsf{lab})$, $\mathsf{key}_2^{\mathsf{lab}} := \mathsf{PEval}(\mathsf{key}_2, \mathsf{lab})$ by identifying $\mathsf{lab} \in \mathcal{L}$ as an element of $\mathbb{F}'$ and $\mathbb{F}$ respectively.

- Output $\sigma := \mathsf{PEval}(\ (r, \mathsf{msg})\ ,\ (\mathsf{lab}, \mathsf{key}_1^{\mathsf{lab}})\ ) + \mathsf{key}_2^{\mathsf{lab}}$. Here we interpret $(r, \mathsf{msg}) \in \mathcal{R} \times \mathcal{M} = \mathbb{F}^{\ell+m}$ as a vector of coefficients in $\mathbb{F}$ and we identify $(\mathsf{lab}, \mathsf{key}_{\mathsf{lab}}^1) \in \mathcal{L} \times \mathbb{F}'$ as an element of $\mathbb{F}$.

Theorem 3. *The above construction is an (ℓ, ε) private MAC, where $\varepsilon = \frac{m+\ell}{|\mathbb{F}'|}$.*

The proof is given in the full version [BPRW15].

4.3 Graph Identification

Here, we define an algorithmic problem called the graph identification problem. This abstracts out the core algorithmic problem that we face in designing our reconstruction algorithm.

Definition 7 (Graph Identification Challenge). *A graph identification challenge $\mathsf{Gen}^{\mathsf{Adv}}(n, t, d)$ is a randomized process that outputs a directed graph $G = (V = [n], E)$, where each vertex $v \in V$ has out-degree d, along with a labeling $L : E \to \{\mathsf{good}, \mathsf{bad}\}$. The process is parameterized by an adversary* Adv *and proceeds as follows.*

Adversarial Components. *The adversary $\mathsf{Adv}(n, t, d)$ does the following:*

1. *It partitions $V = [n]$ into three disjoint sets H, A, P (honest, active and passive) such that $V = H \cup A \cup P$ and $|A \cup P| = t$.*
2. *It chooses the set of edges $E_{A \cup P \to V}$ that originate from $A \cup P$ arbitrarily subject to each $v \in A \cup P$ having out-degree d and no self-loops.*
3. *It chooses the labels $L(e)$ arbitrarily for each edge $e \in E_{A \to (A \cup P)} \cup E_{(A \cup P) \to A}$.*

Honest Components. *The procedure* Gen *chooses the remaining edges and labels as follows:*

1. *It chooses the edges $E_{H \to V}$ that originate from H uniformly at random subject to each vertex having out-degree d and no self-loops. In particular, for each $v \in H$ it randomly selects d outgoing edges (without replacement) to vertices in $V \setminus \{v\}$.*
2. *It sets $L(e) :=$ bad for all $e \in E_{H \to A} \cup E_{A \to H}$.*
3. *It sets $L(e) :=$ good for all $e \in E_{(H \cup P) \to (H \cup P)}$.*

Output. *Output $(G = (V, E), L, H, A, P)$.*

The graph identification challenge is intended to model the authentication graph in our eventual robust secret sharing scheme where the labels will be assigned by verifying MAC tags. It allows us to abstract out a problem about graphs without needing to talk about MACs, secret shares, etc. We will need to show that any adversary on our full robust secret sharing scheme can be translated into an adversary in the graph identification challenge game above.

Note that, in the graph challenge game, we allow the adversary to *choose* the outgoing edges for both active and passive parties. This might seem unnecessary since the adversary in our eventual robust secret sharing scheme cannot *modify* the outgoing edges from passive parties in the authentication graph. However, it can choose which parties are active and which are passive after seeing the outgoing edges from all corrupted parties and therefore the adversary has some (limited) control over the outgoing edges of passive parties. In the definition of the graph challenge game, we therefore simply give the adversary complete control over all such outgoing edges.

As our main result for this problem, we show that there is an efficient algorithm that identifies the set $H \cup P$ given only the graph G and the labeling L. Note that, for our application, we do not crucially need to identify all of $H \cup P$; any subset of $H \cup P$ of size $t+1$ would be sufficient to reconstruct the message from the Shamir shares. However, this does not appear to make the task easier.

Theorem 4. *There exists a polynomial time algorithm* $\mathsf{GraphID}$*, called the* graph identification algorithm*, that takes as input a directed graph $G = (V = [n], E)$ a labeling $L : V \to \{\mathsf{good}, \mathsf{bad}\}$ and outputs a set $B \subseteq V$, such that for any* Adv*, we have:*

$$\Pr\left[B = H \cup P : \begin{array}{c}(G, L, H, A, P) \leftarrow \mathsf{Gen}^{\mathsf{Adv}}(n = 2t+1, t, d), \\ B \leftarrow \mathsf{GraphID}(G, L)\end{array}\right] \geq 1 - 2^{-\Omega(d/\log^2 n - \log n)}$$

In Section 7 we give a simple *inefficient* algorithm for the graph identification problem. Then, in Section 8, we prove Theorem 4 by providing an efficient algorithm.

5 Construction of Robust Secret Sharing

In this section we construct our robust secret sharing scheme using the tools outlined above. We analyze its security by translating an adversary on the scheme into an adversary in the graph identification game.

5.1 The Construction

Let $t, n = 2t+1$ be parameters that are given to us, and let $\mathcal{M}$ be a message space.

Let d be a graph out-degree parameter and let $\mathsf{GraphID}$ be the graph identification algorithm from Theorem 4 with success probability $1 - \delta_{gi}$ where $\delta_{gi} = 2^{-\Omega(d/\log^2 n)}$.

Let $(\mathsf{Share}_{nr}, \mathsf{Rec}_{nr})$ be a t-out-of-n *non-robust* secret sharing (e.g., Shamir secret sharing) with message space $\mathcal{M}$ and share space $\mathcal{S}_{nr}$.

Let $\{\mathsf{MAC}_{\mathsf{key}} : \mathcal{L} \times \mathcal{M}_{mac} \times \mathcal{R} \to \mathcal{T}\}_{\mathsf{key} \in \mathcal{K}}$ be an $(\ell, \varepsilon_{mac})$ *private MAC* with label space $\mathcal{L} = [n]^2 \times \{0,1\}$ and message space $\mathcal{M}_{mac} = \mathcal{S}_{nr} \times [n]^d \times \mathcal{K}$, where $\ell = 3d$.

Finally, let $(\mathsf{Share}_{rds}, \mathsf{Rec}_{rds})$ be a t-out-of-n robust distributed storage (no privacy) with message space $\mathcal{M}_{rds} = \mathcal{T}^{2dn}$, share space $\mathcal{S}_{rds}$ and with robustness δ_{rds}.

Our robust secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ is defined as follows.

$\mathsf{Share}(\mathsf{msg})$**.** On input a message $\mathsf{msg} \in \mathcal{M}$, the sharing procedure proceeds as follows:

S.1. Choose $(\widetilde{s}_1, \dots, \widetilde{s}_n) \leftarrow \mathsf{Share}_{nr}(\mathsf{msg})$ to be a non-robust secret sharing of msg.

S.2. Choose a uniformly random directed graph $G = ([n], E)$ with out-degree d, in-degree at most $2d$ and no self-loops as follows:

  (a) For each $i \in [n]$ choose a random set $E_i \subseteq [n] \setminus \{i\}$ of size $|E_i| = d$. Set
$$E := \{(i, j) \ : \ i \in [n], j \in E_i\}.$$

  (b) Check if there is any vertex in G with in-degree $> 2d$. If so, go back to step (a). [5]

S.3. For each $i \in [n]$, sample a random MAC key $\mathsf{key}_i \leftarrow \mathcal{K}$ and MAC randomness $r_i \leftarrow \mathcal{R}$.

For each $j \in E_i$ define
$$\sigma_{i \to j} := \mathsf{MAC}_{\mathsf{key}_i}((i, j, 0), (\widetilde{s}_j, E_j, \mathsf{key}_j), r_j),$$
$$\sigma_{i \leftarrow j} := \mathsf{MAC}_{\mathsf{key}_j}((i, j, 1), (\widetilde{s}_i, E_i, \mathsf{key}_i), r_i).$$

where we treat $(i, j, 0), (i, j, 1) \in \mathcal{L}$ as a label, and we treat $(\widetilde{s}_j, E_j, \mathsf{key}_j) \in \mathcal{M}_{mac}$ as a message.

S.4. For each $i \in [n]$ define $\mathsf{tags}_i = \{(\sigma_{i \to j}, \sigma_{i \leftarrow j})\}_{j \in E_i} \in \mathcal{T}^{2d}$ and define $\mathsf{tags} = (\mathsf{tags}_1, \dots, \mathsf{tags}_n) \in \mathcal{T}^{2nd}$. Choose $(p_1, \dots, p_n) \leftarrow \mathsf{Share}_{rds}(\mathsf{tags})$ using the robust distributed storage scheme.

S.5. For $i \in [n]$, define $s_i = (\widetilde{s}_i, E_i, \mathsf{key}_i, r_i, p_i)$ to be the share of party i. Output $(s_1, \dots, s_n)$.

$\mathsf{Rec}(s'_1, \dots, s'_n)$**.** On input $s'_1, \dots, s'_n$ with $s'_i = (\widetilde{s}'_i, E'_i, \mathsf{key}'_i, r'_i, p'_i)$ do the following.

R.0. If there is a set of exactly $t + 1$ values $W = \{i \in [n] \ : \ s'_i \neq \bot\}$ then output $\mathsf{Rec}_{nr}((\widetilde{s}'_i)_{i \in W})$. Else proceed as follows.

R.1. Reconstruct $\mathsf{tags}' = (\mathsf{tags}'_1, \dots, \mathsf{tags}'_n) = \mathsf{Rec}_{rds}(p'_1, \dots, p'_n)$. Parse $\mathsf{tags}'_i = \{(\sigma'_{i \to j}, \sigma'_{i \leftarrow j})\}_{j \in E'_i}$.

R.2. Define a graph $G' = ([n], E')$ by setting $E' := \{(i, j) \ : \ i \in [n], j \in E'_i\}$.

R.3. Assign a label $L(e) \in \{\mathsf{good}, \mathsf{bad}\}$ to each edge $e = (i, j) \in E'$ as follows. If the following holds:
$$\sigma'_{i \to j} = \mathsf{MAC}_{\mathsf{key}'_i}((i, j, 1), (\widetilde{s}'_j, E'_j, \mathsf{key}'_j), r'_j) \quad \text{and}$$
$$\sigma'_{i \leftarrow j} = \mathsf{MAC}_{\mathsf{key}'_j}((i, j, 1), (\widetilde{s}'_i, E'_i, \mathsf{key}'_i), r'_i)$$

then set $L(e) := \mathsf{good}$, else set $L(e) := \mathsf{bad}$.

R.4. Call the graph identification algorithm to compute $B \leftarrow \mathsf{GraphID}(G', L)$.

R.5. Choose a subset $B' \subseteq B$ of size $|B'| = t{+}1$ arbitrarily and output $\mathsf{Rec}_{nr}((\widetilde{s}'_i)_{i \in B'})$.

[5] This happens with negligible probability. However, we include it in the description of the scheme in order to get perfect rather than statistical privacy.

5.2 Security Analysis

Theorem 5. *The above scheme* $(\mathsf{Share}, \mathsf{Rec})$ *is a t-out-of-n δ-robust secret sharing scheme for $n = 2t + 1$ with robustness $\delta = \delta_{rds} + \delta_{gi} + dn\varepsilon_{mac} + n2^{-d/3}$.*

We prove Theorem 5 by separately proving that the scheme satisfies perfect privacy, perfect threshold reconstruction with erasures and adaptive δ-robustness in the following three lemmas.

Lemma 1. *The scheme* $(\mathsf{Share}, \mathsf{Rec})$ *satisfies perfect privacy.*

Proof. Let $I \subseteq [n]$ be of size $|I| = t$ and let $\mathsf{msg}, \mathsf{msg}' \in \mathcal{M}$ be any two values. We define a sequence of hybrids as follows:

Hybrid 0: This is $\mathsf{Share}(\mathsf{msg})_I = (s_i)_{i \in I}$. Each $s_i = (\widetilde{s}_i, E_i, \mathsf{key}_i, r_i, p_i)$.

Hybrid 1: In this hybrid, we change the sharing procedure to simply choose all tags $\sigma_{i \to j}$ and $\sigma_{j \leftarrow i}$ for any $j \notin I$ uniformly and independently at random. This is identically distributed by the "privacy over randomness" property of the MAC. In particular, we rely on the fact that the adversary does not see r_j and that there are at most $\ell = 3d$ tags of the form $\sigma_{i \to j}$ and $\sigma_{j \leftarrow i}$ for any $j \notin I$ corresponding to the total degree of vertex j. These are the only tags that rely on the randomness r_j and they are all created with distinct labels.

Hybrid 2: In this hybrid, we choose $(\widetilde{s}_1, \ldots, \widetilde{s}_n) \leftarrow \mathsf{Share}_{nr}(\mathsf{msg}')$. This is identically distributed by the perfect privacy of the non-robust secret sharing scheme. Note that in this hybrid, the shares $s_i \ : \ i \in I$ observed by the adversary do not contain any information about $\widetilde{s}'_j \ : \ j \notin I$.

Hybrid 3: This is $\mathsf{Share}(\mathsf{msg}')_I = (s_i)_{i \in I}$. Each $s_i = (\widetilde{s}_i, E_i, \mathsf{key}_i, r_i, p_i)$. This is identically distributed by the "privacy over randomness" property of the MAC, using same argument as going from Hybrid 0 to 1.

Lemma 2. *The scheme* $(\mathsf{Share}, \mathsf{Rec})$ *satisfies perfect threshold reconstruction with erasures.*

Proof. This follows directly from the fact that the non-robust scheme $(\mathsf{Share}_{nr}, \mathsf{Rec}_{nr})$ satisfies perfect threshold reconstruction with erasures and therefore step R.0 of reconstruction is guaranteed to output the correct answer when there are exactly t erasures.

Lemma 3. *The scheme* $(\mathsf{Share}, \mathsf{Rec})$ *is δ-robust for $\delta = \delta_{rds} + \delta_{gi} + dn\varepsilon_{mac} + n2^{-d/3}$.*

Proof overview. Before giving the formal proof of the lemma, we give a simplified proof intuition. To keep it simple, let's consider non-adaptive robustness experiment where the adversary has to choose the set $I \subseteq [n]$, $|I| = t$ of parties to corrupt at the very beginning of the game (in the full proof, we handle adaptive security). Let $s_i = (\widetilde{s}_i, E_i, \mathsf{key}_i, r_i, p_i)$ be the shares created by the sharing procedure and let $s'_i = (\widetilde{s}'_i, E'_i, \mathsf{key}'_i, r'_i, p'_i)$ be the modified shares submitted by the adversary (for $i \notin I$, we have $s'_i = s_i$). Let us define the set of *actively modified* shares as:

$$A = \{i \in I \ : \ (\widetilde{s}'_i, E'_i, \mathsf{key}'_i, r'_i) \neq (\widetilde{s}_i, E_i, \mathsf{key}_i, r_i)\}.$$

Define $H = [n] \setminus I$ to be the set of *honest* shares, and $P = I \setminus A$ to be the *passive shares.*

To prove robustness, we show that the choice of H, A, P and the labeling L created by the reconstruction procedure follow the same distribution as in the graph identification problem $\mathsf{Gen}^{\mathsf{Adv}'}(n, t, d)$ with some adversary Adv'. Therefore the graph identification procedure outputs $B = H \cup P$ which means that reconstruction outputs the correct message. Intuitively, we rely on the fact that: (1) by the privacy properties of the MAC the adversary does not learn anything about outgoing edges from honest parties and therefore we can think of them as being chosen randomly after the adversarial corruption stage, (2) by the authentication property of the MAC the edges between honest and active parties (in either direction) are labeled bad.

More concretely, we define a sequence of "hybrid" distributions to capture the above intuition as follows:

Hybrid 0. This is the non-adaptive version of the robustness game $\mathbf{Exp}(\mathsf{msg}, \mathsf{Adv})$ with a message msg and an adversary Adv as in Definition 3.

Hybrid 1. During reconstruction, instead of recovering $\mathsf{tags}' = \mathsf{Rec}_{rds}(p'_1, \ldots, p'_n)$ we just set $\mathsf{tags}' = \mathsf{tags}$ to be the correct value chosen by the sharing procedure. This is indistinguishable by the security of the robust-distributed storage scheme.

Hybrid 2. During the sharing procedure, we can change all of the tags $\sigma_{i\to j}, \sigma_{j\leftarrow i}$ with $j \in H$ to uniformly random values. This is identically distributed by the "privacy over randomness" property of the MAC since the adversary does not see r_j for any such $j \in H$, and there are at most $\ell = 3d$ such tags corresponding to the total degree of the vertex j. In particular, this means that such tags do not reveal any (additional) information to the adversary about E_j, key_j for $j \in H$.

Hybrid 3. During the reconstruction process, when the labeling L is created, we automatically set $L(e) = \mathsf{bad}$ for any edge $e = (i, j)$ or $e = (j, i)$ in E' such that $i \in H, j \in A$ (i.e., one end-point honest and the other active). The only time this introduces a change is if the adversary manages to forge a MAC tag under some key key_i for $i \in H$. Each such key was used to create at most $\ell = 3d$ tags with distinct labels and therefore, we can rely on the authentication security of the MAC to argue that this change is indistinguishable. Note that, by the definition of the labeling, we are also ensured that $L(e) = \mathsf{good}$ for any edge (i, j) where $i, j \in H \cup P$.

Hybrid 4. During the sharing procedure, we can change all of the tags $\sigma_{i\to j}, \sigma_{j\leftarrow i}$ with $i \in H$ to uniformly random values. This is identically distributed by the "privacy over keys" property of the MAC since the adversary does not see key_i for any such $i \in H$, and there are at most $\ell = 3d$ such tags corresponding to the total degree of the vertex i. In particular, this means that such tags do not reveal anything about the outgoing edges E_i for $i \in H$ and therefore the adversary gets no information about these edges throughout the game.

Hybrid 5. When we choose the graph $G = ([n], E)$ during the sharing procedure, we no longer require that every vertex has in-degree $\leq 2d$. Instead, we just choose each set $E_i \subseteq [n] \setminus \{i\}$, $|E_i| = d$ uniformly at random. Since the expected in-degree of every vertex is d, this change is indistinguishable by a simple Chernoff bound.

Hybrid 6. During reconstruction, instead of computing $B \leftarrow \mathsf{GraphID}(G', L)$ we set $B = H \cup P$. We notice that, in the previous hybrid, the distribution of G', L, H, A, P is exactly that of the graph reconstruction game $\mathsf{Gen}^{\mathsf{Adv}'}(n, t, d)$ with some adversary Adv'. In particular, the out-going edges from the honest set H are chosen uniformly at random and the adversary does not see any information about them throughout the game. Furthermore, the labeling satisfies the properties of the graph identification game. Therefore, the above modification is indistinguishable by the correctness of the graph identification algorithm.

In the last hybrid, the last step of the reconstruction procedure runs $\mathsf{msg}' = \mathsf{Rec}_{nr}((\widetilde{s}'_i)_{i \in B'})$ where $B' \subseteq H \cup P$ is of size $|B'| = t + 1$. Therefore and $\widetilde{s}'_i = \widetilde{s}_i$ for $i \in B'$ and, by the Perfect Reconstruction with Erasures property of the non-robust secret sharing scheme, we have $\mathsf{msg}' = \mathsf{msg}$. For a formal proof, see the full version [BPRW15].

5.3 Parameters of Construction

Let $\mathcal{M} = \{0,1\}^m$ and $t, n = 2t + 1$ be parameters. Furthermore, let λ be a parameter which we will relate to the security parameter k.

We choose the out-degree parameter $d = \lambda \log^3 n$, which then gives $\delta_{gi} = 2^{-\Omega(d/\log^2 n - \log n)} = 2^{-\Omega(\lambda \log n)}$.

We instantiate the non-robust secret scheme $(\mathsf{Share}_{nr}, \mathsf{Rec}_{nr})$ using t-out-of-n Shamir secret sharing where the share space $\mathcal{S}_{nr}$ is a binary field of size $2^{\max\{m, \lfloor \log n \rfloor + 1\}} = 2^{m + O(\log n)}$.

We instantiate the MAC using the construction from Section 4.2. We choose the field $\mathbb{F}'$ to be a binary field of size $|\mathbb{F}'| = 2^{\lceil 5 \log n + \log m + \lambda \rceil}$ which is sufficiently large to encode a label in $\mathcal{L} = [n]^2 \times \{0,1\}$. We choose the field $\mathbb{F}$ to be of size $|\mathbb{F}| = |\mathbb{F}'| 2^{2\lceil \log n \rceil + 1}$ which is sufficiently large to encode an element of $\mathbb{F}' \times \mathcal{L}$. We set $\ell = 3d = O(\lambda \log^3 n)$. This means that the keys and randomness have length $\log |\mathcal{K}|, \log |\mathcal{R}| = O(d \log |\mathbb{F}|) = O(\lambda \log^3 n(\lambda + \log n + \log m))$ and the tags have length $\log |\mathcal{T}| = \log |\mathbb{F}| = O(\lambda + \log n + \log m)$. We set the message space of the MAC to be $\mathcal{M}_{mac} = \mathbb{F}^{m_{mac}}$ which needs to be sufficiently large to encode the Shamir share, edges, and a key and therefore we set $m_{mac} = \lceil (\max\{m, \lfloor \log n \rfloor + 1\} + \log |\mathcal{K}| + d \log n) / \log |\mathbb{F}| \rceil = O(m + \lambda \log^3 n)$. This gives security $\varepsilon_{mac} = \frac{m_{mac} + \ell}{|\mathbb{F}'|} \leq 2^{\log m + \log \lambda + 3 \log n + O(1) - \log |\mathbb{F}'|} = 2^{-\Omega(\lambda) - 2 \log n}$.

Finally, we instantiate the robust distributed storage scheme using the construction from Section 4.1. We need to set the message space $\mathcal{M}_{rds} = \mathcal{T}^{2dn}$ which means that the messages are of length $m_{rds} = 2dn \log |\mathcal{T}| = O(n\lambda \log^3 n(\lambda +$

$\log n + \log m)$). We set $u = \lceil 8m_{rds}/n \rceil = O(\lambda \log^3 n(\lambda + \log n + \log m))$. This results in a robust-distributed storage share length $3u = O(\lambda \log^3 n(\lambda + \log n + \log m))$ and we get security $\delta_{rds} = O(n^2)/2^u \leq 2^{-\Omega(\lambda)}$.

With the above we get security

$$\delta \leq \delta_{rds} + \delta_{gi} + dn\varepsilon_{mac} + n2^{-d/3} = 2^{-\Omega(\lambda)} \tag{5.1}$$

and total share length $\log |\mathcal{S}_{nr}| + d\lceil \log n \rceil + \log |\mathcal{K}| + \log |\mathcal{R}| + 3u$ which is

$$m + O(\lambda \log^3 n(\lambda + \log n + \log m)) \tag{5.2}$$

By choosing a sufficiently large $\lambda = O(k)$ we get security $\delta \leq 2^{-k}$ and share size

$$m + O(k^2 \mathsf{polylog}(n + m)) = m + \widetilde{O}(k^2).$$

6 Improved Parameters via Parallel Repetition

In the previous section, we saw how to achieve robust secret sharing with security $\delta = 2^{-k}$ at the cost of having a share size $m + \widetilde{O}(k^2)$. We now show how to improve this to $m + \widetilde{O}(k)$. We do so by instantiating the scheme from the previous section with smaller parameters that only provide weak robustness $\delta = \frac{1}{4}$ and share size $m + \widetilde{O}(1)$. We then use parallel repetition of $q = O(k)$ independent copies of this weak scheme. The q copies of the recovery procedure recover q candidate messages, and we simply output the majority vote. A naive implementation of this idea, using q completely independent copies of the scheme, would result in share size $O(km) + \widetilde{O}(k)$ sine the (non-robust) Shamir share of length m is repeated q times. However, we notice that we can reuse the same Shamir shares across all q copies. This is because the robustness security held even for a worst-case choice of such shares, only over the randomness of the other components. Therefore, we only get a total share size of $m + \widetilde{O}(k)$.

Construction. In more detail, let $(\mathsf{Share}, \mathsf{Rec})$ be our robust secret sharing scheme construction from above. For some random coins coins_{nr} of the non-robust (Shamir) secret sharing scheme, we let $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(\mathsf{msg}; \mathsf{coins}_{nr})$ denote the execution of the sharing procedure $\mathsf{Share}(\mathsf{msg})$ where step S.1 uses the fixed randomness coins_{nr} to select the non-robust shares $(\widetilde{s}_1, \ldots, \widetilde{s}_n) \leftarrow \mathsf{Share}_{nr}(\mathsf{msg}; \mathsf{coins}_{nr})$ but steps S.2 – S.5 use fresh randomness to select the graph G, the keys key_i and the randomness r_i. In particular, $\mathsf{Share}(\mathsf{msg}; \mathsf{coins}_{nr})$ remains a randomized algorithm.

We define the q-wise parallel repetition scheme $(\mathsf{Share}', \mathsf{Rec}')$ as follows:

$\mathsf{Share}'(\mathsf{msg})$**:** The sharing procedure proceeds as follows

- Choose uniformly random coins_{nr} for the non-robust sharing procedure Share_{nr}.
- For $j \in [q]$: sample $(s_1^j, \ldots, s_n^j) \leftarrow \mathsf{Share}(\mathsf{msg}; \mathsf{coins}_{nr})$ where s_i^j equals $(\widetilde{s}_i, E_i^j, \mathsf{key}_i^j, r_i^j, p_i^j)$. Note that the non-robust (Shamir) shares $\widetilde{s}_i$ are the same in all q iterations but the other components are selected with fresh randomness in each iteration.

  - For $i \in [n]$, define the party i share as $s_i = (\widetilde{s}_i, \{(E_i^j, \mathsf{key}_i^j, r_i^j, p_i^j)\}_{j=1}^q)$ and output $(s_1, \ldots, s_n)$.

$\mathsf{Rec}'(s_1, \ldots, s_n)$**:** The reconstruction procedure proceeds as follows

  - For $i \in [n]$, parse $s_i = (\widetilde{s}_i, \{(E_i^j, \mathsf{key}_i^j, r_i^j, p_i^j)\}_{j=1}^q)$.
    For $j \in [q]$, define $s_i^j := (\widetilde{s}_i, E_i^j, \mathsf{key}_i^j, r_i^j, p_i^j)$.
  - For $j \in [q]$, let $\mathsf{msg}_j := \mathsf{Rec}(s_1^j, \ldots, s_n^j)$. If there is a majority value msg such that $|\{j \in [q] \ : \ \mathsf{msg} = \mathsf{msg}_j\}| > q/2$ then output msg, else output $\perp$.

Analysis. We prove that the parallel repetition scheme satisfies robustness. Assume that the parameters of $(\mathsf{Share}, \mathsf{Rec})$ are chosen such that the scheme is δ-robust.

We first claim that the scheme $(\mathsf{Share}, \mathsf{Rec})$ remains robust even if we fix the random coin coins_{nr} for the non-robust secret sharing scheme (in step S.1 of the Share function) to some worst-case value but use fresh randomness in all the other steps. The fact that coins_{nr} are random was essential for privacy but it does not affect robustness. In particular, let us consider the robustness experiment $\mathbf{Exp}(\mathsf{msg}, \mathsf{Adv})$ for the scheme $(\mathsf{Share}, \mathsf{Rec})$ and let us define $\mathbf{Exp}(\mathsf{msg}, \mathsf{Adv}; \mathsf{coins}_{nr})$ to be the experiment when using some fixed choice of coins_{nr} but fresh randomness everywhere else. We can strengthen the statement of Lemma 3 which proves the robustness of $(\mathsf{Share}, \mathsf{Rec})$ to show the following.

Lemma 4 (Strengthening of Lemma 3). *The scheme* $(\mathsf{Share}, \mathsf{Rec})$ *remains robust even if* coins_{nr} *is fixed to a worst-case value. In particular, for any* $\mathsf{msg} \in \mathcal{M}$*, any choice of* coins_{nr} *and for all adversaries* Adv *we have*

$$\Pr[\mathbf{Exp}(\mathsf{msg}, \mathsf{Adv}; \mathsf{coins}_{nr}) = 1] \leq \delta.$$

The proof of the above lemma follows the lines of that of lemma 3. See the full version [BPRW15] for more details.

Theorem 6. *Assume that the parameters of* $(\mathsf{Share}, \mathsf{Rec})$ *are chosen such that the scheme is δ-robust for $\delta \leq \frac{1}{4}$. Then the q-wise parallel repetition scheme* $(\mathsf{Share}', \mathsf{Rec}')$ *is a δ'-robust secret sharing scheme with $\delta' = e^{-\frac{3}{128}q}$.*

The proof of the above theorem is given in the full version [BPRW15].

Parameters. We choose the parameters of the underlying scheme $(\mathsf{Share}, \mathsf{Rec})$ to have security $\delta = \frac{1}{4}$. This corresponds to choosing a sufficiently large $\lambda = O(1)$ and results in a share size of $m + O(\log^4 n + \log^3 n \log m)$ bits (equation 5.2). By choosing a sufficiently large $q = O(k)$ and setting $(\mathsf{Share}', \mathsf{Rec}')$ to be the q-wise parallel repetition scheme from above, we get a scheme with robustness $\delta' = 2^{-k}$ and share size

$$m + O(k(\log^4 n + \log^3 n \log m)) = m + \widetilde{O}(k).$$

We note that part of the reason for the large poly-logarithmic factors comes from the parameters of our efficient graph identification algorithm which requires us to set the graph degree to $d = O(\log^3 n)$. If we had instead simply relied on our inefficient graph identification algorithm (Corollary 1) we could set $d = O(\log n)$ and would get an inefficient robust secret sharing scheme with share size $m + O(k(\log^2 n + \log n \log m))$. It remains an interesting challenge to optimize the poly-logarithmic factors.

7 Inefficient Graph Identification via Self-Consistency

We now return to the graph identification problem defined in Section 4.3. We begin by showing a simple inefficient algorithm for the graph identification problem. In particular, we show that with overwhelming probability the set $H \cup P$ is the unique maximum *self-consistent* set of vertices, meaning that there are no bad edges between vertices in the set.

Definition 8 (Self-Consistency). *Let $G = (V, E)$ be a directed graph and let $L : V \to \{\mathsf{good}, \mathsf{bad}\}$ be a labeling. We say that a subset of vertices $S \subseteq V$ is* self-consistent *if for all $e \in E_{S \to S}$ we have $L(e) = \mathsf{good}$. A subset $S \subseteq V$ is* max self-consistent *if $|S| \geq |S'|$ for every self-consistent $S' \subseteq V$.*

Note that, in general, there may not be a unique max self-consistent set in G. However, the next lemma shows that if the components are sampled as in the graph identification challenge game $\mathsf{Gen}^{\mathsf{Adv}}(n, t, d)$, then with overwhelming probability there is a unique max self-consistent set in G and it is $H \cup P$.

Lemma 5. *For any Adv, and for the distribution $(G, L, H, A, P) \leftarrow \mathsf{Gen}^{\mathsf{Adv}}(n, t, d)$, the set $H \cup P$ is the unique max self-consistent set in G with probability at least $1 - 2^{-\Omega(d - \log n)}$.*

Proof. We know that the set $H \cup P$ is self-consistent by the definition of the graph identification challenge. Assume that it is not the unique max self-consistent set in G, which we denote by the event BAD. Then there exists some set $S \neq H \cup P$ of size $|S| = |H \cup P|$ such that S is self consistent. This means that S must contain at least $q \geq 1$ elements from A and at least $t + 1 - q$ elements from H. In other words there exists some value $q \in \{1, \ldots, t\}$ and some subsets $A' \subseteq S \cap A \subseteq A \subseteq A \cup P$ of size $|A'| = q$ and $H' \subseteq S \cap H \subseteq H$ of size $t + 1 - q$ such that $E_{H' \to A'} = \varnothing$. This is because, by the definition of the graph challenge game, every edge in $E_{H' \to A'} \subseteq E_{H \to A}$ is labeled bad and so it must be empty if S is consistent. For any fixed q, A', H' as above, if we take the probability over the random choice of d outgoing edges for each $v \in H'$, we get:

$$\Pr[E_{H' \to A'} = \varnothing] = \left(\frac{\binom{n-1-q}{d}}{\binom{n-1}{d}} \right)^{t+1-q} \leq \left(1 - \frac{q}{n-1} \right)^{d(t+1-q)} \leq e^{-\frac{d(t+1-q)q}{n}}.$$

By taking a union bound, we get

$$\begin{aligned}
\Pr[\mathsf{BAD}] &\leq \Pr\left[\exists \left\{ \begin{array}{c} q \in \{1,\ldots,t\} \\ A' \subseteq A \cup P : |A|' = q \\ H' \subseteq H, |H'| = t+1-q \end{array} \right\} \ : \ E_{H' \to A'} = \varnothing \right] \\
&\leq \sum_{q=1}^{t} \binom{t+1}{t+1-q} \cdot \binom{t}{q} \cdot e^{-\frac{d(t+1-q)q}{n}} \leq \sum_{q=1}^{t} \binom{t+1}{t+1-q} \cdot \binom{t+1}{q} \cdot e^{-\frac{d(t+1-q)q}{n}} \\
&\leq 2 \sum_{q=1}^{(t+1)/2} \binom{t+1}{q}^2 \cdot e^{-\frac{d(t+1-q)q}{n}} \\
&\hspace{15em} \text{(symmetry between } q \text{ and } t+1-q\text{)} \\
&\leq 2 \sum_{q=1}^{(t+1)/2} (t+1)^{2q} \cdot e^{-\frac{d(t+1-q)q}{n}} \\
&\leq 2 \sum_{q=1}^{(t+1)/2} e^{q\left(2\log_e(t+1) - \frac{d(t+1-q)}{n}\right)} \\
&\leq 2 \sum_{q=1}^{(t+1)/2} e^{q\left(2\log_e(t+1) - \frac{(t+1)d}{2n}\right)} && \text{(since } q \leq (t+1)/2\text{)} \\
&\leq 2 \sum_{q=1}^{(t+1)/2} e^{q(2\log_e(t+1) - d/4)} && \text{(since } t+1 > n/2\text{)} \\
&\leq (t+1)e^{(2\log_e(t+1) - d/4)} \leq 2^{-\Omega(d - \log n)}
\end{aligned}$$

As a corollary of the above lemma, we get an inefficient algorithm for the graph identification problem, that simply tries every subset of vertices $S \subseteq V$ and outputs the max self-consistent set.

Corollary 1. *There exists an inefficient algorithm* $\mathsf{GraphID}^{\mathsf{ineff}}$ *such that for any* Adv*:*

$$\Pr\left[B = H \cup P \quad : \quad \begin{array}{c} (G, L, H, A, P) \leftarrow \mathsf{Gen}^{\mathsf{Adv}}(n = 2t+1, t, d), \\ B \leftarrow \mathsf{GraphID}^{\mathsf{ineff}}(G, L) \end{array} \right] \geq 1 - 2^{-\Omega(d - \log n)}$$

Remark. Note that for the analysis of the inefficient graph reconstruction procedure in Lemma 5 and Corollary 1, we did not rely on the fact that edges from active to honest parties $e = (i, j) \ : \ i \in A, j \in H$ are labeled bad. Therefore, if we only wanted an inefficient graph identification procedure, we could relax the requirements in the graph challenge game and allow the adversary to choose arbitrary labels for such edges e. This would also allow us to simplify our robust secret sharing scheme and omit the "reverse-authentication" tags $\sigma_{i \leftarrow j}$.

8 Efficient Graph Identification

In this section, we prove Theorem 4 and given an efficient graph identification algorithm. We begin with an intuitive overview before giving the technical details.

8.1 Overview and Intuition

A Simpler Problem. We will reduce the problem of identifying the full set $H \cup P$ to the simpler problem of only identifying a small set $Y \subseteq H \cup P$ such that $Y \cap H$ is of size at least εn for some $\varepsilon = 1/\Theta(\log n)$. If we are given such a Y, we can use it to identify a larger set S defined as all vertices in $[n]$ with no bad incoming edge originating in Y. We observe that every vertex in $H \cup P$ is included in S, as there are no bad edges from $H \cup P$ to $H \cup P$. On the other hand, since $Y \cap H$ is big enough, it is unlikely that a vertex in A could be included in S, as every vertex in A likely has an incoming edge from $|Y \cap H|$ that is labeled as bad. Therefore, with high probability $S = H \cup P$. There is a bit of subtlety involved in applying this intuition, as it is potentially complicated by dependencies between the formation of the set Y and the distribution of the edges from Y to A. We avoid dealing with such dependencies by "splitting" the graph into multiple independent graphs and building Y from one of these graphs while constructing S in another.

Now the task becomes obtaining such a set Y in the first place. We consider two cases depending on whether the set P is small ($|P| \leq \varepsilon n$) or large ($|P| > \varepsilon n$).

Small P. In this case, there is only a small number of good edges crossing between H and $A \cup P$ (only edges between H and P). Therefore there exists a bisection of the graph into sets H and $A \cup P$ of size $t+1$ and t respectively, where the number of good edges crossing this bisection is approximately εdn. By using an efficient $O(\log n)$-approximation algorithm for the graph bisection problem (on the good edges in G) we can get a bisection X_0, X_1 with very few edges crossing between X_0 and X_1. This means that, with overwhelming probability, one of X_0 or X_1 contains the vast majority of the honest vertices, say $.9|H|$, as otherwise if the honest vertices were split more evenly, we'd expect more edges crossing this partition. We can then refine such an X to get a suitable smaller subset Y which is fully contained in $H \cup P$, by taking all vertices that don't have too many incoming bad edges from X.

Large P. In this case, the intuition is that every vertex in A is likely to have at least $d/2$ in-coming bad edges (from the honest vertices), but honest/passive vertices will only have $d(1/2 - \varepsilon)$ in-coming bad edges on average from the active vertices. So we can differentiate the two cases just by counting. This isn't precise since many active vertices can point bad edges at a single honest vertex to make it "look bad". However, intuitively, this cannot happen too often.

To make this work, we first start with the full set of vertices $[n]$ and disqualify any vertices that have more than $d/2$ out-going bad edges (all honest vertices remain since they only have $d(1/2-\varepsilon)$ outgoing bad edges on expectation). This potentially eliminates some active vertices. Let's call the remaining smaller set of vertices X. We then further refine X into a subset Y of vertices that do not have too many incoming bad edges (more than $d(1/2-\varepsilon/2)$) originating in X. The active vertices are likely to all get kicked out in this step since we expect $d/2$ incoming bad edges from honest vertices. On the other hand, we claim that not too many honest vertices get kicked out. The adversary has at most $(1/2-\varepsilon)dn/2$ out-going bad edges in total under his control in the set $X \cap A$ and has to spend $d(1/2-\varepsilon/2)$ edges to kick out any honest party . Therefore there is a set of at least $\varepsilon n/2$ of the honest parties that must survive. This means that $Y \subseteq H \cup P$ and that Y contains $\Theta(n/\log n)$ honest parties as we wanted.

Unknown P. Of course, our reconstruction procedure will not know a priori whether P is relatively large or small or, in the case that P is small, which one of the bisection sets X_1 or X_2 to use. So it simply tries all of these possibilities and obtains three candidate sets Y_0, Y_1, Y_2, one of which has the properties we need but we do not know which one. To address this, we construct the corresponding sets S_i for each Y_i as described above, and we know that one of these sets S_i is $H \cup P$. From the previous section (Lemma 5), we also know that $H \cup P$ is the unique max self-consistent set in G. Therefore, we can simply output the largest one of the sets S_0, S_1, S_2 which is self-consistent in G and we are guaranteed that this is $H \cup P$.

8.2 Tool: Graph Splitting

As mentioned above, we will need to split the graph G into three sub-graphs G^1, G^2, G^3 such that the outgoing edges from honest parties are distributed randomly and independently in G^1, G^2, G^3. Different parts of our algorithm will use different sub-graphs and it will be essential that we maintain independence between them for our analysis.

In particular, we describe a procedure $(G^1, G^2, G^3) \leftarrow \mathsf{GraphSplit}(G)$ that takes as input a directed graph $G = (V = [n], E)$ produced by $\mathsf{Gen}^{\mathsf{Adv}}(n, t, d)$ and outputs three directed graphs $(G^i = (V, E^i))_{i=1,2,3}$ such that $E^i \subset E$ and the out-degree of each vertex in each graph is $d' := \lfloor d/3 \rfloor$. Furthermore, we require that the three sets $E^i_{H \to V}$ are random and independent subject to each vertex having out-degree d' and no self-loops. Note that forming the sets E^i by simply partitioning the outgoing edges of each vertex into three sets is *not* a good solution, since in that case the sets will always be disjoint and therefore not random and independent. On the other hand, sub-sampling three random subsets of d' outgoing edges from the set of d outgoing edges in E is also not a good solution since in the case the overlap between the sets is likely to be higher than it would be if we sampled random subsets of d' outgoing edges from all possible edges.

Our algorithm proceeds as follows.

$(G^1, G^2, G^3) \leftarrow \mathsf{GraphSplit}(G)$**:** On input a directed graph $G = (V, E)$ with out-degree d.

1. Define $d' = \lfloor d/3 \rfloor$.
2. For each vertex, for each $v \in V$:
   (a) Define $N_v := \{w \in V \mid (v, w) \in E\}$, the set of neighbors of v in G
   (b) Sample three uniform and independent sets $\{N_v^i\}_{i=1,2,3}$ with $N_v^i \subseteq V \setminus \{v\}$ and $|N_v^i| = d'$.
   (c) Sample a uniformly random injective function $\pi_v : \bigcup_{i=1,2,3} N_v^i \to N_v$.
   (d) Define $\hat{N}_v^i = \pi_v(N_v^i) \subseteq N_v$.
3. Define $E^i := \{(v, w) \in E \mid w \in \hat{N}_v^i\}$ and output $G^i = (V, E^i)$ for $i = 1, 2, 3$.

Intuitively, for each vertex v, we first sample the three sets of outgoing neighbors N_v^i independently at random from all of $V \setminus \{v\}$, but then we apply an injective function $\pi_v(N_v^i)$ to map them into a the original neighbors N_v. The last step ensures that $E^i \subseteq E$.

Lemma 6. *Let $(G = (V, E), L, H, A, P) \leftarrow \mathsf{Gen}^{\mathsf{Adv}}(n, t, d)$ for some adversary Adv. Let $(G^i = (V, E^i))_{i=1,2,3} \leftarrow \mathsf{GraphSplit}(G)$. Then the joint distribution of $(E^i_{H \to V})_{i=1,2,3}$ is identical to choosing each set $E^i_{H \to V}$ randomly and independently subject to each vertex having out-degree d' and no self-loops; i.e., for each $i = 1, 2, 3$ form the set $E^i_{H \to V}$ by taking each $v \in H$ and choosing a set of d' outgoing edges uniformly at random (without replacement) to vertices in $V \setminus \{v'\}$.*

Proof. For each $v \in H$, define $c_{\{1,2\}} = |N_v^1 \cap N_v^2|, c_{\{1,3\}} = |N_v^1 \cap N_v^3|, c_{\{2,3\}} = |N_v^2 \cap N_v^3|$ and $c_{\{1,2,3\}} = |N_v^1 \cap N_v^2 \cap N_v^3|$. We call these numbers the *intersection pattern of* $\{N_v^i\}_{i=1,2,3}$ and denote it by C. Analogously, we define the *intersection pattern of* $\{\hat{N}_v^i\}_{i=1,2,3}$ and denote it by $\hat{C}$.

It's easy to see that, for any fixed choice of $\{N_v^i\}_{i=1,2,3}$ with intersection pattern C, the sets $\{\hat{N}_v^i\}_{i=1,2,3}$ are uniformly random and independent subject to their intersection pattern being $\hat{C} = C$. This follows from the random choice of N_v and the injective function π_v.

Furthermore, since the distribution of the intersection pattern $\hat{C} = C$ is the same for $\{N_v^i\}_{i=1,2,3}$ and for $\{\hat{N}_v^i\}_{i=1,2,3}$, the distribution of $\{\hat{N}_v^i\}_{i=1,2,3}$ is identical to that of $\{N_v^i\}_{i=1,2,3}$. In other words, for each $v \in H$ the three sets of outgoing neighbors of v in G^1, G^2, G^3 are random and independent as we wanted to show.

8.3 The Graph Identification Algorithm

We now define the efficient graph identification algorithm $B \leftarrow \mathsf{GraphID}(G, L)$.

Usage. Our procedure $\mathsf{GraphID}(G, L)$ first runs an initialization phase ***Initialize***, and then runs two procedures ***Small*** P and ***Large*** P sequentially. It uses the data generated in these two procedures to then run the output phase **Output**.

Initialize.

1. Let b be a constant such that there exists a polynomial-time $b \log n$-approximate graph bisection algorithm Bisect, such as the one provided in [Räc08]. Let $c = \frac{800}{9} b$, and let $\varepsilon = 1/(c \cdot \log(n))$.
2. Run $(G^1, G^2, G^3) \leftarrow \mathsf{GraphSplit}(G)$ as defined in section 8.2 . This produces three graphs $G^i = (V, E^i)$ such that $E^i \subseteq E$ and the out-degree of each vertex in G^i is $d' = \lfloor d/3 \rfloor$.

Small P**.**

1. Run $(X_0, X_1) \leftarrow \mathsf{Bisect}(G^*)$, where $G^* = (V, E^*)$ is the undirected graph induced by the good edges of G^1:

$$E^* = \left\{ \{i, j\} \;:\; \begin{array}{l} (e = (i,j) \in E^1 \text{ and } L(e) = \mathsf{good}) \text{ or} \\ (e = (j,i) \in E^1 \text{ and } L(e) = \mathsf{good}) \end{array} \right\}$$

2. For $i = 0, 1$: contract X_i to a set of *candidate good* vertices Y_i that have fewer than $0.4d'$ incoming bad edges in the graph G^2.

$$Y_i := \left\{ v \in X_i \;:\; \left| \{ e \in E^2_{X_i \to \{v\}} \;:\; L(e) = \mathsf{bad} \} \right| < 0.4d' \right\}.$$

Large P**.**

1. Define a set of *candidate legal* vertices X_2 as the set of vertices having fewer than $d'/2$ outgoing bad edges in G^1.

$$X_2 := \left\{ v \in V \;:\; \left| \{ e \in E^1_{\{v\} \to V} \;:\; L(e) = \mathsf{bad} \} \right| < d'/2 \right\}.$$

2. Contract X_2 to a set of *candidate good* vertices Y_2, defined as the set of vertices in X_2 having fewer than $d'\,(1/2 - \varepsilon/2)$ incoming bad edges from legal vertices in the graph G^1.

$$Y_2 := \left\{ v \in X_2 \;:\; \left| \{ e \in E^1_{X_2 \to \{v\}} \;:\; L(e) = \mathsf{bad} \} \right| < d'\,(1/2 - \varepsilon/2) \right\}.$$

Output. This subprocedure takes as input the sets Y_0, Y_1 (generated by **Small** P), and Y_2 (generated by **Large** P) and outputs a single set B, according to the following algorithm.

1. For $i = 0, 1, 2$: define S_i as the set of vertices that only have incoming good edges from Y_i in G^3. Formally,

$$S_i := \left\{ \;\; v \in V \;:\; \forall e \in E^3_{Y_i \to v} \;\; L(e) = \mathsf{good} \;\; \right\}$$

2. For $i = 0, 1, 2$: if $L(e) = \mathsf{good}$ for all $e \in E_{S_i \to S_i}$, define $B_i := S_i$; otherwise, define $B_i = \varnothing$. This ensures that each B_i is self-consistent (Definition 8) in G.
3. Output a set B defined as any of the largest sets among B_0, B_1, B_2.

8.4 Analysis of Correctness – Overview

In this section, we give an intuition for why the algorithm $\mathsf{GraphID}$ outlined above satisfies Theorem 4.

We first fix an arbitrary adversary Adv in the graph challenge game. We consider the distribution induced by running the randomized processes $(G, L, H, A, P) \leftarrow \mathsf{Gen}^{\mathsf{Adv}}(n = 2t+1, t, d)$ and $B \leftarrow \mathsf{GraphID}(G, L)$. Note that without loss of generality we can assume Adv is deterministic and therefore the sets H, A, P are fixed. The only randomness in experiment consists of the choice of edges $E_{H\to V}$ in the execution of $\mathsf{Gen}^{\mathsf{Adv}}(n = 2t + 1, t, d)$ and the randomness of the graph splitting procedure $(G^1, G^2, G^3) \leftarrow \mathsf{GraphSplit}(G)$ during the execution of $\mathsf{GraphID}(G, L)$. By the property of graph splitting (Lemma 6) we can think of this as choosing three independent sets $(E^i_{H\to V})_{i=1,2,3}$. This induces a distribution on the sets X_i, Y_i, S_i, B_i and B defined during the course of the $\mathsf{GraphID}$ algorithm and we analyze the probability of various events over this distribution.

We define a *sufficient event*:

$$O := \text{“there exists } i \in \{0, 1, 2\} \text{ such that } |Y_i \cap H| \geq \varepsilon \cdot n/2 \text{ and } Y_i \subseteq H \cup P\text{”}$$

In the full version [BPRW15], we prove the following technical lemmas that allow us to prove Theorem 4.

Lemma 7. *The conditional probability of $B = H \cup P$, given the occurrence of event O, is $1 - 2^{-\Omega(d/\log n)}$.*

Lemma 8. *If $|P| < \varepsilon \cdot n$, then the probability that O occurs is at least $1 - 2^{-\Omega(d/\log n)}$.*

Lemma 9. *If $|P| \geq \varepsilon \cdot n$, then the probability that O occurs is at least $1 - 2^{-\Omega(d/\log^2 n - \log n)}$.*

In Lemma 7, the probability is over the random edges $E^3_{H\to V}$ in G^3, while in Lemmas 8 and 9, the probability is over the random edges $E^i_{H\to V}$ in G^i for $i = 1, 2$. Therefore, conditioning on the event O in Lemma 7 does not effect the probability distribution.

$$\begin{tikzcd}
{|P| \leq \varepsilon \cdot n} \arrow[dr] & & {|P| > \varepsilon \cdot n} \arrow[dl] \\
& O \arrow[d] & \\
& B = H \cup P &
\end{tikzcd}$$

Fig. 1. Structure of our analysis: arrows denote logical implications (happening with high probability).

Our analysis is summarized in figure 1. We now complete the proof of Theorem 4.

Proof of Theorem 4. By Lemmas 8 and 9, we obtain that the event O occurs with probability at least $1-2^{-\Omega(d/\log^2 n-\log n)}$. Putting together with Lemma 7, we obtain that the probability that the set B returned by the algorithm equals $H \cup P$ is at least $1-2^{-\Omega(d/\log^2 n-\log n)}$ completing the proof of the theorem.

9 Conclusion

We constructed an efficient robust secret sharing scheme for the maximal corruption setting with $n = 2t+1$ parties with nearly optimal share size of $m+\widetilde{O}(k)$ bits, where m is the length of the message and 2^{-k} is the failure probability of the reconstruction procedure with adversarial shares.

One open question would be to optimize the poly-logarithmic terms in our construction. It appears to be an interesting question to attempt to go all the way down to $m + O(k)$ or perhaps even just $m + k$ bits for the share size, or to prove a lower bound that (poly)logarithmic factors in n, m are necessary. We leave this as a challenge for future work.

References

Bla79. George Robert Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, pages 313–317. IEEE Computer Society, 1979.

BPRW15. Allison Bishop, Valerio Pastro, Rajmohan Rajaraman, and Daniel Wichs. Essentially optimal robust secret sharing with maximal corruptions. *IACR Cryptology ePrint Archive*, 2015:1032, 2015.

BS97. Carlo Blundo and Alfredo De Santis. Lower bounds for robust secret sharing schemes. *Inf. Process. Lett.*, 63(6):317–321, 1997.

CDD$^+$15. Ronald Cramer, Ivan Bjerre Damgård, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In Elisabeth Oswald and Marc Fischlin, editors, *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*, volume 9057 of *Lecture Notes in Computer Science*, pages 313–336. Springer, 2015.

CDF01. Ronald Cramer, Ivan Damgård, 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, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 503–523. Springer, 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, 27th Annual International Conference on* *the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2008.

Cev11. Alfonso Cevallos. Reducing the share size in robust secret sharing. http://www.algant.eu/documents/theses/cevallos.pdf, 2011.

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, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 195–208. Springer, 2012.

Che15. Mahdi Cheraghchi. Nearly optimal robust secret sharing. *IACR Cryptology ePrint Archive*, 2015:951, 2015.

CSV93. 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, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 118–125. Springer, 1993.

FK02. Uriel Feige and Robert Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM J. Comput.*, 31(4):1090–1118, 2002.

GJS76. M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.

JS13. Mahabir Prasad Jhanwar and Reihaneh Safavi-Naini. Unconditionally-secure robust secret sharing with minimum share size. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, volume 7859 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2013.

LP14. Allison Bishop Lewko and Valerio Pastro. Robust secret sharing schemes against local adversaries. *IACR Cryptology ePrint Archive*, 2014:909, 2014.

Räc08. Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 255–264. ACM, 2008.

RB89. Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85. ACM, 1989.

Sha79. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

Sud97. Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.