Towards Optimal Robust Secret Sharing with Security Against a Rushing Adversary

Serge Fehr[1,2] and Chen Yuan[1]

[1] CWI, Amsterdam, The Netherlands
[2] Mathematical Institute, Leiden University, The Netherlands

Abstract. Robust secret sharing enables the reconstruction of a secret-shared message in the presence of up to t (out of n) *incorrect* shares. The most challenging case is when $n = 2t + 1$, which is the largest t for which the task is still possible, up to a small error probability $2^{-\kappa}$ and with some overhead in the share size.
Recently, Bishop, Pastro, Rajaraman and Wichs [3] proposed a scheme with an (almost) optimal overhead of $\widetilde{O}(\kappa)$. This seems to answer the open question posed by Cevallos et al. [6] who proposed a scheme with overhead of $\widetilde{O}(n+\kappa)$ and asked whether the linear dependency on n was necessary or not. However, a subtle issue with Bishop et al.'s solution is that it (implicitly) assumes a *non-rushing* adversary, and thus it satisfies a *weaker* notion of security compared to the scheme by Cevallos et al. [6], or to the classical scheme by Rabin and BenOr [13].
In this work, we almost close this gap. We propose a new robust secret sharing scheme that offers full security against a rushing adversary, and that has an overhead of $O(\kappa n^{\varepsilon})$, where $\varepsilon > 0$ is arbitrary but fixed. This n^{ε}-factor is obviously worse than the polylog(n)-factor hidden in the $\widetilde{O}$ notation of the scheme of Bishop et al. [3], but it greatly improves on the linear dependency on n of the best known scheme that features security against a rushing adversary (when κ is substantially smaller than n).
A small variation of our scheme has the same $\widetilde{O}(\kappa)$ overhead as the scheme of Bishop et al. *and* achieves security against a rushing adversary, but suffers from a (slightly) superpolynomial reconstruction complexity.

1 Introduction

Background. Robust secret sharing is an extended version of secret sharing as originally introduced by Shamir [14] and Blakley [4], where the reconstruction is required to work even if some of the shares are incorrect (rather than missing, as in the standard notion). Concretely, a robust secret sharing scheme needs to satisfy t-privacy: any t shares reveal no information on the secret, as well as t-robust-reconstructability: as long as no more than t shares are incorrect the secret can be reconstructed from the full set of n (partly correct, partly incorrect) shares. For $t < n/3$ this can easily be achieved by means of error correct techniques, whereas for $t \geq n/2$ the task is impossible. Thus, the interesting region is $n/3 \leq t < n/2$, respectively $n = 2t + 1$ if we want t maximal, where robust secret sharing is possible but only up to a small error probability $2^{-\kappa}$ (which can be controlled by a statistical security parameter κ) and only with some overhead in the share size (beyond the size of the "ordinary", e.g., Shamir share). There are many works [13, 5, 8, 6, 9, 2, 7, 3, 11] in this direction.[3]

The classical scheme proposed by Rabin and BenOr [13] has an overhead in share size of $O(\kappa n)$, i.e., next to the actual share of the secret, each player has to hold an additional $O(\kappa n)$ bits of information as part of his share. This additional information is in the form of $n-1$ authentication tags and keys. Concretely, every player P_i holds $n-1$ authentication keys $key_{i,j}$ that allow him to verify the (Shamir) shares s_j of all parties P_j, plus $n-1$ authentication tags $\sigma_{i,j}$ that allow the other parties to verify his share s_i by means of their keys. By this way, the honest parties can recognize all incorrect shares — and, in case the reconstructor is not a share holder, he would keep those shares that are correctly verified by at least $t+1$ other parties and dismiss the others (note that a dishonest share holder may also lie about his authentication key, and thus make look a a correct share incorrect).

Cevallos, Fehr, Ostrovsky and Rabani [6] proposed an improvement, which results in an overhead in share size of $\widetilde{O}(n+\kappa)$ instead. The core insight is that in the Rabin-BenOr scheme, one can reduce the size of the authentication keys and tags (and thus weaken the security of the authentication) at the expense of a slightly more involved reconstruction procedure — and a significantly more involved analysis. Since the linear dependency of the overhead on κ is unavoidable, they posed the question of whether the linear dependency on n is necessary, or whether an overhead of $\widetilde{O}(\kappa)$ is possible.

Bishop, Pastro, Rajaraman and Wichs [3] gave a positive answer to this question by proposing a scheme that indeed has an overhead in share size of $\widetilde{O}(\kappa)$. At first glance, this seems to settle the case. However, a subtle issue is that their scheme is proven secure only against a weaker attacker than what is considered in the above works. Concretely, the security of their scheme relies on the (implicit) assumption that the attacker is *non-rushing*, whereas the above discussed schemes remain secure in the presence of a *rushing* attacker. As such, the open question of Cevallos et al. [6] is not fully answered.

Recall that for the attacker to be rushing, it means that during the reconstruction procedure, when the parties announce their shares, he can decide on the incorrect shares of the corrupt parties *depending* on the shares that the honest parties announce (rather than on the shares of the corrupt parties alone). This is in particular meaningful and desirable if it is the parties themselves that do the reconstruction — in this case there is little one can do to prevent the corrupt parties from waiting and receiving the honest parties's shares, and then "rush" and announce their own shares before the end of the communication round. Even if the shares (which may include authentication tags and keys etc.) are announced gradually, in multiple rounds, in each round the attacker can still rush in that sense.

[3] In particular, [9, 7, 3, 11] use partly similar tools than we do but achieve weaker or incomparable results.

To the best of our knowledge, it has not been pointed out before that the scheme of Bishop et al. does not (necessarily) offer security against a rushing attacker. It is also not explicitly discussed in [3], but becomes clear when inspecting the considered security definition carefully.

The Scheme in [3]. We briefly discuss some of the features of the scheme by Bishop, Pastro, Rajaraman and Wichs [3], and why it is not secure against a rushing adversary. Like the schemes above, their scheme is also based on verification of shares by means of pairwise authentication using a message authentication code (MAC). However, in order to reduce the number of keys and tags so as to obtain an overhead that is independent of n, every party can now verify only a *subset* of the shares of the other parties, where the subset is randomly chosen (during the sharing phase) and of constant size. However, this makes the reconstruction procedure much more delicate, and Bishop et al. [3] need to pair this basic idea with various additional clever tricks in order to get the reconstruction working. One of these enhancements is that they need to avoid that a dishonest party can make an honest party look dishonest by announcing an incorrect authentication key without being identified as a cheater by other honest parties. This is done by authenticating not only the Shamir share of the party under consideration, but also that party's authentication keys. Concretely, if P_j is chosen to be one of the parties that P_i can verify, then this verification is enabled by means of an authentication tag $\sigma_{i,j}$ that is computed as

$$\sigma_{i,j} = MAC_{key_{i,j}}(s_j, key_j)$$

where $key_{i,j}$ is P_i's verification key, and key_j is the collection of keys that P_j holds for verification of the shares (and keys) that he can verify.[4]

It is now not hard to see that this construction design is inherently insecure against a rushing adversary. Even if the reconstruction is done in multiple rounds where first the Shamir shares and authentication tags are announced and only then the keys (which is what one has to do to make the Rabin-BenOr scheme and the scheme by Cevallos et al. secure in the rushing setting), given that a rushing adversary can choose an incorrect key_j *depending* on the authentication key $key_{i,j}$, the MAC offers no security. Worse, this cannot be fixed by, say, enhancing the MAC: either the adversary has some freedom in choosing incorrect key_j once given $key_{i,j}$, or then it is uniquely determined by $key_{i,j}$ and so P_i knows it — and so it cannot serve the purpose of an authentication key.[5]

We emphasize that we do not claim an explicit rushing attack against the scheme of Bishop et al. [3]. What the above shows is the existence of an attack that prevents a certain property on the consistency graph to hold upon which

[4] One might feel uncomfortable about that there seems to be some circularity there; but it turns out that this is no issue.

[5] The actual scheme is significantly more involved than the simplifies exposition given here, e.g., the identities of the parties that P_j can verify are authenticated as well, and the authentication tags are not stored "locally" but in a "robust and distributed" manner, but the issue pointed out here remains.

the reconstruction procedure appears to crucially rely — certainly the proof does. Thus, our claim is that we see no reason why the scheme of Bishop et al. should offer security against a rushing adversary.

Our Result. In this work, we propose a new robust secret sharing scheme. Our new scheme is secure against a rushing adversary *and* close to optimal in terms of overhead. By "close to optimal" we mean that our new scheme has an overhead of $O(\kappa n^{\varepsilon})$ and runs in polynomial time for any arbitrary fixed constant $\varepsilon > 0$. This is obviously slightly worse than the scheme of Bishop et al. [3], which has an overhead of $O(\kappa \cdot \mathrm{polylog}(n))$, but it greatly improves over the best known scheme that features security against a rushing adversary when n is significantly larger than κ.

Our approach recycles some of the ideas from Bishop et al. [3] (e.g. to use "small random subsets" for the verifying parties, and to store the authentication tags in a "robust and distributed" manner) but our scheme also differs in many aspects. The crucial difference is that we do not require the authentication keys to be authenticated; this is what enables us to obtain security against a rushing adversary. Also, how the reconstruction actually works — and *why* it works — is very different. In our approach, we mainly exploit the expander property of the "verification graph" given by the randomly chosen set of neighbors for each P_i, i.e., the set of parties whose share P_i can verify.

For instance, in a first step, our reconstruction procedure checks if the number of incorrect Shamir shares is almost t (i.e., maximal) or whether there is a small *linear* gap. It does so by checking if there are $t+1$ parties that accept sufficiently more than half of the shares they verify. This works because by the random choice of the neighbors, the local view of each honest party provides a good estimate of the global picture (except with small probability). e.g., if almost half of all the shares are incorrect, then for each honest party roughly half of his neighbors has an incorrect share.

If the outcome is that there is a (small but positive) linear gap between the number of incorrect shares and t then we can employ list-decoding to obtain a poly-size list of possible candidates for the secret. In order to find the right secret from the list we need to further inspect the "consistency graph", given by who accepts whom. Concretely, for every secret on the list (and the corresponding *error-corrected* list of shares) it is checked if there exist $t+1$ parties whose shares are deemed correct *and* who accept a party whose share is deemed incorrect. It is clear that this cannot happen if the secret in question is the right one, because no one of the $t+1$ honest parties would accept an incorrect share. And, vice versa, if the secret in question is incorrect then, because of the promised redundancy in the correct shares, there must be a certain number of correct shares that are deemed incorrect and, with the parameters suitably chosen, each honest party has one of them as neighbor (except with small probability) and so will accept that one.

If, on the other hand, the outcome of the initial check is that there are almost t incorrect shares, then the reconstruction procedure uses a very different approach to find the correct shares. Explaining the strategy in detail is beyond the scope of this high-level sketch, but the idea is to start with a set that consists of a single party and then recursively pull those parties into the set that are accepted by the current parties in the set. The hope is that when we start with an honest party then we keep mostly including other honest parties. Of course, we cannot expect to end up with honest parties only, because an honest party may pull a party into the set that has announced a correct share (and thus looks honest) but which is actually dishonest and accepts incorrect shares of other dishonest parties, which then get pulled into the set as well. But note, given that we are in the case of almost t incorrect shares, there are not many such "passively dishonest" parties, and we can indeed control this and show that if we stop at the right moment, the set will consist of mainly honest parties and only a few dishonest parties with incorrect shares. By further inspection of the "consistency graph", trying to identify the missing honest parties and removing dishonest ones, we are eventually able to obtain a set of parties that consists of all honest parties, plus where the number of "actively dishonest" parties is at most half the number of "passively dishonest" parties (except with small probability), so that we have sufficient redundancy in the shares to recover the secret (using Reed-Solomon error correction).

By choosing the out-degree of the "verification graph" (i.e., the number of parties each party can verify) appropriately, so that the above informal reasoning can be rigorously proven (to a large extent by exploiting the randomness of each party's neighborhood and applying the Chernoff-Hoeffding bound), we obtain the claimed overhead $O(\kappa n^{\varepsilon})$ for an arbitrary choice of $\varepsilon > 0$.

As a simple variation of our approach, by choosing the out-degree of the "verification graph" to be polylog(n), we obtain the same (asymptotic) $\widetilde{O}(\kappa)$ overhead as Bishop et al. [6] and still have security against a rushing adversary, but then the reconstruction becomes (slightly) superpolynomial (because the size of the list produced by the list-decoder becomes superpolynomial).

2 Preliminaries

2.1 Graph Notation

Let $G = (V, E)$ be a graph with vertex set V and edge set E. By convention, $(v, w) \in E$ is the edge directed from v to w. For $S \subseteq V$, we let $G|_S$ be the restriction of G to S, i.e., $G|_S = (S, E|_S)$ with $E|_S = \{(u, v) \in E : u, v \in S\}$. Furthermore, we introduce the following notation.

For $v \in V$, we set

$$N^{\mathsf{out}}(v) = \{w \in V : (v, w) \in E\} \quad \text{and} \quad N^{\mathsf{in}}(v) = \{w \in V : (w, v) \in E\}.$$

We often write E_v as a short hand for $N^{\mathsf{out}}(v)$, and call it the *neighborhood* of v. For $S \subseteq V$, we set

$$N_S^{\mathsf{out}}(v) = N^{\mathsf{out}}(v) \cap S \quad \text{and} \quad N_S^{\mathsf{in}}(v) = N^{\mathsf{out}}(v) \cap S\,.$$

We extend this notation to a *labeled* graph, i.e., when G comes with a function $L : E \to \{\mathsf{good}, \mathsf{bad}\}$ that labels each edge. Namely, for $v \in V$ we set

$$N^{\mathsf{out}}(v, \mathsf{good}) = \{w \in N^{\mathsf{out}}(v) : L(v, w) = \mathsf{good}\},$$
$$N^{\mathsf{in}}(v, \mathsf{good}) = \{w \in N^{\mathsf{in}}(v) : L(w, v) = \mathsf{good}\},$$

and similarly $N^{\mathsf{out}}(v, \mathsf{bad})$ and $N^{\mathsf{in}}(v, \mathsf{bad})$. Also, $N^{\mathsf{out}}_S(v, \mathsf{good})$, $N^{\mathsf{in}}_S(v, \mathsf{good})$, $N^{\mathsf{out}}_S(v, \mathsf{bad})$ and $N^{\mathsf{in}}_S(v, \mathsf{bad})$ are defined accordingly for $S \subseteq V$. Finally, we set

$$n^{\mathsf{out}}(v) = |N^{\mathsf{out}}(v)| \quad \text{and} \quad n^{\mathsf{in}}_S(v, \mathsf{bad}) = |N^{\mathsf{in}}_S(v, \mathsf{bad})|$$

and similarly for all other variations.

We refer to a graph $G = (V, E)$ as a *randomized* graph if the edges E are chosen in a randomized manner, i.e., if E is actually a random variable. We are particularly interested in randomized graphs where (some or all of) the E_v's are uniformly random and independent subsets $E_v \subset V \setminus \{v\}$ of a given size d. For easier terminology, we refer to such neighborhoods E_v as being *random and independent.*

2.2 Chernoff Bound

Like for [3], much of our analysis relies on the Chernoff-Hoeffding bound, and its variation to "sampling without replacement". Here and throughout, $[n]$ is a short hand for $\{1, 2, \ldots, n\}$.

Definition 1 (Negative Correlation [1]). *Let $X_1, \ldots, X_n$ be binary random variables. We say that they are negatively correlated if for all $I \subset [n]$:*

$$\Pr[X_i = 1 \, \forall \, i \in I] \le \prod_{i \in I} \Pr[X_i = 1], \quad \Pr[X_i = 0 \, \forall \, i \in I] \le \prod_{i \in I} \Pr[X_i = 0].$$

Theorem 1 (Chernoff-Hoeffding Bound). *Let $X_1, \ldots, X_n$ be random variables that are independent and in the range $0 \le X_i \le 1$, or binary and negatively correlated, and let $u = E\big[\sum_{i=1}^n X_i\big]$. Then, for any $0 < \delta < 1$:*

$$\Pr\left[\sum_{i=1}^n X_i \le (1-\delta)u\right] \le e^{-\delta^2 u/2} \quad \text{and} \quad \Pr[\left[\sum_{i=1}^n X_i \ge (1+\delta)u\right] \le e^{-\delta^2 u/3}.$$

As immediate consequence, we obtain the following two bounds. The first follows from Chernoff-Hoeffding with independent random variables, and the latter from Chernoff-Hoeffding with negatively correlated random variables. We refer to [1] for more details, e.g., for showing that the random variables $X_j = 1$ if $j \in E_v$ and 0 otherwise are negatively correlated for E_v as in Corollary 1

Corollary 1. *Let G be a randomized graph with the property that, for some fixed $v \in V$, the neighborhood E_v is a random subset of $V \setminus \{v\}$ of size d. Then, for any fixed subset $T \subset V$, we have*

$$\Pr\Big[n^{\mathsf{out}}_T(v) \ge (1+\epsilon)\tfrac{|T|d}{|V|}\Big] \le e^{-\epsilon^2 \frac{|T|d}{2|V|}} \text{ and } \Pr\Big[n^{\mathsf{out}}_T(v) \le (1-\epsilon)\tfrac{|T|d}{|V|}\Big] \le e^{-\epsilon^2 \frac{|T|d}{3|V|}}.$$

Corollary 2. *Let G be a randomized graph with the property that, for some fixed $T \subset V$, the neighborhoods E_v for $v \in T$ are random and independent of size d (in the sense as explained in Sect. 2.1). Then, for any $v \notin T$, we have*

$$\Pr\Big[n^{\mathsf{in}}_T(v) \geq (1+\epsilon)\tfrac{|T|d}{|V|}\Big] \leq e^{-\epsilon^2 \frac{|T|d}{2|V|}} \text{ and } \Pr\Big[n^{\mathsf{in}}_T(v) \leq (1-\epsilon)\tfrac{|T|d}{|V|}\Big] \leq e^{-\epsilon^2 \frac{|T|d}{3|V|}}.$$

We emphasize than when we apply these corollaries, we consider a graph G where *a priori* all E_v's are random and independent. However, our reasoning typically is applied *a posteriori*, given some additional information on G, like the adversaries view. As such, we have to be careful each time that the considered neighborhoods are still random *conditioned* on this additional information on G.

2.3 Robust Secret Sharing

A robust secret sharing scheme consists of two interactive protocols: the *sharing* protocol **Share** and the *reconstruction* protocol **Rec**. The sharing protocol is executed by a *dealer D* and n parties $1, \ldots, n$: the dealer takes as input a message **msg**, and each party $i \in \{1, \ldots, n\}$ obtains as output a so-called *share*. Typically, these shares are locally computed by the dealer and then individually sent to the parties. The reconstruction protocol is executed by a *receiver R* and the n parties: each party is supposed to use its share as input, and the goal is that R obtains **msg** as output. Here, the protocol is typically so that the parties send their shares to R (possibly "piece-wise", distributed over multiple rounds of communication), and R then performs some local computation.

Such a robust secret sharing should be "secure" in the presence of an adversary that can adaptively corrupt up to t of the parties $1, \ldots, n$. Once a party is corrupted, the adversary is able to see the share of this party, and he can choose the next corruption based on the shares of the currently corrupt parties. Furthermore, in the reconstruction protocol, the corrupt parties can arbitrarily deviated from the protocol and, e.g., use incorrect shares. The following captures the security of a robust secret sharing in the list of such an adversary.

Definition 2 (Robust Secret Sharing). *Such a pair* (**Share**, **Rec**) *of protocols is called a* (t, δ)-robust secret sharing scheme *if it satisfies the following properties hold for any distribution of* **msg** *(from a given domain).*

– Privacy: *Before* **Rec** *is started, the adversary has no more information on the shared secret* **msg** *than he had before the execution of* **Share**.
– Robust reconstructability: *At the end of* **Rec**, *the reconstructor R output* $\mathbf{msg}' = \mathbf{msg}$ *except with probability at most δ.*

2.4 On the Power of Rushing

As defined above, there is still some ambiguity in the security notion, given that we have not specified yet the adversary's (dis)ability of *eavesdropping* on the communication of the sharing and the reconstruction protocols. Obviously, for the *privacy* condition to make sense, it has to be assumed that during the execution of the sharing protocol the adversary has no access to the communication between D and the uncorrupt parties. On the other hand, it is commonly assumed that the adversary *has* access to the communication between the parties and R during the execution of the reconstruction protocol. This in particular means that the adversary can choose the incorrect shares, which the corrupt parties send to R, *depending* on the honest parties' (correct) shares.[6] Such an adversary is referred to as a *rushing* adversary. In contrast, if it is assumed that the adversary has to choose the incorrect shares depending on the shares of the corrupt parties only, one speaks of a *non-rushing* adversary. Thus, Definition 2 above comes in two flavors, depending on whether one considers a rushing or a non-rushing adversary. Obviously, considering a rushing adversary gives rise to a *stronger* notion of security. In order to deal with a rushing adversary, it is useful to reveal the shares "in one go" but piece-by-piece, so as to limit the dependence between incorrect and correct shares.

In this work, in order to be in-par with [13] and [6], we require security against a *rushing* adversary. On the other hand, the scheme by Bishop, Pastro, Rajaraman and Wichs [3] offers security against a non-rushing adversary only — and, as explained in the introduction, there are inherent reasons why it cannot handle a rushing adversary.

3 Overview of Scheme

Our Approach. As in [3], the sharing phase is set up in such a manner that every party i can verify (by means of a MAC) the Shamir shares of the parties j of a *randomly sampled* subset $E_i \subset [n]\backslash\{i\}$ of parties. However, in contrast to [3], in our scheme *only* the Shamir share is authenticated; in particular, we do not authenticate the authentication keys (nor the set E_j).

If the reconstruction is then set up in such a way that first the Shamir shares are announced, and only afterwards the authentication keys, it is ensured (even in the presence of a rushing adversary) that the consistency graph, which labels an edge from i to $j \in E_i$ as "good" if and only if i correctly verifies j's Shamir share, satisfies the following:

- All edges from honest parties to passive or honest parties are labeled good.
- All edges form honest parties to active parties are labeled bad.

Here, and in the remainder, a corrupt party is called *active* if it announced an incorrect Shamir share in the reconstruction, and it is called *passive* if it announced a correct Shamir share, but may still lie about other parts, like the

[6] This may look artificial at first glance, but one motivation comes from the fact that in some applications one might want to do the reconstruction *among the parties*, where then each party individually plays the role of R (and performs the local computation that the reconstruction protocol prescribes). In this case, every party sends his share to every other party, and thus the corrupt parties unavoidably get to see the shares of the honest parties and can decide on the incorrect shares depending on those.

authentication keys. This is a significant difference to [3], where it is also ensured that corrupt parties that lie about their authentication keys are recognized as well.

Divide the Discussion. Similarly to [3], the reconstructor first tries to distinguish between the (non-exclusive) cases $|P| \leq \epsilon n$ and $|P| \geq \frac{\epsilon n}{4}$, where P denotes the set of passive parties (as defined above). In order to do so, we observe that the honest party is expected to have $(|P| + |H|)\frac{d}{n}$ good outgoing edges, where H is the set of honest parties. Thus, if there exist $t+1$ parties with more than $(1+\epsilon)d/2$ good outgoing edges, we are likely to be in the case $|P| \geq \frac{\epsilon n}{4}$, and otherwise $|P| \leq \epsilon n$.

Based on this distinction, the reconstructor will then refer to either of the following two algorithms to recover the secret.

Code Based Algorithm. This algorithm is used to handle the case $|P| \geq \frac{\epsilon n}{4}$. Here, one can use the redundancy provided by the correct shares of the parties in P to do *list-decoding.* This works given that the Shamir sharing is done by means of a *folded* Reed-Solomon code. Since those are maximum distance separable (MDS) codes, the corresponding secret sharing scheme is still threshold; moreover, it enjoys the nice feature that we can apply list decoding to correct up to $t - \epsilon n/4$ corruptions for any small constant ϵ. Finding the right entry in the list can then be done by a further inspection of the consistency graph.

Graph Algorithm. This graph algorithm is used in case $|P| \leq \epsilon n$. The basic algorithm starts off with a particular party, and produces the correct secret (with high probability) if that party happens to be honest. Hence, applying this algorithm to all choices for that party and taking a majority enables to reconstruct the secret.

The algorithm consists of three steps.

- The first step is to find a big subset V that contains many honest parties and very small proportion of dishonest parties. We do so by starting off with $V = \{i\}$ for a particular party i (which we assume to be honest for the discussion) and recursively include all parties into V that are correctly verified by the parties in V. A simple argument shows that in each step, we expect to include $d/2$ honest parties and at most ϵd passive parties.
  By the expander property of the consistency graph restricted to the honest parties ensures that the set V will soon be expanded to a set containing many honest parties. On the other hand, we can limit the "damage" done by passive parties by only including parties that have at most $\frac{d}{2}(1+3\epsilon)$ good outgoing edges. Given that there are only very few passive parties and that we limit the number of active parties they can pull into V, we can show that V can be expanded to a set of size $\Omega(\epsilon n)$ such that at most a $O(\sqrt{\epsilon})$-fraction of the parties in V are corrupt.
- The next step is to rely on the authentication of parties in V to include all honest parties and few dishonest parties where the majority is passive. We

first expand V to contain *all* honest parties and at most $O(\sqrt{\epsilon}n)$ dishonest parties. Then, we remove all active parties from V (but possibly also some honest and passive ones). Let W be the set of all parties removed from V. We show that the resulting set V still contains almost all honest parties and few passive parties, and W is of size $O(\sqrt{\epsilon}n)$ and contains the rest of honest parties.

A subtle issue now is that the sets V and W above depend on the E_i's of the honest parties; this then means that now given these two sets, we cannot rely anymore on the randomness of the E_i's. In order to circumvent this, we resort to another layer of authentication that is done in parallel to the former, with fresh E_i''s, and by means of this, we can then eventually identify a subset $S \subseteq [n]$ that contains all $t+1$ honest parties, as well as some number h of passive parties and at most $\frac{h}{2}$ active parties (with high probability).

– Given that we have "sufficiently more redundancy that errors", the secret can now be recovered by means of Reed-Solomon error correction (noting that a codeword of a folded Reed-Solomon code is also a codeword of some classic Reed-Solomon code).

4 Building Blocks

We present three building blocks here which are used in our construction.

4.1 Shamir Secret Sharing with List Decoding

It is well known that the share-vector in Shamir's secret sharing scheme is nothing else than a codeword of a Reed-Solomon code. Thus, Reed-Solomon decoding techniques can be applied when we are in the regime of unique decoding. Furthermore, if we use a *folded* Reed-Solomon code, then we still get a Shamir-like threshold secret sharing scheme, but in addition we can employ *list-decoding* when we are in a regime were decoding is not unique anymore.

In summary, we have the following (see Appendix A.1 and [12] for the details).

Proposition 1. *Let γ be any small constant. There exists $2t+1$-party threshold secret sharing scheme over $\mathbb{F}_q$ with $q = t^{O(\frac{1}{\gamma^2})}$ such that:*

– *This scheme enjoys t-privacy and $t+1$-reconstruction.*
– *There is a randomized list decoding algorithm that corrects up to $t-\gamma(2t+1)$ incorrect shares and outputs a list of candidates containing the correct secret with probability at least $1-2^{-\Omega(t)}$. The list size is $\lambda = (\frac{1}{\gamma})^{\frac{1}{\gamma}\log\frac{1}{\gamma}}$ and this list decoding algorithm runs in time poly(t,λ)*
– *There exists an efficient decoding algorithm that reconstructs the secret from any $t+1+2a$ shares of which at most a are incorrect.*

4.2 MAC Construction

Similarly to [3], our construction requires a message authentication code (MAC) with some additional features. There is some overlap with the features needed in [3], but also some differences.

Definition 3. *A message authentication code (MAC) for a finite message space $\mathcal{M}$ consists of a family of functions $\{MAC_{key} : \mathcal{M} \times \mathcal{R} \to \mathcal{T}\}_{key \in \mathcal{K}}$. This MAC is said to be (ℓ, ϵ)-secure if the following three conditions hold.*

1. Authentication security: *For all $(m, r) \neq (m', r') \in \mathcal{M} \times \mathcal{R}$ and all $\sigma, \sigma' \in \mathcal{T}$,*
$$\Pr_{key \leftarrow \mathcal{K}}[MAC_{key}(m', r') = \sigma' | MAC_{key}(m, r) = \sigma] \leq \epsilon.$$
2. Privacy over Randomness: *For all $m \in \mathcal{M}$ and $key_1, \ldots, key_\ell \in \mathcal{K}$, the distribution of ℓ values $\sigma_i = MAC_{key_i}(m, r)$ is independent of m over the choice of random string $r \in \mathcal{R}$, i.e.,*
$$\Pr_{r \leftarrow \mathcal{R}}[(\sigma_1, \ldots, \sigma_\ell) = \mathbf{c} | m] = \Pr_{r \leftarrow \mathcal{R}}[(\sigma_1, \ldots, \sigma_\ell) = \mathbf{c}]$$
for any $\mathbf{c} \in \mathcal{T}^\ell$.
3. Uniformity: *For all $(m, r) \in \mathcal{M} \times \mathcal{R}$, the distribution of $\sigma = MAC_{key}(m, r)$ is uniform at random over the random element $key \in \mathcal{K}$.*

The above privacy condition will be necessary for the privacy of the robust secret sharing scheme, since the Shamir shares will be authenticated by means of such a MAC but the corresponding tags will not be hidden from the adversary.

The uniformity property will be crucial in a *lazy sampling* argument, were we need to "simulate" certain tags *before* we know which messages they actually authenticate. With the uniformity property, this can obviously be done by picking σ uniformly at random from $\mathcal{T}$. When m and r become available, we can then sample a uniformly random key *key* subject to $MAC_{key}(m, r) = \sigma$. This has the same distribution as when *key* is chosen uniformly at random and σ is computed as $\sigma = MAC_{key}(m, r)$.

The following variation of the standard polynomial-evaluation MAC construction meets all the requirements.

Theorem 2 (Polynomial Evaluation). *Let $\mathbb{F}$ be a finite field. Let $\mathcal{M} = \mathbb{F}^a$, $\mathcal{R} = \mathbb{F}^\ell$ and $\mathcal{T} = \mathbb{F}$ such that $\frac{a+\ell}{|\mathbb{F}|} \leq \epsilon$. Define the family of MAC functions $\{MAC_{(x,y)} : \mathbb{F}^a \times \mathbb{F}^\ell \to \mathbb{F}\}_{(x,y) \in \mathbb{F}^2}$ such that*
$$MAC_{(x,y)}(\mathbf{m}, \mathbf{r}) = \sum_{i=1}^{a} m_i x^{i+\ell} + \sum_{i=1}^{\ell} r_i x^i + y$$
for all $\mathbf{m} = (m_1, \ldots, m_a) \in \mathbb{F}^a$, $\mathbf{r} = (r_1, \ldots, r_\ell) \in \mathbb{F}^\ell$ and $(x, y) \in \mathbb{F}^2$. Then, this family of MAC functions is (ℓ, ϵ)-secure.

4.3 Robust Distributed Storage

A *robust distributed storage scheme* is a robust secret sharing scheme as in Definition 2 but without the privacy requirement. This was used in [3] in order to ensure that dishonest parties cannot lie about the tags that authenticate their shares (so as to, say, provoke disagreement among honest parties about the correctness of the share).[7] Also our construction uses a robust distributed storage scheme for storing the authentication tags. It does not play such a crucial role here as in [3], but it makes certain things simpler.

A robust distributed storage scheme can easily be obtained by encoding the message by means of a list-decodable code and distribute the components of the code word among the parties, and to give each party additionally a random key and the hash of the message under the party's key (using almost-universal hashing). In order to reconstruct, each party runs the list-decoding algorithm and uses his key and hash to find the correct message in the list. The exact parameters then follow from list-decoding parameters.

Therefore, each party i holds two components, the i-th share of list-decodable code, p_i, and a hash-key and the hash of the message, jointly referred to as q_i. While [3] did not consider a rushing adversary, it is easy to see that security of this robust distributed storage scheme against a rushing adversary can be obtained by having the parties reveal p_i and q_i in two different communication rounds (so that the adversary has to decide on an incorrect p_i *before* he knows the keys that the honest parties will use). Therefore, the following result from [3], which is obtained by using suitable parameters for the list decoding an hashing, is also applicable in rushing-adversary model.

Theorem 3 ([3]). *For any $n = 2t + 1$ and $u \geq \log n$, there exists a robust distributed storage with messages of length $m = \Omega(nu)$ and shares of length $O(u)$ that can recover the message with probability $1 - O(\frac{n^2}{2^u})$ up to t corruptions.*

In our application, the length of shares is $O(u) = O(n^{\sqrt{\epsilon}})$ and the length of messages is $m = \Omega(nu)$. If we apply this theorem directly, the size of $\mathbb{F}_q$ in their construction is 2^u which is unnecessarily big. Instead, We pick $\mathbb{F}_q$ with $q = \Omega(n^5)$. Then, we obtain following.

Theorem 4. *For any $n = 2t + 1$ and $u = O(n^{\sqrt{\epsilon}})$ for small constant ϵ, there exists a robust distributed storage against rushing adversary with messages of length $m = \Omega(nu)$, shares of length $O(u)$ that can recover the message with probability $1 - O(\frac{1}{n^2})$ up to t corruptions. In this robust distributed storage, party i holds two components, p_i and q_i, revealed in two rounds.*

[7] On the other hand, this is why the additional privacy property of the MAC is necessary, since the robust distributed storage does not offer privacy, and thus the tags are (potentially) known.

5 The Robust Secret Sharing Scheme

5.1 Sharing Protocol

Let t be an arbitrary positive integer and $n = 2t + 1$. Let $\epsilon > 0$ be a small constant and $d = n^{\sqrt{\epsilon}}$. Let $(\mathbf{Sh}, \mathbf{Lis})$ be the sharing and list decoding algorithm of the threshold secret sharing scheme in Proposition 1 with $\gamma = \frac{\epsilon}{4}$. Also, we use the MAC construction from Theorem 2 with $\ell = 4d$, and with the remaining parameters to be determined later (but chosen so that a share produced by **Sh** can be authenticated).

On input $\mathbf{msg} \in \mathbb{F}_q$, our sharing procedure $\mathbf{Share}(\mathbf{msg})$ proceeds as follows.

1. Let $(s_1, \ldots, s_n) \leftarrow \mathbf{Sh}(\mathbf{msg})$ to be a non-robust secret sharing of $\mathbf{msg}$.
2. For each $i \in [n]$, sample MAC randomness $r_i \leftarrow \mathcal{T}^{4d}$ and do the following operation twice.
   (a) For each $i \in [n]$, choose a random set $E_i \subseteq [n] \backslash \{i\}$ of size d. If there exists $j \in [n]$ with in-degree more than $2d$, do it again.[8]
   (b) For each $i \in [n]$, sample the d random MAC keys $key_{i,j} \in \mathcal{T}^2$ for $j \in E_i$. Define $\mathcal{K}_i = \{key_{i,j} : j \in E_i\}$ to be the collection of these d random keys.
   (c) Compute the MAC
   $$\sigma_{i \to j} = MAC_{key_{i,j}}(s_j, r_j) \in \mathcal{T} \quad \forall j \in E_i.$$
   Let E_i, $key_{i,j}$ and $\sigma_{i\to j}$ be the output of the first round and E'_i, $key'_{i,j}$ and $\sigma'_{i\to j}$ be the output of the second round.
3. For each $i \in [n]$, define $\mathbf{tag}_i = \{\sigma_{i\to j} : j \in E_i\} \in \mathcal{T}^d$ and $\mathbf{tag}'_i = \{\sigma'_{i\to j} : j \in E'_i\} \in \mathcal{T}^d$. Let $\mathbf{tag} = (\mathbf{tag}_1, \mathbf{tag}'_1, \ldots, \mathbf{tag}_n, \mathbf{tag}'_n) \in \mathcal{T}^{2nd}$. Use the robust distributed storage scheme to store **tag**. Party i holds p_i and q_i.
4. For $i \in [n]$, define $\mathbf{s}_i = (s_i, E_i, E'_i, \mathcal{K}_i, \mathcal{K}'_i, r_i, p_i, q_i)$ to be the share of party i. Output $(\mathbf{s}_1, \ldots, \mathbf{s}_n)$.

5.2 Reconstruction Protocol

1. The first round: Every party i sends (s_i, r_i, p_i) to the reconstructor R.
2. The second round: Every party i sends $(q_i, E_i, \mathcal{K}_i)$ to the reconstructor R.
3. The third round: Every party i sends $(E'_i, \mathcal{K}'_i)$ to the reconstructor R.

Remark 1. We emphasize that since the keys for the authentication tags are announced *after* the Shamir shares, it is ensured that the MAC does its job also in the case of a rushing adversary. Furthermore, it will be crucial that also the E_i's are revealed in the second round only, so as to ensure that once the (correct and incorrect) Shamir shares are "one the table", the E_i's for the honest parties are still random and independent. Similarly for the E'_i's in the third round.

[8] This is for the privacy purpose.

On receiving the shares of n parties, our reconstruction scheme $\mathbf{Rec}(\mathbf{s_1}, \ldots, \mathbf{s_n})$ goes as follows:

1. R collects the share of robust distributed storage: $(p_i, q_i)_{i \in [n]}$.
2. Reconstruct the $\mathbf{tag} = (\mathbf{tag}_1, \mathbf{tag}'_1, \ldots, \mathbf{tag}_n, \mathbf{tag}'_n)$ and parse $\mathbf{tag}_i = \{\sigma_{i \to j} : j \in E_i\}$ and $\mathbf{tag}'_i = \{\sigma'_{i \to j} : j \in E'_i\}$.
3. Define two graphes $G = ([n], E)$ and $G' = ([n], E')$ such that $E = \{(i,j) : i \in [n], j \in E_i\}$ and $E' = \{(i,j) : i \in [n], j \in E'_i\}$.
4. Assign a label $L(e) \in \{\texttt{good}, \texttt{bad}\}$ to each edge $e = (i,j) \in E$ such that $L(e) = \texttt{good}$ if
$$\sigma_{i \to j} = MAC_{key_{i,j}}(s_j, r_j)$$
and $\texttt{bad}$ otherwise. Do the same thing to the edge $e \in E'$.
5. Run the Check(G, L, ϵ),
   (a) If the output is Yes, Let $\mathbf{s} = (s_1, \ldots, s_n)$ and $\mathbf{c} = \text{List}(G, \mathbf{s}, \epsilon/4)$.
   (b) Otherwise, for each $i \in [n]$, let $\mathbf{c}_i = \text{Graph}(G, G', \epsilon, i)$. If there exists a codeword $\mathbf{c}_i$ repeating at least $t+1$ times, let $\mathbf{c} = \mathbf{c}_i$. Otherwise, $\mathbf{c} = \perp$.
6. Output $\mathbf{c}$.

Note that step 5 in the reconstruction refers to subroutines: Check, List and Graph, which we specify only later.

5.3 The Privacy Property

Theorem 5. *The scheme* (**Share**, **Rec**) *satisfies perfect privacy.*

Proof. Let $C \subset [n]$ be of size t. We let $\mathbf{msg} \in \mathcal{M}$ be arbitrarily distributed and consider $\mathbf{Share}(\mathbf{msg}) = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$. Our goal is to show that the distribution of $(\mathbf{s}_i)_{i \in C}$ is independent of $\mathbf{msg}$. Note that $\mathbf{s}_i = (s_i, r_i, p_i, q_i, E_i, E'_i, \mathcal{K}_i, \mathcal{K}'_i)$. Since our threshold secret sharing scheme has t-privacy, the collection of shares s_i for $i \in C$ is independent of $\mathbf{msg}$. By construction, $r_i, E_i, E'_i, \mathcal{K}_i, \mathcal{K}'_i$ are independently chosen as well. Since the (p_i, q_i)'s are computed from $\mathbf{tag}$ (using independent randomness), it suffices to show that $\mathbf{tag}$ reveals no information on $\mathbf{msg}$. Recall that $\mathbf{tag}$ is used to verify the integrity of (s_j, r_j) for all j. For any $j \notin C$, there are at most $4d$ tags $\sigma_{i \to j} = MAC_{key_{i,j}}(s_j, r_j)$ corresponding to the total degree of vertex i in two graphs. By the "privacy over randomness" of the MAC, $\mathbf{tag}$ is independent of these shares s_j, and hence the privacy of $\mathbf{msg}$ is ensured.

6 The Robustness Property

6.1 Preliminary Observations

From the security properties of the robust distributed storage and of the MAC, we immediately obtain the following results.

Lemma 1. **tag** *is correctly reconstructed expect with probability* $\epsilon_{tag} = O(1/n^2)$.

This is not negligible; this will be dealt with later by parallel repetition.

Here and in the remainder of the analysis of the robustness property, H denotes the set of honest parties, and C denotes the set of dishonest parties. Furthermore, we decompose C into $C = A \cup P$, where A is the set of dishonest parties that announced an *incorrect* (s_i, r_i) in the first communication round and P denotes the (complementary) set of dishonest parties that announced a *correct* (s_i, r_i). The parties in A are called *active* parties, and the parties in P are referred to as *passive* parties.

Proposition 2. *If* **tag** *was correctly reconstructed then the labelling of the graph G satisfies the following, except with probability $\epsilon_{mac} \leq 4(t+4d)dt/|\mathcal{T}|$. For every $h \in H$ and for every edge $e = (h, j) \in E_h = N^{\mathsf{out}}(h)$, it holds that*

$$L(e) = \begin{cases} \texttt{bad} & \text{if } j \in A \\ \texttt{good} & \text{if } j \in H \cup P \end{cases} .$$

I.e., all the edges from honest parties to active parties are labeled bad and all edges from honest parties to honest parties or passive parties are labeled good. The same holds for G'.

Proof. By the definition of passive parties and the construction of MAC, all edges from honest parties to passive parties are labeled good. It remains to prove the first half of the claim. Let us fix an active party i. According to the definition of active party, he claims $(s'_j, r'_j) \neq (s_j, r_j) \in \mathcal{T}^a \times \mathcal{T}^{4d}$. For any honest party i with $j \in E_i$ or $j \in E'_i$, the edge (i, j) is label good if $\sigma_{i \to j} = MAC_{key_{i,j}}(s'_j, r'_j)$. This event happens with

$$\Pr_{key_{i,j} \leftarrow \mathcal{T}^2}[\sigma_{i \to j} = MAC_{key_{i,j}}(s'_j, r'_j)] \leq \frac{a + 4d}{|\mathcal{T}|}$$

due to the authentication of MAC. Note that each vertex has at most $4d$ incoming edges. Taking a union bound over all these edges and the active parties, the desired result follows.

6.2 On the Randomness of the Graph

Much of our analysis relies on the randomness of the graph G (and G'). A subtle point is that even though *a priori* all of the E_i are chosen to be random and independent (in the sense as explained in Sect. 2.1), we have to be careful about the *a posteriori* randomness of the E_i *given the adversary's view.* In particular, since the adversary can corrupt parties *adaptively* (i.e., depending on what he has seen so far), if we consider a particular dishonest party j then the mere fact that this party is dishonest may affect the *a posteriori* distribution of G.

However, and this is what will be crucial for us is that, conditioned on the adversary's view, the E_i's *of the honest parties* i remain random and independent.

Proposition 3. *Up to right before the second communication round of the reconstruction protocol, conditioned on the adversary's view of the protocol, the graph G is such that the E_i for $i \in H$ are random and independent.*
The corresponding holds for G' up to right before the third communication round.

Proof. The claim follows from a straightforward lazy-sampling argument. It follows by inspection of the protocol that one can delay the random choice of each E_i (and E_i') to the point where party i gets corrupted, or is announced in the corresponding round in the reconstruction protocol. The only subtle issue is that, at first glance it seems that the computation of the tags $\sigma_{i \to j} = MAC_{key_{i,j}}(s_j, r_j)$ for $j \in E_i$ requires knowledge of E_i. However, by the uniformity property of MAC, these tags can instead be "computed" by sampling d tags uniformly at random from $\mathcal{T}$, and once party i gets corrupted and E_i is sampled, one can choose the keys $key_{i,j}$ appropriately for $j \in E_i$.

Remark 2. In the sequel, when making probabilistic statements, they should be understood as being *conditioned* on an arbitrary but fixed choice of the adversary's view. This in particular means that H, P and A are *fixed* sets then (since they are determined by the view of the adversary), and we can quantify over, say, all honest parties. The randomness in the statements then stems from the randomness of the E_i's (and E_i''s) of the honest parties $i \in H$, as guaranteed by Proposition 3 above.

Remark 3. The remaining analysis below is done under the implicit assumption that **tag** is correctly reconstructed and that the labelling of G and of G' is as specified in Proposition 2. We will incorporate the respective error probabilities then in the end.

6.3 The Check Subroutine

Roughly speaking, the following subroutine allows the reconstructor to find out if $|P|$, the number of passive parties, is linear in n or not.

Check(G, L, ϵ)

- Input: $G = ([n], E, L)$ and ϵ.
- If $|\{i \in [n] : n^{\mathsf{out}}(i, \mathsf{good}) \geq \frac{d}{2}(1+\epsilon)\}| \geq t+1$, then output "Yes".
- Otherwise, output "No".

Theorem 6. *Except with probability $\epsilon_{check} \leq 2^{-\Omega(\epsilon d)}$,* Check*$(G, L, \epsilon)$ outputs "Yes" if $|P| \geq \epsilon n$ and "No" if $|P| \leq \epsilon n/4$ (and either of the two otherwise).*

Proof. We only analyze the case that $|P| \leq \frac{\epsilon n}{4}$. The same kind of reasoning can be applied to the case $|P| \geq \epsilon n$. By Proposition 3 (and Remark 2), we know that for given P and H, the E_i's for $i \in H$ are random and independent. It follows that $n^{\mathsf{out}}(i, \mathsf{good}) = n^{\mathsf{out}}_{P \cup H}(i)$ is expected to be $\frac{|P|+|H|}{n} \leq \frac{d}{2}(1+\frac{\epsilon}{2})$, and thus, by Corollary 1,

$$\Pr\big[n^{\mathsf{out}}(i, \mathsf{good}) \geq \tfrac{d}{2}(1+\epsilon)\big] \leq 2^{-\Omega(\epsilon d)}.$$

Taking a union bound over all honest parties, we have

$$\Pr\big[\exists\, i \in H : n^{\mathsf{out}}(i, \mathsf{good}) \geq \tfrac{d}{2}(1+\epsilon)\big] \leq (t+1)2^{-\Omega(\epsilon d)} = 2^{-\Omega(\epsilon d)}.$$

Thus, except with probability $(t+1)2^{-\Omega(\epsilon d)}$, $n^{\mathsf{out}}(i, \mathsf{good}) \geq \frac{d}{2}(1+\epsilon)$ can only hold for dishonest parties i, and in this case $\mathrm{Check}(G, L, \epsilon)$ outputs "No". The desired result follows.

Remark 4. The subroutine $\mathrm{Check}(G, L, \epsilon)$ allows us to find out if $|P| \geq \frac{\epsilon n}{4}$ or $|P| \leq \epsilon n$. If $\mathrm{Check}(G, L, \epsilon)$ tells us that $|P| \geq \frac{\epsilon n}{4}$ (by outputting "Yes") then we use the redundancy provided by the shares of the parties in P to recover the secret by means of the code based algorithm $\mathrm{List}(G, \mathbf{s}, \epsilon/4)$. If $\mathrm{Check}(G, L, \epsilon)$ tells us that $|P| \leq \epsilon n$ (by outputting "No") then we run the graph algorithm $\mathrm{Graph}(G, G', \epsilon, v)$ for every choice of party v. For honest v, it is ensure to output the correct secret (with high probability), and so we can do a majority decision. We leave the description and the analysis of the code based algorithm and the graph algorithm to the respective next two sections.

6.4 Code Based Algorithm

Recall that $\gamma = \frac{\epsilon}{4}$. H is the set of honest parties, P is the set of passive parties and A is the set of active parties. In this section, we present an algorithm $\mathrm{List}(G, \mathbf{s}, \gamma)$ based on the list decoding algorithm of secret sharing scheme in Proposition 1 up to $t - \gamma n$ errors.

Code Based Algorithm, List$(G, \mathbf{s}, \gamma)$

- Input $G = ([n], E, L), \mathbf{s}, \gamma$.
- Run the list decoding algorithm on $\mathbf{s}$ to correct up to $t - \gamma n$ errors and output the list of candidates $(\mathbf{c}_1, \ldots, \mathbf{c}_\ell)$.
- Let S_i (T_i) be the set of parties whose shares agree (do not agree) with $\mathbf{c}_i$.
- For $1 \leq i \leq \ell$, run $\mathrm{Cand}(G, S_i, T_i)$. If the output is "succeed" then output $\mathbf{c}_i$.
- Output "fail".

We proceed to the analysis of the algorithm. The input of the list decoding algorithm is n shares of $\mathbf{s}$. Since $|P| \geq \gamma n$, there are at most $t - \gamma n = \frac{n}{2}(1 - 2\gamma)$ shares that are corrupted. Thus, by Proposition 1, the output of this list decoding algorithm will contain the correct codeword with probability at least $1 - 2^{-\Omega(n)}$. Moreover, the list size of this algorithm is at most $(\frac{1}{\gamma})^{O(1/\gamma \log 1/\gamma)}$. We may assume that this list only include codewords that are at most $t - \gamma n$ away from the n shares. Let $\mathbf{c}_1, \ldots, \mathbf{c}_\ell$ be the candidates on this list. To find the correct one, we resort to the labelled graph $G = ([n], E, L)$. Note that $\mathbf{c}_i$ for $1 \leq i \leq \ell$ are determined right after the first communication round. Thus, by Proposition 3, conditioned on $\mathbf{c}_1, \dots, \mathbf{c}_\ell$ (and the entire view of the adversary at this point), the E_i's for $i \in H$ are random and independent. For each candidate $\mathbf{c}_i$, we run the algorithm $\mathrm{Cand}(G, S_i, T_i, \gamma)$ to check if it is the correct codeword. For the correct codeword $\mathbf{c}_r$, we claim that this algorithm $\mathrm{Cand}(G, S_r, T_r, \gamma)$ will always output succeed. To see this, we notice that S_r must contain the set of $t+1$ honest parties H. Meanwhile, T_r is a subset of active parties A. By our assumption, there does not exist good edge from H to T_r. The desired results follows as these $t+1$ honest parties will remain in S_r after calling $\mathrm{Cand}(G, S_r, T_r, \gamma)$.

Verify the candidate, Cand(G, S, T)

- Input: $G = ([n], E, L)$, S, T.
- Remove all i from S if $n_T^{\mathsf{out}}(i, \mathsf{good}) \geq 1$.
- If $|S| \geq t+1$, output "succeed". Otherwise, output "fail".

It remains to show that with high probability this algorithm will output fail for all of the incorrect candidates.

Lemma 2. *If $\mathbf{c}_i$ is not a correct codeword then the algorithm* $\mathrm{Cand}(G, S_i, T_i)$ *will output fail except with probability at most* $2^{-\Omega(\gamma d)}$

Proof. By the guarantee of the list decoding algorithm, it is ensured that $|T_i| \leq t - \gamma n$ and thus $|S_i| \geq t + 1 + \gamma n$. Let W_i be the set of passive and honest parties in T_i. We observe that $|W_i| \geq \gamma n$; otherwise, $\mathbf{c}_i$ and the correct codeword would have $t+1$ shares in common, which would imply that they are the same codeword. Furthermore, for every honest party j in S_i, we have that

$$n_T^{\mathsf{out}}(j, \mathsf{good}) \geq n_{W_i}^{\mathsf{out}}(j, \mathsf{good}) = n_{W_i}^{\mathsf{out}}(j)$$

and Corollary 1 ensures that this is 0 with probability at most $2^{-\Omega(\gamma d)}$. Taking union bound over all honest parties in S_i, with probability at least $1 - 2^{-\Omega(\gamma d)}$, all the honest parties will be removed from S_i. The desired result follows as S_i has size at most t when all honest parties are removed.

Taking a union bound over all these ℓ candidates, we obtain the following.

Theorem 7. *Assume $|P| \geq \gamma n$. With probability ϵ_{code} at least $1 - 2^{-\Omega(\gamma d)}$, the algorithm List($G, \mathbf{s}, \gamma$) will output the correct codeword in time poly$\big(m, n, (\frac{1}{\epsilon})^{\widetilde{O}(\frac{1}{\epsilon})}\big)$.*

6.5 Graph Algorithm

In this section, we assume that our graph algorithm $\mathrm{Graph}(G, G', \epsilon, v)$ starts with an honest party v. Under this assumption and $|P| \leq \epsilon n$, we show that this algorithm will output the correct secret with high probability. Recall that the out-degree of vertices in G and G' is $d = n^{\sqrt{\epsilon}}$ for some small constant ϵ and that by assumption (justified by Proposition 2) the edges from honest parties to active parties are labeled bad, and the edges from honest parties to honest or passive parties are labeled good.

We also recall that, by definition, whether a corrupt party $i \in C$ is *passive* or *active*, i.e., in P or in A, only depends on s_i and r_i announced in the first communication round in the reconstruction protocol; a passive party may well lie about, say, his neighborhood E_i. Our reasoning only relies on the neighborhoods of the honest parties, which are random and independent conditioned on the adversary's view, as explained in Proposition 3 and Remark 2.

The graph algorithm Graph(G, G', ϵ, v) goes as follows. Note that ${n'}_W^{\mathsf{out}}$ refers to n_W^{out} but for the graph G' rather than G, and similarly for ${n'}_V^{\mathsf{in}}$.

The algorithm Graph(G, G', ϵ, v)

i. Input $G = ([n], E, L), G' = ([n], E', L'), d, \epsilon$ and $v \in [n]$.

ii. Expand set $V = \{v\}$ to include more honest parties:

$$\text{While } |V| \leq \frac{\epsilon t}{d} \text{ do } V := \text{Expan}(G, V, \epsilon).$$

iii. Include all honest parties into V:

$$V := V \cup \left\{v \notin V : n_V^{\mathsf{in}}(v, \mathsf{good}) \geq \tfrac{d|V|}{2n}\right\}.$$

iv. Remove all active parties from V (and maybe few honest parties as well):

$$W := \left\{v \in V : n_V^{\mathsf{in}}(v, \mathsf{bad}) \geq \tfrac{d}{4}\right\} \quad \text{and} \quad V := V \setminus W.$$

v. 1. Bound the degree of parties in V:

$$V := V \setminus \left\{v \in V : {n'}_W^{\mathsf{out}}(v) \geq \tfrac{d}{8}\right\}.$$

2. Include the honest parties from W (and perhaps few active parties):

$$V := V \cup \left\{v \in W : {n'}_V^{\mathsf{in}}(v, \mathsf{good}) \geq \tfrac{d}{4}\right\}.$$

3. Error correction: run the unique decoding algorithm algorithm on the shares of parties in V and output the result.

Remark 5. Each time we call Expan(G, V, ϵ), the size of V increases. After Step *ii*, we hope that the V has size $\Omega(\epsilon n)$ instead of barely bigger than $\frac{\epsilon t}{d}$. To achieve this, we require that the input V of the last loop to be of size $\Omega(\frac{\epsilon t}{d})$. It can be achieved as follows. Assume that $|V| > \frac{\epsilon t}{d}$. Take out each party in V with same probability such that the expectation of resulting V is less than $\frac{\epsilon t}{2d}$. Then, the proportion of honest party and dishonest party stays almost the same but the size of V is below the threshold $\frac{\epsilon t}{d}$ with probability at least $1 - 2^{-\Omega(\epsilon n/d)}$. It will not affect our randomness argument since we treat each party equally. We skip this step for simplicity. In our following analysis, we assume that V has size $\Omega(\epsilon n)$ at the end of Step *ii*.

Graph expansion algorithm Expan(G, V, ϵ)

- Input: $G = ([n], E, L)$, V and ϵ.
- Set $V' = \emptyset$. For each vertex $v \in V$ do the following:

$$\text{if } n^{\mathsf{out}}(v, \mathsf{good}) \le \frac{d}{2}(1 + 3\epsilon) \text{ then } V' := V' \cup N^{\mathsf{out}}(v, \mathsf{good}).$$

- Output $V' \cup V$.

Theorem 8. *Under the assumption in Remark 3, and assuming that the graph algorithm takes an honest party v as input and that $|P| \le \epsilon n$, the following holds. Except with failure probability $\epsilon_{graph} \le 2^{-\Omega(\epsilon^2 d)}$, the algorithm will output a correct secret. Moreover, it runs in time poly$(n, m, \frac{1}{\epsilon})$.*

We will prove this theorem in the following subsections.

6.6 Graph Expansion

We start by analyzing the expansion property of $G|_H$, the subgraph of G restricted to the set of honest parties H.

Lemma 3 (Expansion property of $G|_H$). *If $H' \subset H$ is so that $|H'| \le \frac{\epsilon |H|}{d}$ and the E_v's for $v \in H'$ are still random and independent in G when given H' and H, then*

$$n_H^{\mathsf{out}}(H') := \left| \bigcup_{v \in H'} N_H^{\mathsf{out}}(v) \right| \ge \frac{d}{2}(1 - 2\epsilon)|H'|$$

except with probability $2^{-\Omega(\epsilon^2 d |H'|)}$.

Informally, this ensures that, as long as H' is still reasonably small, including all the honest "neighbours" increases the set essentially by a factor $d/2$, as is to be expected: each party in H' is expected to pull in $d/2$ new honest parties. The formal proof is almost the same as the proof for a random expander graph except that we require a different parameter setting for our own purpose.

Proof. By assumption on the E_i's and by Corollary 1, the probability for any vertex $v \in H'$ to have $n_H^{\mathsf{out}}(v) < \frac{1}{2}(1-\epsilon)d$ is at most $\le e^{-\epsilon^2 d/4} = 2^{-\Omega(\epsilon^2 d)}$. Taking the union bound, this hold for all $v \in H'$. In the remainder of the proof, we may thus assume that $N_H^{\mathsf{out}}(v)$ consist of $d' := \frac{1}{2}(1 - \epsilon)d$ random outgoing edges. Let $N := |H|$, $N' := |H'|$, and let $v_1, \dots, v_{d'N'}$ denote the list of neighbours of all $v \in H'$, with repetition. To prove the conclusion, it suffices to bound the probability p_f that more than $\frac{d}{2}\epsilon N'$ of these $d'N'$ vertices are repeated.

The probability that a vertex v_i is equal to one of $v_1, \dots, v_{i-1}$ is at most

$$\frac{i}{N-1} \le \frac{d'N'}{N-1} = \frac{1}{2}(1-\epsilon)d \cdot \frac{\epsilon |H|}{d} \cdot \frac{1}{N-1} \le \frac{\epsilon}{2}\,.$$

Taking over all vertex sets of size $\frac{d}{2}\epsilon N'$ in these $d'N'$ neighbours, the union bound shows that p_f is at most

$$\begin{aligned}\binom{d'N'}{\frac{d}{2}\epsilon N'}\left(\frac{\epsilon}{2}\right)^{\frac{d}{2}\epsilon N'} &\le 2^{d'N'H(\frac{\epsilon}{1-\epsilon})+\frac{d}{2}\epsilon N'(\log\epsilon-1)}\\ &\le 2^{\frac{d(1-\epsilon)}{2}N'(-\frac{\epsilon}{1-\epsilon}\log\epsilon+\frac{\epsilon}{\ln 2}+O(\epsilon^2))+\frac{d}{2}N'\epsilon(\log\epsilon-1)}\\ &\le 2^{\frac{d}{2}N'\epsilon(\frac{1}{\ln 2}-1+O(\epsilon))}\\ &\le 2^{-\Omega(dN'\epsilon)}\end{aligned}$$

The first inequality is due to that $\binom{n}{k}\le 2^{nH(\frac{k}{n})}$ and the second due to

$$\begin{aligned}H\Big(\frac{\epsilon}{1-\epsilon}\Big) &= -\frac{\epsilon}{1-\epsilon}\log\frac{\epsilon}{1-\epsilon}-\frac{1-2\epsilon}{1-\epsilon}\log\frac{1-2\epsilon}{1-\epsilon}\\ &\le -\frac{\epsilon}{1-\epsilon}\log\epsilon-\log\Big(1-\frac{\epsilon}{1-\epsilon}\Big)\\ &= -\frac{\epsilon}{1-\epsilon}\log\epsilon+\frac{1}{\ln 2}\Big(\frac{\epsilon}{1-\epsilon}+O(\epsilon^2)\Big)\\ &\le -\frac{\epsilon}{1-\epsilon}\log\epsilon+\frac{\epsilon}{\ln 2}+O(\epsilon^2)\end{aligned}$$

for small ϵ and the Taylor series $\ln(1-x)=\sum_{i\ge 1}\frac{x^i}{i}$.

6.7 Analysis of Step ii

The following shows that after Step ii, at most an $O(\sqrt{\epsilon})$-fraction of the parties in V is dishonest. This is pretty much a consequence of Lemma 3.

Proposition 4. *At the end of Step ii, with probability at least* $1-2^{-\Omega(\epsilon^2 d)}$, *$V$ is a set of size $\Omega(\epsilon n)$ with $|H\cap V|\ge(1-O(\sqrt{\epsilon}))|V|$ and $|C\cap V|\le O(\sqrt{\epsilon})|V|$.*

Proof. Let V_i be the set V after Expan has been called i times, i.e., $V_0=\{v\}$, $V_1=\text{Expan}(G,V_0,\epsilon)$ etc., and let $H_0=\{v\}$ and $H_1=\text{Expan}(G,H_0,\epsilon)\cap H$, $H_2=\text{Expan}(G,H_2,\epsilon)\cap H$ etc. be the corresponding sets when we include only honest parties into the sets.

Using a similar lazy-sampling argument as for Proposition 3, it follows that conditioned on $H_0,H_1,\dots,H_i$, the E_j's for $j\in H_i\setminus H_{i-1}$ are random and independent for any i.[9] Therefore, we can apply Lemma 3 to $H_i'=H_i\setminus H_{i-1}$ to obtain that $|H_{i+1}|\ge|H_i'|\frac{d}{2}(1-2\epsilon)$. It follows[10] that $|H_i|\ge(\frac{d}{2}(1-2\epsilon))^i$ except with probability $2^{-\Omega(\epsilon^2 d)}$. According to Remark 5, our algorithm jumps out of Step *ii* when V is of size $\Omega(\epsilon n)$. We bound the number of rounds in this step. For $i=\frac{2}{\sqrt{\epsilon}}$, noting that $d=n^{\sqrt{\epsilon}}$, it thus follows that

$$|V_i|\ge|H_i|\ge\Big(\frac{d}{2}(1-2\epsilon)\Big)^i=\frac{n^2}{2^{\frac{2}{\sqrt{\epsilon}}}}(1-2\epsilon)^{\frac{2}{\sqrt{\epsilon}}}\ge\Omega(n^2).$$

[9] The crucial point here is that H_i is determined by the E_j's with $j\in H_{i-1}$ only.

[10] The size of H_{i-1} is negligible compared to H_i; indeed, $|H_i|=\Omega(d|H_{i-1}|)$ and thus $|H_i\setminus H_{i-1}|=(1-o(1))|H_i|$. So, we may ignore the difference between H_i and H_i'.

That means $\text{Expan}(G, V, \epsilon)$ is called $r \leq \frac{2}{\sqrt{\epsilon}}$ times assuming n is large enough.

On the other hand, we trivially have $|V_r| \leq (\frac{d}{2}(1+3\epsilon))^r$ by specification of Expan. Thus,

$$\begin{aligned}
&|V_r| - |H_r| \leq \Big(\frac{d}{2}(1+3\epsilon)\Big)^r - \Big(\frac{d}{2}(1-2\epsilon)\Big)^r \\
&= \frac{5\epsilon d}{2}\left(\sum_{i=0}^{r-1}\big(\frac{d}{2}(1+3\epsilon)\big)^i\big(\frac{d}{2}(1-2\epsilon)\big)^{r-1-i}\right) \leq \frac{5\epsilon d}{2} r \Big(\frac{d}{2}(1+3\epsilon)\Big)^{r-1} \\
&\leq 5r\epsilon\Big(\frac{d}{2}(1+3\epsilon)\Big)^r \leq 10\sqrt{\epsilon}|V_r|.
\end{aligned}$$

The first equality is due to $a^n - b^n = (a-b)(\sum_{i=0}^{n-1} a^i b^{n-1-i})$ and the last one is due to $r \leq \frac{2}{\sqrt{\epsilon}}$.

This upper bound implies that there are at least $|V_r|(1-10\sqrt{\epsilon})$ honest parties in V_r while the number of dishonest parties is at most $10\sqrt{\epsilon}|V_r|$.

6.8 Analysis of Step iii

The intuition for the next observation is simply that because V consists almost entirely of honest parties, every honest party v not yet in V will get sufficient support in Step iii from the parties in V to be included as well; indeed, any such v is expected have have close to $\frac{d}{n}|V|$ good incoming edges from the parties in V.

Proposition 5. *At the end of Step iii, with probability at least* $1 - 2^{-\Omega(\epsilon d)}$, V *contains* all *honest parties and* $O(\sqrt{\epsilon} n)$ *dishonest parties.*

Proof. Recall the notation from the proof in the previous section, and the observation that conditioned on H_r, the E_i's for $i \in H_r \setminus H_{r-1}$ are random and independent.

Setting $\tilde{H} := H_r \setminus H_{r-1}$ and $d_1 := \frac{|V|d}{n} = \Omega(\epsilon d)$, and using Corollary 2 for the final bound, it follows that for a given honest party $v \notin \tilde{H}$,

$$\Pr\left[n_V^{\mathsf{in}}(v, \mathsf{good}) < \frac{d_1}{2}\right] \leq \Pr\left[n_{\tilde{H}}^{\mathsf{in}}(v, \mathsf{good}) < \frac{d_1}{2}\right] = \Pr\left[n_{\tilde{H}}^{\mathsf{in}}(v) < \frac{d_1}{2}\right] \leq 2^{-\Omega(\epsilon d)}.$$

By union bound over all honest parties outside $\tilde{H}$, all these honest parties are added to V with probability at least $1 - 2^{-\Omega(\epsilon d)}$.

On the other hand, any active party w outside V needs at least $\frac{d_1}{2}$ good incoming edges to be admitted. These edges must come from dishonest parties in V. Since there are at most $O(\sqrt{\epsilon})|V|$ of them in V and each of them contributes to at most d good incoming edges, the number of active parties admitted to V is at most $\frac{O(\sqrt{\epsilon})|V|d}{d_1/2} = O(\sqrt{\epsilon} n)$.

6.9 Analysis of Step iv

By construction, after Step iv, V and W together obviously still contain all honest parties. Furthermore, as we show below, there is now no active party left in V and only few honest parties ended up in V. The idea here is that the active parties in V will be recongnized as being dishonest by many honest parties in V.

Proposition 6. *At the end of Step iv, with probability at least* $1 - 2^{-\Omega(\epsilon d)}$*,* V *consists of* $t + 1 - O(\sqrt{\epsilon}n)$ *honest parties and no active parties, and* W *consists of the rest of honest parties and* $O(\sqrt{\epsilon}n)$ *dishonest parties.*

Proof. Observe that $\frac{|H|d}{n} \geq \frac{d}{2}$. It follows, again using Corollary 2, that for an active party w in V, we have

$$\Pr\left[n_V^{\mathsf{in}}(w, \mathtt{bad}) < \frac{d}{4}\right] \leq \Pr\left[n_H^{\mathsf{in}}(w, \mathtt{bad}) < \frac{d}{4}\right] = \Pr\left[n_H^{\mathsf{in}}(w) < \frac{d}{4}\right] \leq 2^{-\Omega(d)}.$$

By union bound over all active parties in V, all of them are removed from V with probability at least $1 - t2^{-\Omega(d)} = 1 - 2^{-\Omega(d)}$.

On the other hand, if the honest party v is removed from V, he must receive at least $\frac{d}{4}$ bad incoming edges from dishonest parties in V. Since the number of dishonest parties is at most $a := O(\sqrt{\epsilon}n)$, there are at most $\frac{ad}{d/4} = O(\sqrt{\epsilon}n)$ honest parties removed from V in Step 2.

In order to analyze the last step (see next section), we introduce the following notation. We partition V into the set of honest parties V_H and the set of passive parties V_P. We also partition W into the set of honest parties W_H and the set of dishonest parties W_C. From above, we know that $|W| = |W_H| + |W_C| = O(\sqrt{\epsilon}n)$, $V_H \cup W_H = H$ and $|V_H| = t + 1 - O(\sqrt{\epsilon}n)$.

6.10 Analysis of Step v

Proposition 7. *Except with probability* $2^{-\Omega(d)}$*, after Step v the set* V *will contain all honest parties and at least twice as many passive parties as active ones. Therefore, Step v will output the correct secret with probability at least* $1-2^{-\Omega(d)}$*.*

Note that, given the aversary's strategy, all the previous steps of the graph algorithm are determined by the graph G. Therefore, by Proposition 3, at this point in the algorithm the E_i''s for $i \in H$ are still random and independent given V_H, V_P, W_C, W_H.

Proof. **Step v.1.** For any $i \in V_H$, ${n'}_W^{\mathsf{out}}(i)$ is expected to be $\frac{|W|d}{n} = O(\sqrt{\epsilon}d)$. By Corollary 1 we thus have

$$\Pr\left[{n'}_W^{\mathsf{out}}(i) \geq \frac{d}{8}\right] \leq 2^{-\Omega(d)}\,.$$

Hence, by union bound, all honest parties in V remain in V except with probability $2^{-\Omega(d)}$.

Let V'_P be the set of passive parties left in V after this step, and set $p := |V'_P|$. Note that $n^{\mathsf{out}}_W(v) \le d/8$ for every $v \in V$.

Step v.2. Observe that $\frac{d|V_H|}{n} = (\frac{1}{2} - O(\sqrt{\epsilon}))d$. It follows from Corollary 2 that for any honest party $i \in W_H$,

$$\Pr\Big[{n'}^{\mathsf{in}}_{V}(i, \mathsf{good}) \le \frac{d}{4}\Big] \le \Pr\Big[{n'}^{\mathsf{in}}_{V_H}(i, \mathsf{good}) \le \frac{d}{4}\Big] = \Pr\Big[{n'}^{\mathsf{in}}_{V_H}(i) \le \frac{d}{4}\Big] \le 2^{-\Omega(d)}\,.$$

Thus, all honest parties in W are added to V, except with probability $2^{-\Omega(d)}$.

On the other hand, the active parties only receive good incoming edges from passive parties in V'_P. Observe that each party in V is allowed to have at most $\frac{d}{8}$ outgoing neighbours in W. This implies there are at most $\frac{pd/8}{d/4} = \frac{p}{2}$ active parties admitted to V in this step, proving the first part of the statement.

Step v.3. Observe that the shares of the parties in S form a code with length $|V|$ and dimension $(t{+}1)$. Since the fraction of errors is at most $\frac{p}{2|V|} < \frac{(|V|-t-1)}{2|V|}$, by Proposition 1, the unique decoding algorithm will output a correct secret.

7 Parameters of Construction and Parallel Repetition

We first determine the parameters in our algorithm and then show how to reach the security parameter κ by parallel repeating this algorithm for $O(\kappa)$ times. This parallel repetition idea comes from [3]. Assume that there are n parties and m-bit secret **msg** to share among these n parties. Note that we have already set $d = n^{\sqrt{\epsilon}}$ with ϵ a small constant. Let $\log q = O(\frac{m+\log n}{\epsilon^2})$. We choose $\log|\mathcal{T}| = \log m + 5\log n$ and then the random string r_i has length $4d\log|\mathcal{T}|$. The key $key_{i,j}$ is defined over $\mathcal{T}^2$ and thus has length $2\log|\mathcal{T}|$. It follows that $|\mathcal{K}_i| = |\mathcal{K}'_i| = 2d\log|\mathcal{T}|$ and **tag** has length $4nd\log|\mathcal{T}|$. By theorem 4, (p_i, q_i) has length $\widetilde{O}(d)$. By Theorem 2 and plug $a = O(\frac{m+\log n}{\epsilon^2})$, the error probability of the MAC ϵ_{mac} is at most $\frac{4(a+4d)dt}{|\mathcal{T}|} = O(\frac{mnd}{\epsilon^2 mn^5}) \le O(\frac{1}{n^3})$. The failure probability of our reconstruction scheme consists of the error probability $\epsilon_{mac} = O(1/n^3)$ of the MAC authentication, error probability $\epsilon_{tag} = O(1/n^2)$ of reconstructing **tag** , error probability $\epsilon_{check} = 2^{-O(\epsilon d)}$ of determining the situation whether $|P| \ge \epsilon n/4$ or $|P| \le \epsilon n$, error probability $\epsilon_{code} = 2^{-\Omega(\epsilon d)}$ of code based algorithm and error probability $\epsilon_{graph} = 2^{-\Omega(\epsilon^2 d)}$ of graph algorithm. The total error probability of our algorithm is

$$\delta = \epsilon_{mac} + \epsilon_{tag} + \epsilon_{check} + (t+1)\epsilon_{graph} + \epsilon_{code} = O(\frac{1}{n^2}).$$

We summarize our result as follows.

Theorem 9. *The scheme* (**Share, Rec**) *is a* $2t{+}1$*-party* $(t, O(\frac{1}{n^2}))$*-robust secret sharing scheme with running time poly*$\big(m, n, (\frac{1}{\epsilon})^{\widetilde{O}(\frac{1}{\epsilon})}\big)$ *and share size* $\widetilde{O}(m{+}n^{\sqrt{\epsilon}})$.

Next, we describe how to achieve the security parameter $\delta = 2^{-\kappa}$ based on this "weakly-robust" secret sharing scheme (**Share**, **Rec**) with $\delta = O(\frac{1}{n^2})$. Given a secret **msg**, we run **Share**(**msg**) $q = O(\kappa)$ times to produce q robust sharings **msg**, except that the first step of **Share**(**msg**) is executed only *once*, i.e., only one set of non-robust shares $(s_1, \ldots, s_n) \leftarrow \mathbf{Sh}(\mathbf{msg})$ is produced, and then re-used in the otherwise independent q executions of **Share**(**msg**). This is exactly the same idea as that in [3]. The resulting share size is then $\widetilde{O}(m + \kappa n^{\sqrt{\epsilon}})$.

The analysis is almost the same as that in [3], so we omit it here.

Theorem 10. *The scheme* (**Share**$'$, **Rec**$'$) *is a* $2t{+}1$*-party* $(t, 2^{-\kappa})$*-robust secret sharing scheme against rushing adversary with share size* $\widetilde{O}(m + \kappa n^{\sqrt{\epsilon}})$ *and running time poly*$\big(\kappa, m, n, (\frac{1}{\epsilon})^{\widetilde{O}(\frac{1}{\epsilon})}\big)$.

8 Further Improvement and Existence Result

If we do not consider the efficiency of our algorithm, by setting proper parameters, our algorithm can also achieve the optimal $\widetilde{O}(m + \kappa)$ share size. The algorithm is exactly the same as we described above except that we set $\epsilon = \frac{1}{\log^2 n}$, $d = \Omega(\log^5 n)$ and $\gamma = \frac{\epsilon}{4} = O(\frac{1}{\log^2 n})$. This parameter setting will affect the efficiency and error probability of our algorithm. We briefly review this improvement by pointing out the differences. Since $d = \Omega(\log^5 n)$, we use the tag construction in Theorem 3 with $u = \Omega(\log^5 n)$. The error probability ϵ_{tag} of reconstructing tags now becomes $2^{-\Omega(u)} = 2^{-\Omega(\log^5 n)}$. The error probability ϵ_{Check} becomes $2^{-\Omega(\epsilon d)} = 2^{-\Omega(\log^3 n)}$. In the code based algorithm, the list size $(\frac{1}{\gamma})^{\widetilde{O}(\frac{1}{\gamma})}$ now becomes $2^{\widetilde{O}(\log^2 n)}$.[11] By taking union bound over candidates on this new list, we get

$$\epsilon_{code} = 2^{\widetilde{O}(\log^2 n)} 2^{-\Omega(\gamma d)} = 2^{\widetilde{O}(\log^2 n)} 2^{-\Omega(\log^4 n)} = 2^{-\Omega(\log^4 n)}.$$

In the graph algorithm, we bound the size of V_r and H_r. First, we notice that $d^{\log n} = n^{\log\log n}$. This implies that $r < \log n = \sqrt{\frac{1}{\epsilon}}$. In worst case scenario, we assume that $|V_r| = (\frac{d}{2}(1 + 3\epsilon))^r$ and $|H_r| = (\frac{d}{2}(1 - 2\epsilon))^r$. It follows that the number of dishonest parties is at most

$$\begin{aligned} |V_r| - |H_r| &= \Big(\frac{d}{2}(1+3\epsilon)\Big)^r - \Big(\frac{d}{2}(1-2\epsilon)\Big)^r \\ &= \frac{5\epsilon d}{2}\left(\sum_{i=0}^{r-1} \big(\frac{d}{2}(1+3\epsilon)\big)^i \big(\frac{d}{2}(1-2\epsilon)\big)^{r-1-i}\right) \le 5r\epsilon\Big(\frac{d}{2}(1+3\epsilon)\Big)^r \le 5\sqrt{\epsilon}|V_r|. \end{aligned}$$

The rest of the algorithm are the same. Therefore, the error probability of our graph algorithm now becomes $\epsilon_{graph} = 2^{-\Omega(\epsilon^2 d)} = 2^{-\Omega(\log n)}$. It follows that the total error probability of our algorithm is $2^{-\Omega(\log n)}$. The overhead of share size is $O(d) = O(\log^5 n)$. By the parallel repetition technique, we can reduce it to $2^{-\kappa}$. The share size then becomes $\widetilde{O}(m + \kappa)$. As a trade-off, the running time of our algorithm now becomes $2^{\widetilde{O}(\log^2 n)}$ which is super-polynomial in n.

[11] Here, we hide the poly$(\log\log n)$ in $\widetilde{O}(\cdot)$

Theorem 11. *There exists $2t+1$-party $(t, 2^{-\kappa})$-robust secret sharing scheme against rushing adversary with share size $\widetilde{O}(m+\kappa)$ and running time $2^{\widetilde{O}(\log^2 n)}$.*

9 Acknowledgements

CY was partially supported by the European Union Horizon 2020 research and innovation programme under grant agreement No.74079 (ALGSTRONGCRYPTO) and the National Research Foundation, Prime Minister's Office, Singapore, under its Strategic Capability Research Centres Funding Initiative.

References

1. Anne Auger and Benjamin Doerr. *Theory of Randomized Search Heuristics*. WORLD SCIENTIFIC, 2011.
2. 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, *Public-Key Cryptography – PKC 2016*, pages 327–356. Springer Berlin Heidelberg, 2016.
3. 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*, pages 58–86. Springer Berlin Heidelberg, 2016.
4. G. R. Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on(AFIPS)*, pages 313–317, 12 1979.
5. 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*, pages 118–125. Springer Berlin Heidelberg, 1994.
6. 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*, pages 195–208. Springer Berlin Heidelberg, 2012.
7. M. Cheraghchi. Nearly optimal robust secret sharing. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2509–2513, July 2016.
8. 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*, pages 503–523. Springer Berlin Heidelberg, 2001.
9. 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*, pages 313–336. Springer Berlin Heidelberg, 2015.
10. Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Information Theory*, 54(1):135–150, 2008.
11. Brett Hemenway and Rafail Ostrovsky. Efficient robust secret sharing from expander graphs. *Cryptography and Communications*, 10(1):79–99, 2018.
12. Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded reed-solomon and multiplicity codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:91, 2018.

13. Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85, 1989.
14. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

A Appendix

A.1 Folded Reed-Solomon Codes

Instead of using the Reed-Solomon codes to share our secret, our robust secret sharing scheme is encoded by the folded Reed-Solomon codes. Since the folded Reed-Solomon code is a class of MDS codes, it is an eligible candidate for threshold secret sharing scheme. Moreover, the folded Reed-Solomon codes first introduced by Guruswami and Rudra [10] can be list decoded up to $1 - R - \gamma$ fraction of errors for any constant γ. This extra nice property allows us to divide our reconstruction scheme into two scenarios, one with small number of passive parties and another with big one. Let us first introduce the formal definition of fold Reed-Solomon codes.

Let q be a prime power, $n+1 \le \frac{q-1}{s}$ and β be a primitive element of $\mathbb{F}_q$. The folded Reed-Solomon code $\mathsf{FRS}_{q,s}(n+1,d)$ is a code over $\mathbb{F}_q^s$. To every polynomial $P(X) \in \mathbb{F}_q[X]$ of degree at most d, the encoding algorithm goes as follows:

$$P(X) \mapsto \mathbf{c}_P = \left(\begin{bmatrix} P(\beta) \\ P(\beta^2) \\ \vdots \\ P(\beta^{s-1}) \end{bmatrix}, \begin{bmatrix} P(\beta^s) \\ P(\beta^{s+1}) \\ \vdots \\ P(\beta^{2s-1}) \end{bmatrix}, \cdots, \begin{bmatrix} P(\beta^{ns}) \\ P(\beta^{ns+1}) \\ \vdots \\ P(\beta^{(n+1)s-1}) \end{bmatrix} \right).$$

It is easy to verify that $\mathsf{FRS}_{q,s}(n+1,d)$ is an $\mathbb{F}_q$-linear code with code length $n+1$, rate $\frac{d+1}{(n+1)s}$ and distance at least $(n+1) - \lfloor \frac{d}{s} \rfloor$. The folded Reed-Solomon code is a class of MDS code when $d+1$ is divisible by s. In our robust secret sharing scheme, we set $n = 2t+1$ and $d+1 = (t+1)s$. For every secret $\mathbf{s} \in \mathbb{F}_q^s$, we find the $P(X)$ of degree at most d uniform at random such that $\mathbf{s} = (P(\beta), P(\beta^2), \dots, P(\beta^{s-1}))$. The party i receives the $i+1$-th component of $\mathbf{c}_P$. It is easy to verify that this scheme is a threshold secret sharing scheme with t-privacy and $t+1$-reconstruction. Moreover, if we write the n shares as

$$(P(\beta^s), P(\beta^{s+1}), \dots, P(\beta^{(n+1)s-1})) \in \mathbb{F}_q^{ns}.$$

Then, it becomes a classic Reed-Solomon codes with length ns, dimension $(t+1)s$ and distance $(n-(t+1))s+1$. We will use this fact in our robust secret sharing scheme.

Besides the MDS property, the folded Reed-Solomon codes enjoy a large list decoding radius up to the Singleton bound while the list size is bounded by a polynomial in q. There are many works aimed at reducing the list size of the folded Reed-Solomon codes. Recently, Kopparty et.al., [12] proved that the list size of the folded Reed-Solomon codes is at most a constant in γ.

Theorem 12 (Theorem 3.1 [12]). *Let $\gamma > 0$ such that $\frac{16}{\gamma^2} \leq s$. The folded Reed-Solomon code $\mathsf{FRS}_{q,s}(n,d)$ can be list decoded up to $1 - \frac{d}{sn} - \gamma$ with list size at most $(\frac{1}{\gamma})^{\frac{1}{\gamma}\log\frac{1}{\gamma}}$. Moreover, there exists a randomized algorithm that list decodes this code with above parameters in time poly$(\log q, s, d, n, (\frac{1}{\gamma})^{\frac{1}{\gamma}\log\frac{1}{\gamma}})$.*

Remark 6. By running this polynomial list decoding algorithm n times and taking the union of all its output, with probability at least $1 - 2^{-\Omega(n)}$, we will find all the codewords within distance $1 - \frac{d}{sn} - \gamma$ to the corrupted vector. This error probability is good enough for our robust secret sharing scheme. Compared with the approach in [10], the new algorithm runs faster and ensures a significantly small list of candidates.

A.2 Proof of Theorem 2

Proof. We need to verify three conditions in Definition 3.

Privacy over Randomness: It suffices to consider that all the ℓ keys are distinct. Otherwise, we keep one key for each value and apply the argument to these distinct keys. Let $(x_1, y_1), \ldots, (x_\ell, y_\ell) \in \mathbb{F}^2$ be the ℓ distinct keys. Let $\sigma_i = MAC_{(x_i,y_i)}(\mathbf{m}, \mathbf{r})$. For any $\mathbf{m} \in \mathbb{F}^a$, we will show that $(\sigma_1, \ldots, \sigma_\ell) \in \mathbb{F}^\ell$ are distributed uniformly at random. To see this, we write

$$MAC_{x,y}(\mathbf{m}, \mathbf{r}) = f_{\mathbf{m}}(x) + g_{\mathbf{r}}(x) + y$$

where $f_{\mathbf{m}}(x) = \sum_{i=1}^{a} m_i x^{i+\ell}$ and $g_{\mathbf{r}}(x) = \sum_{i=1}^{\ell} r_i x^i$. For any ℓ-tuple $(\sigma_1, \ldots, \sigma_\ell)$ $\in \mathbb{F}^\ell$, we obtain the evaluation of $g_{\mathbf{r}}(x)$ at ℓ points, i.e., $g_{\mathbf{r}}(x_i) = \sigma_i - f_{\mathbf{m}}(x_i) - y_i$. Since $g_{\mathbf{r}}$ is a polynomial of degree $\ell - 1$, the polynomial interpolation yields an unique $g_{\mathbf{r}}(x)$. This implies that for any $\mathbf{m} \in \mathbb{F}^a$, the distribution of $(\sigma_1, \ldots, \sigma_\ell)$ is uniform at random over $\mathbf{r} \in \mathbb{F}^\ell$.

Authentication: For $(\mathbf{m}, \mathbf{r}) \neq (\mathbf{m}', \mathbf{r}') \in \mathbb{F}^a \times \mathbb{F}^\ell$, $MAC_{(x,y)}(\mathbf{m}, \mathbf{r})$-$MAC_{(x,y)}(\mathbf{m}', \mathbf{r}')$ is a nonzero polynomial in x of degree at most $t + \ell$ over $\mathbb{F}$. Thus, for any $b \in \mathbb{F}$, the equation

$$MAC_{(x,y)}(\mathbf{m}, \mathbf{r}) - MAC_{(x,y)}(\mathbf{m}', \mathbf{r}') = b$$

has at most $(a + \ell)|\mathbb{F}|$ pairs (x, y) as its solutions. The desired result follows as $\frac{(a+\ell)(|\mathbb{F}|)}{|\mathbb{F}|^2} \leq \epsilon$.

Uniformity: We need to show that given any $(\mathbf{m}, \mathbf{r}) \in \mathbb{F}^a \times \mathbb{F}^\ell$, the tag $\sigma = MAC_{(x,y)}(\mathbf{m}, \mathbf{r})$ is uniform at random over the random key $(x, y) \in \mathbb{F}^2$. Let us fix $(\mathbf{m}, \mathbf{r})$. By the definition of MAC, we have

$$\sigma = MAC_{(x,y)}(\mathbf{m}, \mathbf{r}) = f_{\mathbf{m}}(x) + g_{\mathbf{r}}(x) + y.$$

For each $\sigma \in \mathbb{F}$, there exists exactly q distinct keys (x, y) to satisfy this MAC. Thus, the tag σ is uniform at random over the random key. The desired result follows.