Linear Secret Sharing Schemes from Error Correcting Codes and Universal Hash Functions

Ronald Cramer[1,2], Ivan Bjerre Damgård [⋆3], Nico Döttling [⋆,⋆⋆3], Serge Fehr[1], and Gabriele Spini [⋆⋆⋆1,2,4]

[1] CWI Amsterdam
[2] Mathematical Institute, Leiden University
[3] Dept. of Computer Science, Aarhus University
[4] Institut de Mathématiques de Bordeaux, UMR 5251, Université de Bordeaux

Abstract. We present a novel method for constructing linear secret sharing schemes (LSSS) from linear error correcting codes and linear universal hash functions in a blackbox way. The main advantage of this new construction is that the privacy property of the resulting secret sharing scheme essentially becomes independent of the code we use, only depending on its rate. This allows us to fully harness the algorithmic properties of recent code constructions such as efficient encoding and decoding or efficient list-decoding. Choosing the error correcting codes and universal hash functions involved carefully, we obtain solutions to the following open problems:

- A linear near-threshold secret sharing scheme with both linear time sharing and reconstruction algorithms and large secrets (i.e. secrets of size $\Omega(n)$). Thus, the computational overhead per shared bit in this scheme is *constant*.
- An efficiently reconstructible robust secret sharing scheme for $n/3 \leq t < (1-\epsilon) \cdot n/2$ corrupted players (for any constant $\epsilon > 0$) with shares of optimal size $O(1+\lambda/n)$ and secrets of size $\Omega(n+\lambda)$, where λ is the security parameter.

Keywords: Linear Secret Sharing Schemes, Linear Time Sharing, Robust Secret Sharing

1 Introduction

Linear secret sharing schemes (LSSS) are the central building block for information-theoretically secure cryptographic primitives such as multiparty computation, robust secret sharing, as well as for two-party primitives via the so-called MPC-in-the-head paradigm [17,19]. Naturally, the computational efficiency of the LSSS directly influences the efficiency of the implied primitive, so it is interesting to construct schemes where both sharing a secret and reconstruction is as efficient as possible.

It is well known that there is a natural correspondence between linear codes and LSSS [22,4,10]. Since there is a rich body of literature about codes with efficient encoding and decoding, one might hope that this would lead to very efficient secret sharing schemes, ideally with linear time (in the number of players) to share and reconstruct a secret. However, for applications, one typically needs an LSSS where both the privacy and reconstruction thresholds are constant fractions of the number of players. If we try to reach this goal using the standard method for going from codes to LSSSs, we will need (a family of) codes where both the code itself and its dual are (asymptotically) good codes. But unfortunately, the known codes that are efficiently (linear time) en- and decodable have very bad dual codes. Therefore it was previously an open problem to construct LSSSs with linear time sharing and reconstruction.

⋆ The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.
⋆⋆ Supported by European Research Commission Starting Grant no. 279447.
⋆⋆⋆ Supported by the Algant-Doc doctoral program.

In this paper, we suggest a new paradigm for constructing LSSSs based on linear codes and linear universal hash functions. The main advantage of this approach is that it gives us a good privacy threshold *no matter which code we start from*. We can therefore use the full power of the known constructions of linear codes with efficient encoding and (list) decoding. We also suggest several applications of the technique. We remark however that there is no obvious way to obtain multiplicative secret sharing schemes via our construction paradigm. But constructions of general MPC protocols such as [6,4] require multiplicative secret sharing schemes. Thus, we consider it an interesting open problem to extend our construction paradigm to multiplicative secret sharing schemes.

A paradigm for building LSSS. First we note that any LSSS can be seen as being derived from a linear code C of length n and a linear function $\mathsf{h} : \mathbb{F}^k \mapsto \mathbb{F}^\ell$. We obtain an LSSS as follows: to share a secret $\mathbf{s} \in \mathbb{F}^\ell$ we choose at random $\mathbf{x} \in \mathbb{F}^k$ subject to $\mathsf{h}(\mathbf{x}) = \mathbf{s}$, encode $\mathbf{x}$ in C, and give each entry in the resulting codeword to a player, where in the most general case, each player may receive more than one value.

We can now say which player subsets can reconstruct the secret and which sets have no information: Let A be a player subset and let $\Pi_A : \mathbb{F}^k \mapsto \mathbb{F}^{|A|}$ be the linear mapping that on input $\mathbf{x}$ outputs the shares given to A when $\mathbf{x}$ is the randomness used in the LSSS. Then A can reconstruct the secret if and only if $\mathsf{dim}(\mathsf{h}(\mathsf{ker}(\Pi_A))) = 0$, and A has no information on the secret if and only if $\mathsf{dim}(\mathsf{h}(\mathsf{ker}(\Pi_A))) = \ell$, i.e., h is surjective on $\mathsf{ker}(\Pi_A)$. This characterisation was first given in Theorem 10 of [4], which we have rephrased here to match our notation (see also Lemma 3 below). It was noted already in [4] that the privacy threshold can be estimated via the dual distance, but that this bound is not sharp. Nevertheless, previous works have often established privacy for LSSS schemes from the dual distance. A notable exception of this is [2], where large privacy of certain secret sharing schemes is established via a descent field technique, but the dual distance of the corresponding code is merely constant. Thus [2] uses the characterization of [4] in its full generality, however for a very special class of codes. As we shall see, the dual distance bound is generally very far from being sharp, and there is great potential in avoiding the dual distance approach.

We now explain our construction of an LSSS from a code C, such that some constant fraction of the players can reconstruct and any set smaller than some (other) constant fraction has no information. Reconstruction is easy to handle: if we have at least r shares, then we can compute the secret if C allows decoding from $n - r$ erasures, and r can be $\Theta(n)$ if C is (asymptotically) good. However, it much less clear what we can say about privacy in general (e.g., if C has a bad dual distance).

Our main idea for solving this is to notice that a player set A has partial information on $\mathbf{x}$ from its shares, namely $\mathbf{x}$ must be in some subspace defined by the shares known to A. Now, suppose we choose h at random from a universal family of linear hash functions. It is well known that such a random function acts as a good randomness extractor, so we may hope that A has little or no information on $\mathsf{h}(\mathbf{x})$, at least if A is small enough.

In the following, we show how this intuition can be formalised. It turns out that because the hash functions are linear and the partially unknown string resides in a subspace, things are even better than what the general theory of extractors would predict: fix any small enough corrupted set A and choose h at random from the family. Then with very high probability, the h we have chosen will satisfy that $\mathsf{h}(\mathbf{x})$ is uniform in the view of the adversary. We can then simply apply a union bound over all the desired privacy sets to conclude that a random choice of h yields an LSSS with privacy threshold a constant fraction of the number of players. For constant rate (family of) codes, this fraction can be chosen as a constant arbitrarily close to the rate.

We emphasise that the random choice of h only needs to be done once and for all when the LSSS is set up. Then, with overwhelming probability, the LSSS we have constructed has perfect privacy and reconstruction as required for an LSSS. If we were willing to spend a very long (exponential) time in the set-up phase we could verify that a candidate h indeed gives us privacy for all the desired player sets, and this way remove the probabilistic aspect from the construction.

1.1 Applications

Linear time LSSS Clearly, for any linear secret sharing scheme a secret can be shared using a quadratic amount (in the number of players) of field operations. In the light of the above mentioned linear time encodable codes, the question arises whether secret sharing schemes with a linear time sharing phase can be constructed. Recently, Druk and Ishai [9] provided a construction of a linear time near-threshold linear secret sharing scheme. For a near-threshold scheme, one can choose the (relative) privacy and reconstruction thresholds as arbitrarily close constants. Their main tool for this construction is a family of linear time encodable linear codes with good distance and good dual distance. Their construction of linear secret sharing schemes follows Massey's blueprint [22], i.e. exploiting the dual distance to establish privacy. The codes constructed in [9] are not known to be linear time erasure decodable. While their construction allows to compute shares in linear time, reconstruction is more expensive for their scheme, i.e. it requires quadratic time with preprocessing. However more importantly, the secret space of their scheme is limited to a constant number of bits. But this means that in their scheme the computational overhead per shared bit is linear, rather than constant.

Following the paradigm sketched in the last paragraph, we obtain a linear time near-threshold LSSS with secrets of size $\Omega(n)$. In particular, the computational overhead per shared bit in this scheme is constant. Just like in the scheme of Druk and Ishai [9], we can choose the (relative) privacy and reconstruction thresholds as arbitrarily close constants. Our construction uses the following ideas. First, we construct a linear secret sharing scheme where sharing and reconstruction of *random* secrets can be performed in linear time. We do this by plugging a linear time computable linear hash function and linear time en- and decodable code C into our basic construction. We can then choose a random $\mathbf{x}$ and compute a (random) secret $\mathbf{s} = \mathsf{h}(\mathbf{x})$ and a share vector $\mathbf{c}$ by encoding $\mathbf{x}$ in C. Note that this can be done without having to invert the hash function which would be too inefficient with known constructions.

We bootstrap this into a standard secret sharing scheme by the following trick. To share a given secret $\mathbf{s}$, first compute a random secret $\mathbf{s}'$ together with corresponding shares $\mathbf{c}$. We can now use $\mathbf{s}'$ to one-time-pad encrypt $\mathbf{s}$, i.e. compute a ciphertext $\mathbf{y} = \mathbf{s} + \mathbf{s}'$. To distribute $\mathbf{y}$ to the players, we *disperse* $\mathbf{y}$, i.e. we encode $\mathbf{y}$ using a linear time encodable erasure correcting code and share the codeword symbols among the players. Note that we have effectively shared $\mathbf{y}$ non-privately, but this is not a problem as $\mathbf{y}$ is not private anyway. Thus, the overall overhead to share the secret $\mathbf{s}$ is linear. To reconstruct, we can use linear time erasure correction algorithms (provided by the codes). Therefore, both sharing and reconstruction can be performed in linear time.

Linear time UC commitments. In a commitment scheme, a prover commits to a bit string towards a verifier who does not learn the string at commitment time, yet the prover is committed to his choice and can later reveal it to the verifier in a convincing way. Universally composable (UC) commitments provide the strongest possible security for commitments schemes, guaranteeing security in any context. Until recently, we only knew UC commitments based on expensive public-key primitives. In [8] (see also [11]), Damgård et al. propose a general scheme that constructs UC commitments with small amortised overhead from any sufficiently good LSSS, assuming a once-and-for-all preprocessing phase where some oblivious transfers are executed. They show how to get UC commitments with linear complexity for the verifier (linear in the size of the string committed to), but left it as an open problem to get linear complexity also for the prover. This problem was solved very recently by Cascudo et al. in [1] using a new construction of a non-threshold LSSS and a new variant of the MPC-in-the-head paradigm.

Our results can be used to give an alternative and simpler solution: since the efficiency of the original construction in [8] is inherited directly from the underlying LSSS, we immediately get linear complexity for both parties by simply plugging in our linear time LSSS. It is also interesting to note that if our scheme can be made multiplicative, then this and another result from [8] would immediately imply non-interactive UC Zero-Knowledge proofs with linear complexity for both prover and verifier.

Robust secret sharing with constant size shares. A *robust* secret sharing scheme is a secret sharing scheme with the additional property that reconstruction of the secret is possible (and, ideally, computationally

feasible) even if some of the shares are incorrect. More concretely, a robust secret sharing scheme satisfies standard t-privacy as well as robust-reconstructability, where the latter means that given all n shares, the secret can be reconstructed even if t of them come from dishonest players and may be incorrect. In this work, we consider robust secret sharing in the setting of a non-rushing adversary; this means that the dishonest players have to announce their incorrect shares *before* getting to see the shares of the honest players.

If $t < n/3$ then standard error correction provides robustness for free. On the other extreme, if $t \geq n/2$ then robust secret sharing is not possible. Thus, the interesting range is $n/3 \leq t < n/2$. Here, robust secret sharing is possible, but we have to allow a small error probability of $2^{-\lambda}$, and additional "checking data" needs to be appended to the actual shares. The goal is to optimize the tradeoff between error probability and the increase in share size.

Cramer, Damgård and Fehr [5] gave a construction of a robust secret sharing scheme based on so-called *Algebraic Manipulation Detection* (AMD) codes (even though the terms robust secret sharing and AMD codes were not used there). Roughly speaking, an AMD code enables to detect certain manipulations — namely *algebraic* manipulations — of encoded messages. The robust secret sharing scheme then simply works by sharing an AMD encoding of the secret (using a standard linear secret sharing scheme), and the robust reconstruction is by going through all sets of possibly honest players, reconstruct from their shares, and verify correctness of the reconstructed AMD encoding. By making the AMD codeword large enough, resulting in an overhead in the share size of $O(\lambda+n)$, this procedure finds the correct secret except with probability $2^{-\lambda}$. An obvious downside of this scheme is that the robust reconstruction procedure is not efficient, as there is an exponential number of sets of possibly honest players to be considered.

In [3], based on very different techniques, Cevallos, Fehr, Ostrovsky and Rabani proposed a robust secret sharing scheme, with similar parameters: overhead $O(\lambda + n \log n)$ for an error probability of $2^{-\lambda}$, but which offers an *efficient* robust reconstruction. Both these schemes work for any fraction $t/n < \frac{1}{2}$, and neither becomes significantly better in terms of this error probability versus the size of the checking data if we bound t/n away from $\frac{1}{2}$ by a small constant.

Based on our new paradigm for building LSSSs, we construct a new robust secret sharing scheme. Our construction works when t/n is bounded away from $\frac{1}{2}$ by an arbitrary small positive constant. In this regime, we can consider *ramp* schemes, for which there is a gap between the privacy threshold t and the standard reconstruction threshold r, while still allowing for robust reconstruction in the presence of t faulty shares. In ramp schemes, the (actual) shares may be *smaller* than the secret (by a factor $r-t$). In our construction, we can additionally reduce the size of the checking data per share; this is in contrast to the above mentioned constructions when generalized to ramp schemes where the size of the checking data stays $O(\lambda)$.

Our construction can be seen as an efficient variant of the approach from [5]. We will secret share an AMD codeword, but this time using our construction of LSSS from above and choosing the underlying code C to be one that allows efficient *list decoding*. This means that we can consider the contributed shares as a codeword with errors and apply the list decoding algorithm. This will return a small (i.e., polynomial size) list of possible code words from C, each of these will suggest a possible AMD codeword. Thus, we only have a small number of candidates to check for correctness of the AMD encoding. This not only provides efficiency of the reconstruction (in contrast to the scheme of [5]), but also allows better parameters: using a highly list-decodable code as underlying code in our construction, we obtain that for every constant $\tau < \frac{1}{2}$ there exists a robust secret sharing scheme for threshold $t = \tau n$ that supports secrets of size linear in $n + \lambda$ and has shares of size $O(1 + \lambda/n)$, i.e. the size of the shares actually *de*creases in n.

2 Preliminaries

We will assume basic concepts from linear algebra such as linear maps and their kernels. For any prime power q, we will denote the finite field with q elements by $\mathbb{F}_q$. We will denote vectors $\mathbf{x}$ with boldface letters. We will also consider vectors whose components are vectors, e.g. a vector $\mathbf{x} \in (\mathbb{F}_q^m)^n$ whose components are $\mathbb{F}_q^m$ vectors. For a set $A \subseteq \{1, \ldots, n\}$ we will use $\Pi_A : (\mathbb{F}_q^m)^n \to (\mathbb{F}_q^m)^{|A|}$ to denote the projection onto the components in A. For a vector $\mathbf{x} \in (\mathbb{F}_q^m)^n$ and a set $A \subseteq \{1, \ldots, n\}$ we will also use the notation $\mathbf{x}_A = \Pi_A(\mathbf{x})$.

2.1 Probability

The binary entropy function $H : [0, 1/2] \to [0, 1]$ is given by $H(0) := 0$ and $H(x) := -x \cdot \log(x) - (1-x) \cdot \log(1-x)$ for $x \in (0, 1/2]$. For $0 \leq t/n \leq 1/2$ we can upper bound binomial coefficients by $\binom{n}{t} \leq 2^{H(t/n)\cdot n}$, for a proof see e.g. [23]. We will also use the Markov inequality (see also [23]).

Lemma 1 (Markov Inequality). *Let X be a non-negative random variable defined on $\mathbb{R}$ for which $\mathsf{E}[X]$ exists. Then it holds for every $x > 0$ that*

$$\Pr[X \geq x] \leq \frac{\mathsf{E}[X]}{x}.$$

Corollary 1. *Let X be a random variable with finite support $\mathcal{X} \subseteq \mathbb{R}$ which assumes its minimum at x_0 and its second smallest value at $x_1 > x_0$. Then it holds that*

$$\mathsf{E}[X] \geq x_0 + (x_1 - x_0) \cdot \Pr[X \neq x_0].$$

Proof. The expectation $\mathsf{E}[X]$ exists as X has a finite support. Since X assumes its minimum at x_0 it holds that $X - x_0$ is non-negative. By the Markov inequality it holds that

$$\Pr[X \neq x_0] = \Pr[X \geq x_1] = \Pr[X - x_0 \geq x_1 - x_0] \leq \frac{\mathsf{E}[X] - x_0}{x_1 - x_0},$$

as $\mathsf{E}[X - x_0] = \mathsf{E}[X] - x_0$ by linearity of expectation. Thus the claim follows.

2.2 Universal Hashing

Universal hash functions are a central tool in information-theoretically secure cryptography.

Definition 1 (Universal Hash Functions). *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. A family $\mathcal{H}$ of functions $\mathcal{X} \to \mathcal{Y}$ is called* family of universal hash functions *if it holds for all distinct $x, x' \in \mathcal{X}$ that*

$$\Pr_{\mathsf{H} \leftarrow_\$ \mathcal{H}}[\mathsf{H}(x) = \mathsf{H}(x')] \leq \frac{1}{|\mathcal{Y}|},$$

where H is chosen uniformly from $\mathcal{H}$.

For families $\mathcal{H}$ of $\mathbb{F}_q$*-linear* functions, meaning that both $\mathcal{X}$ and $\mathcal{Y}$ are $\mathbb{F}_q$-vector spaces and each $\mathsf{h} \in \mathcal{H}$ is a $\mathbb{F}_q$-linear function, the condition of Definition 1 can be rephrased as follows: $\mathcal{H}$ is a family of universal hash functions if and only if for all $\mathbf{x} \in \mathcal{X} \setminus \{\mathbf{0}\}$

$$\Pr_{\mathsf{H} \leftarrow_\$ \mathcal{H}}[\mathsf{H}(\mathbf{x}) = \mathbf{0}] \leq \frac{1}{|\mathcal{Y}|}.$$

We then naturally refer to $\mathcal{H}$ as a *family of $\mathbb{F}_q$-linear universal hash functions.*

There are various efficient families of linear universal hash functions, such random matrices or random Toeplitz matrices (see e.g. [21]). Ishai et al. [17] constructed a linear time computable family of linear universal hash functions, c.f. Section 5.

2.3 Error Correcting Codes

We assume basic concepts from coding theory. Error correcting codes are used to encode messages in such a way that the encoding is resilient against certain types of errors. Formally, a $\mathbb{F}_q$-linear error correcting code C of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_q^n$. We say that C is an $m-$*folded code*, if C is a k-dimensional subspace of $(\mathbb{F}_q^m)^n$. This basically means that the alphabet of C is $\mathbb{F}_q^m$ rather than $\mathbb{F}_q$. An m-folded code C of length n can be naturally interpreted as a code of length $m \cdot n$. In this view,

the possible error patterns in a folded code are *burst errors* rather than symbol errors. The rate R of an m-folded $[n,k]$ code is defined by $R = \frac{k}{mn}$, i.e. $1/R$ is the factor by which the code expands messages. We will denote distinguished encoding and decoding algorithms[5] for a linear code C by $\mathsf{C.Encode}$ and $\mathsf{C.Decode}$. We will denote the (generalized) Hamming distance for vectors $\mathbf{x}, \mathbf{y} \in (\mathbb{F}_q^m)^n$ by $\mathsf{d}(\mathbf{x}, \mathbf{y}) = |\{i \mid \mathbf{x}_i \neq \mathbf{y}_i\}|$, i.e. $\mathsf{d}(\mathbf{x}, \mathbf{y})$ counts in how many blocks $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{F}_q^m$ the vectors $\mathbf{x}$ and $\mathbf{y}$ differ.

2.4 Secret Sharing Schemes

A secret sharing scheme allows a dealer to distribute a secret to n players in such a way that the players of any large enough set of players can jointly reconstruct the secret from their shares, whereas small coalitions of players have no information on the secret. A secret sharing scheme is called *linear*, if any linear combination of valid share vectors results in a valid share vector of the linear combination applied to the respective secrets. This is summarized in the following definition.

Definition 2. *Let $\mathbb{F}_q$ be a finite field, and let l, m and $t < r \leq n$ be positive integers. A linear secret sharing scheme* LSSS *consists of two algorithms* $\mathsf{LSSS.Share}(\cdot)$ *and* $\mathsf{LSSS.Reconstruct}(\cdot)$. *For every $\mathbf{s} \in \mathbb{F}_q^l$,* $\mathsf{LSSS.Share}(\mathbf{s})$ *outputs a vector of shares $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_n) \in (\mathbb{F}_q^m)^n$. We require the following three properties.*

- *t-privacy: for all $\mathbf{s}, \mathbf{s}' \in \mathbb{F}_q^l$ and every $A \in \{1, \ldots, n\}$ of size $|A| = t$, the restrictions $\mathbf{c}_A$ and $\mathbf{c}'_A$ of $\mathbf{c} = \mathsf{LSSS.Share}(\mathbf{s})$ and $\mathbf{c}' = \mathsf{LSSS.Share}(\mathbf{s}')$ to the coordinates in A have the same probability distribution.*
- *r-reconstructability: for every $\mathbf{s} \in \mathbb{F}_q^l$ and every $Q \in \{1, \ldots, n\}$ of size $|Q| = r$, it holds for $\mathbf{c} = \mathsf{LSSS.Share}(\mathbf{s})$ that $\mathsf{LSSS.Reconstruct}(\tilde{\mathbf{c}}) = \mathbf{s}$, where $\tilde{\mathbf{c}}$ is a vector with $\tilde{\mathbf{c}}_Q = \mathbf{c}_Q$ and $\tilde{\mathbf{c}}_{\bar{Q}}$ only contains erasure symbols, i.e. $\tilde{\mathbf{c}}_{\bar{Q}} = \perp^{|\bar{Q}|}$.*
- *Linearity: If $\mathbf{c}_1$ and $\mathbf{c}_2$ are respective sharings of $\mathbf{s}_1$ and $\mathbf{s}_2$, then $\alpha\mathbf{c}_1 + \beta\mathbf{c}_2$ is a sharing of $\alpha\mathbf{s}_1 + \beta\mathbf{s}_2$.*

We emphasize that we do not require $r = t+1$; secret sharing schemes with $r > t+1$ are sometimes referred to as *ramp* schemes. We may use this terminology sometimes to emphasize that we allow $r > t+1$. For schemes with $r = t+1$, it is well known that the size of the secret cannot be bigger than the size of a share, i.e., $l \leq m$. For a ramp schemes, this generalizes to $l \leq (r-t) \cdot m$. The *rate* of a secret sharing scheme is given by $\rho = \frac{l}{mn}$. Using this terminology, the above can be expressed as follows. For any n-player ramp scheme that satisfies τn-privacy and σn-reconstructability, the rate of the scheme can be at most $\rho \leq \sigma - \tau$.

3 Subspace Surjectivity of Linear Universal Hash Functions

In this section, we provide a general theorem about universal hash functions. The theorem states that if we fix an r-dimensional subspace V of $\mathbb{F}_q^k$, then a randomly chosen *linear* universal hash function H from a family which maps $\mathbb{F}_q^k$ to $\mathbb{F}_q^l$ is surjective on V, except with probability $q^{-(r-l)}$. By saying that H is surjective on V, we mean that $\mathsf{H}(V) = \mathbb{F}_q^l$.

This theorem can be interpreted in information theoretic terms. We can identify a subspace V with the uniform distribution $\mathbf{v}$ on V and consider $\mathbf{v}$ as a *linear* source of randomness. Since V has dimension l, the q-ary min-entropy of $\mathbf{v}$ is at least l. From this point of view, the theorem states that universal hash functions are good extractors for linear sources, i.e. they extract such sources *perfectly*, except with probability $q^{-(r-l)}$. Perfect extraction in this context means that $\mathsf{H}(\mathbf{v})$ is exactly the uniform distribution. The leftover hash lemma [16] states that universal hash functions yield good extractors for sources with a sufficient amount of min-entropy. We can actually establish a weaker version of this theorem based on the leftover hash lemma. However, the parameters obtained by our theorem are tighter than parameters obtainable by the leftover hash lemma. The best probability of failure obtainable via the leftover hash lemma is $q^{-(r-l)/2}$, which is worse than the bound given in the theorem.

[5] such as linear time algorithms for these tasks

Theorem 1. *Let $\mathcal{H}$ be a family of linear universal hash functions $\mathbb{F}_q^k \to \mathbb{F}_q^l$. Further let V be a subspace of $\mathbb{F}_q^k$ of dimension at least r. Let $\mathsf{H} \leftarrow_\$ \mathcal{H}$ be chosen uniformly at random and then fixed. Then it holds that $\mathsf{H}(V) = \mathbb{F}_q^l$ (i.e. H is surjective on V), except with probability $q^{-(r-l)}$ over the choice of H.*

Proof. For any linear function $\mathsf{h} \in \mathcal{H}$, it holds that $\mathsf{h}(V) = \mathbb{F}_q^l$ if and only if $\dim(V \cap \ker(\mathsf{h})) = \dim(V) - l$, which is equivalent to $|V \cap \ker(\mathsf{h})| = \frac{|V|}{q^l}$. Now, let $\mathsf{H} \leftarrow_\$ \mathcal{H}$ and define the random variable $X = |V \cap \ker(\mathsf{H})|$ (depending on H). By the above it holds that H is surjective on V if and only if $X = |V|/q^l$. For each $\mathbf{v} \in V$, define the random variable

$$X_{\mathbf{v}} = \begin{cases} 1 & \text{if } \mathsf{H}(\mathbf{v}) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Clearly, it holds that $X = \sum_{\mathbf{v} \in V} X_v$. Since $X_0 = 1$, we have that $X = 1 + \sum_{\mathbf{v} \in V \setminus \{\mathbf{0}\}} X_{\mathbf{v}}$. Moreover, X assumes its minimum at $x_0 = \frac{|V|}{q^l}$ and its second smallest value at $x_1 = \frac{|V|}{q^{l-1}}$. We will now compute the expectation of X. For each $\mathbf{v} \in V \setminus \{\mathbf{0}\}$ it holds that

$$\mathsf{E}[X_{\mathbf{v}}] = \Pr_{\mathsf{H} \leftarrow_\$ \mathcal{H}}[\mathsf{H}(\mathbf{v}) = \mathbf{0}] \leq q^{-l},$$

as $\mathcal{H}$ is a family of universal hash functions. By linearity of expectation, it holds that

$$\mathsf{E}[X] = 1 + \sum_{\mathbf{v} \in V \setminus \{\mathbf{0}\}} \mathsf{E}[X_{\mathbf{v}}] = 1 + \frac{|V| - 1}{q^l}.$$

By Corollary 1 and the fact that $|V| \geq q^r$ it holds that

$$\begin{aligned} \Pr\left[X \neq \frac{|V|}{q^l}\right] &\leq \frac{1 + \frac{|V|-1}{q^l} - \frac{|V|}{q^l}}{\frac{|V|}{q^{l-1}} - \frac{|V|}{q^l}} \\ &= \frac{q^l - 1}{|V| \cdot (q-1)} \\ &\leq \frac{q^l}{|V|} \leq q^{-(r-l)}. \end{aligned}$$

Consequently, it holds that $\mathsf{H}(V) = \mathbb{F}_q^l$, except with probability $q^{-(k-l)}$.

Given a collection $\mathcal{V}$ of at most r-dimensional subspaces of $\mathbb{F}^k$, taking a union bound over all $V \in \mathcal{V}$ and applying Theorem 1 yields that it holds for all $V \in \mathcal{V}$ that $\mathsf{H}(V) = \mathbb{F}_q^l$, except with probability $|\mathcal{V}| \cdot q^{-(r-l)}$. This is summarized in Corollary 2.

Corollary 2. *Let $\mathcal{H}$ be a family of linear universal hash functions $\mathbb{F}_q^k \to \mathbb{F}_q^l$ and $\mathcal{V}$ be a collection of subspaces of $\mathbb{F}_q^k$, each of dimension at least r. Let $\mathsf{H} \leftarrow_\$ \mathcal{H}$ be chosen uniformly at random and then fixed. Then it holds for all $V \in \mathcal{V}$ that $\mathsf{H}(V) = \mathbb{F}_q^l$ (i.e. H is surjective on V), except with probability $|\mathcal{V}| \cdot q^{-(r-l)}$ over the choice of H.*

4 Linear Secret Sharing Schemes from Codes and Universal Hash Functions

In this section, we will provide our basic LSSS construction. In the following sections, we will provide applications based on this scheme. The scheme $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ is defined by an m-folded $\mathbb{F}_q$-linear code C and an $\mathbb{F}_q$-linear surjective function h. A secret $\mathbf{s}$ is shared by first sampling a random preimage $\mathbf{x}$ of $\mathbf{s}$ under the function h, and then encoding $\mathbf{x}$ using the (folded) code C, obtaining a share vector $\mathbf{c} \in (\mathbb{F}_q^m)^n$. Each share $\mathbf{c}_i$ is a vector in $\mathbb{F}_q^m$. Notice that we can efficiently sample a preimage $\mathbf{x}$ of $\mathbf{s}$ under the function h by using basic

linear algebra, since the function h is linear. More specifically, we can sample such an $\mathbf{x}$ by first computing any preimage $\mathbf{x}_1$ of $\mathbf{s}$ and then randomize $\mathbf{x}_1$ by adding a uniformly random $\mathbf{x}_2 \leftarrow_\$ \mathsf{ker}(\mathsf{h})$ to $\mathbf{x}_1$, i.e. setting $\mathbf{x} \leftarrow \mathbf{x}_1 + \mathbf{x}_2$. Though this sharing algorithm Share is efficient, it still involves a rather costly inversion of h, which has overhead $O(n^3)$ when implemented naively. Thus, even if both h and $\mathsf{C.Encode}$ can be computed *super-efficiently* (e.g. in linear time), Share does not achieve the same efficiency.

In order to take full advantage of super-efficient h and $\mathsf{C.Encode}$, we will provide an alternative sharing algorithm $\mathsf{ShareRandom}$ which computes both h and $\mathsf{ShareRandom}$ only in forward direction. Thus, if both h and $\mathsf{C.Encode}$ are super-efficient, then so is $\mathsf{ShareRandom}$. However, $\mathsf{ShareRandom}$ only generates shares for randomly chosen secrets. In Section 5 we show how a secret sharing scheme with super-efficient random sharing algorithm can be bootstrapped into a secret sharing scheme with super-efficient standard sharing algorithm Share. We will now provide our construction.

Construction 1 *Let* C *be an* m*-folded* $\mathbb{F}_q$*-linear* $[n,k]$ *code with encoding and decoding procedures* $\mathsf{C.Encode}$ *and* $\mathsf{C.Decode}$ *and let* $\mathsf{h} : \mathbb{F}_q^k \to \mathbb{F}_q^l$ *be a surjective* $\mathbb{F}_q$*-linear function. The secret sharing scheme* $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ *is given by the following sharing and reconstruction procedures.*

$\mathsf{Share}(\mathbf{s})$:
- $\mathbf{x} \leftarrow_\$ \mathsf{h}^{-1}(\mathbf{s})$
- $\mathbf{c} \leftarrow \mathsf{C.Encode}(\mathbf{x})$
- Output share vector $\mathbf{c}$

$\mathsf{Reconstruct}(\tilde{\mathbf{c}})$:
- $\mathbf{x} \leftarrow \mathsf{C.Decode}(\tilde{\mathbf{c}})$
- If $\mathbf{x} = \perp$
  - Output $\perp$
- $\mathbf{s} \leftarrow \mathsf{h}(\mathbf{x})$
- Output $\mathbf{s}$

$\mathsf{ShareRandom}()$:
- $\mathbf{x} \leftarrow_\$ \mathbb{F}_q^k$
- $\mathbf{c} \leftarrow \mathsf{C.Encode}(\mathbf{x})$
- $\mathbf{s} \leftarrow \mathsf{h}(\mathbf{x})$
- Output secret $\mathbf{s}$ and share vector $\mathbf{c}$

First observe that the linearity of $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ follows straightforwardly from the linearity of the code C and the function h. Moreover, all reconstruction properties of $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ follow from corresponding properties of the code C.

Lemma 2. *Let* C *be an* m*-folded* $\mathbb{F}_q$*-linear* $[n,k]$ *code and* $\mathsf{h} : \mathbb{F}_q^k \to \mathbb{F}_q^l$ *be a surjective* $\mathbb{F}_q$*-linear function. Secrets are elements of* $\mathbb{F}_q^l$*, whereas (single) shares are elements of* $\mathbb{F}_q^m$*. Assume that* $\mathsf{C.Decode}$ *can correct* $n-r$ *erasures. Then* $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ *is an* n*-player LSSS which fulfills the linearity and* r*-reconstructability properties. Moreover,* $\mathsf{ShareRandom}$ *implements the same functionality as choosing* $\mathbf{s}$ *at random, computing* $\mathbf{c} \leftarrow \mathsf{Share}(s)$ *and outputting* $(\mathbf{s}, \mathbf{c})$*.*

Proof. First notice that since h is surjective, the sharing algorithm Share can compute a share vector $\mathbf{c}$ for every message $\mathbf{s} \in \mathbb{F}_q^l$. The $\mathbb{F}_q$-linearity property follows directly from the $\mathbb{F}_q$-linearity if C and h. If r shares are given, we can assemble a vector $\tilde{\mathbf{c}}$ that has at most $n-r$ erasures. Consequently, $\mathsf{C.Decode}(\tilde{\mathbf{c}})$ will recover the correct $\mathbf{x}$ and we can compute the secret $\mathbf{s} = \mathsf{h}(\mathbf{x})$. To see that $\mathsf{ShareRandom}$ computes the same functionality as choosing $\mathbf{s}$ uniformly at random and computing $\mathbf{c} \leftarrow \mathsf{Share}(\mathbf{s})$, notice that the $\mathbf{x}$ computed by $\mathsf{Share}(\mathbf{s})$ can be written as $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$, where $\mathbf{x}_1$ is a vector uniquely determined by $\mathbf{s}$ in an l-dimensional subspace W of $\mathbb{F}_q^k$ with $\mathsf{h}(W) = \mathbb{F}_q^l$ and $\mathbf{x}_2$ is chosen uniformly at random from $\mathsf{ker}(\mathbf{h})$. Thus if $\mathbf{s}$ is chosen uniformly at random, then $\mathbf{x}$ is also distributed uniformly at random in $\mathbb{F}_q^k$, just as the $\mathbf{x}$ computed by $\mathsf{ShareRandom}()$. The claim follows.

We will now determine under which conditions $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ fulfills the privacy property. In the first step, we first derive a general condition on the function h which is actually a necessary and sufficient requirement. In the second step, we will show that this requirement is met with overwhelming probability when the function h is chosen randomly from a family of universal hash functions. To simplify the analysis, we will identify the linear function $\mathsf{h} : \mathbb{F}_q^k \to \mathbb{F}_q^l$ with another linear function $\Phi : \mathsf{C} \to \mathbb{F}_q^l$. This is always possible as C is

a k-dimensional $\mathbb{F}_q$-vectorspace and thus isomorphic to $\mathbb{F}_q^k$. In fact, we can basically define the function Φ by $\Phi(\mathbf{c}) = \mathsf{h}(\mathsf{C.Decode}(\mathbf{c}))$ for all $\mathbf{c} \in \mathsf{C}$. We will denote projections of shares to a subset A of players by $\Pi_A : \mathsf{C} \to (\mathbb{F}_q^m)^{|A|}$.

Lemma 3. *A set $A \subseteq \{1, \dots, n\}$ has privacy* if and only if $\Phi(\ker(\Pi_A)) = \mathbb{F}_q^l$, *where* $\ker(\Pi_A) = \{\mathbf{x} \in \mathsf{C} \mid \Pi_A(\mathbf{x}) = \mathbf{0}\}$.

Proof. First assume that $\Phi(\ker(\Pi_A)) = \mathbb{F}_q^l$. Fix a subspace $W \subseteq \ker(\Pi_A)$ of dimension l such that $\Phi(W) = \mathbb{F}_q^l$. As $W \cap \ker(\Phi) = \{\mathbf{0}\}$, it holds that $\mathsf{C} = W \oplus \ker(\Phi)$, i.e. we can write every $\mathbf{c} \in \mathsf{C}$ as $\mathbf{c} = \mathbf{c}_s + \mathbf{c}_r$, for unique $\mathbf{c}_s \in W$ and $\mathbf{c}_r \in \ker(\Phi)$. Now, let $\mathbf{c} = \mathsf{Share}(\mathbf{s})$. As $\mathbf{c} \in \mathsf{C}$, we can write $\mathbf{c}$ as

$$\mathbf{c} = \mathbf{c}_s + \mathbf{c}_r,$$

where $\mathbf{c}_s \in W$ is a unique vector such that $\Phi(\mathbf{c}_s) = \mathbf{s}$ and $\mathbf{c}_r$ is chosen uniformly at random from $\ker(\Phi)$. Now, it holds that

$$\Pi_A(\mathbf{c}) = \Pi_A(\mathbf{c}_s + \mathbf{c}_r) = \Pi_A(\mathbf{c}_s) + \Pi_A(\mathbf{c}_r) = \Pi_A(\mathbf{c}_r),$$

as $\mathbf{c}_s \in W \subseteq \ker(\Pi_A)$. Thus, $\Pi_A(\mathbf{c}) = \Pi_A(\mathbf{c}_r)$ is distributed independently of $\mathbf{s}$ and we can conclude that privacy holds.

For the converse direction, assume that $\Phi(\ker(\Pi_A)) \subsetneq \mathbb{F}_q^l$, i.e. there exists an $\mathbf{s}^* \in \mathbb{F}_q^l \backslash \Phi(\ker(\Pi_A))$. We will now show that if $\mathbf{c} = \mathsf{Share}(\mathbf{0})$ and $\mathbf{c}^* = \mathsf{Share}(\mathbf{s}^*)$, then the projections $\Pi_A(\mathbf{c})$ and $\Pi_A(\mathbf{c}^*)$ can always be distinguished. We claim that $\Pi_A(\mathbf{c}^*) \notin \Pi_A(\ker(\Phi))$. To see this, assume towards contradiction that $\Pi_A(\mathbf{c}^*) \in \Pi_A(\ker(\Phi))$. Then there exists a $\mathbf{h} \in \ker(\Phi)$ such that $\Pi_A(\mathbf{c}^*) = \Pi_A(\mathbf{h})$. Since Π_A is linear, it holds that $\Pi_A(\mathbf{c}^* - \mathbf{h}) = \mathbf{0}$ and consequently $\mathbf{c}^* - \mathbf{h} \in \ker(\Pi_A)$. From this, however follows

$$\Phi(\mathbf{c}^*) = \Phi(\mathbf{c}^* - \mathbf{h}) \in \Phi(\ker(\Pi_A)),$$

as $\mathbf{h} \in \ker(\Phi)$. This however contradicts $\mathbf{s}^* = \Phi(\mathbf{c}^*) \notin \Phi(\ker(\Pi_A))$ and we conclude that it must hold that $\Pi_A(\mathbf{c}^*) \notin \Pi_A(\ker(\Phi))$. Finally, notice that $\mathbf{c} \in \ker(\Phi)$ as $\mathbf{c} = \mathsf{Share}(\mathbf{0})$. Thus it holds that $\Pi_A(\mathbf{c}) \in \Pi_A(\ker(\Phi))$. We can therefore easily (and perfectly) distinguish $\Pi_A(\mathbf{c})$ from $\Pi_A(\mathbf{c}^*)$ by checking whether it is in $\Pi_A(\ker(\Phi))$. This contradicts the privacy property and we can conclude the proof.

We will now use the characterization of Lemma 3 and Corollary 2 to show that if we instantiate h with a randomly chosen linear universal hash function, then we obtain a good linear secret sharing scheme.

Lemma 4. *Let C be a m-folded $\mathbb{F}_q$-linear $[n, k]$ code and let $\mathcal{H}$ be a family of $\mathbb{F}_q$-linear universal hash functions $\mathbb{F}_q^k \to \mathbb{F}_q^l$. Let $R = \frac{k}{mn}$ be the rate of C, let $\rho = \frac{l}{nm}$ and let $\tau > 0$ and $\eta > 0$ be constants. Given that $R \geq \rho + \eta + \tau + H(\tau)/(m \cdot \log(q))$, then there exists a $\mathsf{h} \in \mathcal{H}$ such that $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ has τn-privacy. Moreover, such a function h can be chosen randomly with success-probability $1 - q^{-\eta n m}$.*

Proof. Let $\Psi : \mathsf{C} \to \mathbb{F}_q^k$ be the isomorphism that corresponds to the function $\mathsf{C.Decode}(\cdot)$. For each set $A \subseteq \{1, \dots, n\}$ of size at most $t = \tau n$, it holds that $\ker(\Pi_A) \subseteq \mathsf{C}$ is a subspace of dimension at least $k - mt$, as C has dimension k and the image of Π_A has dimension at most mt. Thus, $\Psi(\ker(\Pi_A)) \subseteq \mathbb{F}_q^k$ also has dimension at least $k - mt$, as Ψ is an isomorphism. Consequently $\mathcal{V} = \{\Psi(\ker(\Pi_A)) \mid A \in \{1, \dots, n\}, |A| = t\}$ is a collection of subspaces of dimension at least $k - mt$. Moreover, as A is taken over all subsets of $\{1, \dots, n\}$ of size t, it holds that

$$|\mathcal{V}| \leq \binom{n}{t} \leq 2^{H(t/n) \cdot n} = 2^{H(\tau) \cdot n} = q^{\frac{H(\tau)}{m \log(q)} \cdot mn}.$$

By Lemma 3, $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ has privacy for all A with $|A| \leq t$ if it holds $\mathsf{h}(V) = \mathbb{F}_q^l$ for each $V \in \mathcal{V}$. By Corollary 2, it holds for all $V \in \mathcal{V}$ that $\mathsf{H}(V) = \mathbb{F}_q^l$, except with probability

$$|\mathcal{V}| \cdot q^{-(k-mt-l)} \leq q^{-(k-mt-l-\frac{H(\tau)}{m \log(q)} \cdot mn)} = q^{-(R-\rho-\tau-\frac{H(\tau)}{m \log(q)}) \cdot mn} \leq q^{-\eta mn},$$

over the choice of $\mathsf{H} \leftarrow_\$ \mathcal{H}$, as $R \geq \rho + \eta + \tau + H(\tau)/(m \cdot \log(q))$. Thus, $\mathsf{LSSS}_{\mathsf{C},\mathsf{H}}$ has t-privacy, except with probability $q^{-\eta mn}$ over the choice of $\mathsf{H} \leftarrow_\$ \mathcal{H}$. This concludes the proof.

Remark 1. It can be seen rather easily that the function H in Lemma 4 must either be chosen randomly or depending on the code C, i.e. for any fixed function h we can find a code C^* such that $\mathsf{LSSS}_{\mathsf{C}^*,\mathsf{h}}$ does not provide any privacy. Thus, to obtain a construction that is oblivious of the specific code we use, randomization is strictly necessary.

5 Linear Time Sharing and Reconstruction

As our first application, we will show how to construct a linear secret sharing scheme with linear time sharing and reconstruction phase. Choosing a linear time encodable code C and a linear time computable function h yields that the sharing procedure $\mathsf{ShareRandom}$ of Construction 1 is also linear time computable. This is, in turn, not true for the sharing procedure Share of Construction 1. We circumvent this issue by providing a sharing algorithm that first computes shares of a random secret $\mathbf{s}'$ using $\mathsf{ShareRandom}$, then uses $\mathbf{s}'$ to one-time-pad encrypt the actual secret $\mathbf{s}$. The ciphertext $\mathbf{s}+\mathbf{s}'$ is then distributed by applying a standard information dispersal technique, i.e. we encode $\mathsf{s}+\mathsf{s}'$ using an erasure correcting code and append the codeword symbols to the shares. This basically results in a doubling of the share size. This technique bears resemblance to the construction of Krawczyk [20]. However, while in [20] information dispersal is applied to reduce the share size (using a computationally secure encryption scheme), we use this technique to salvage the linear time computability of the (non-random) sharing algorithm.

We will start by providing an instantiation of Construction 1 with linear time random sharing and reconstruction algorithms. Moreover, the scheme we provide will be *near*-threshold. We need both a family of linear time computable universal hash functions and linear time en- and decodable codes. Ishai et al. [18] construct a family of $\mathbb{F}_2$-linear universal hash functions which can be computed in linear time. This result has recently been generalized by Druk and Ishai [9] to any finite field, but the binary case of [18] is sufficient for our application.

Theorem 2 (Ishai et al. [18]). *For every $l < k$, there exists a family $\mathbb{F}_2$-linear universal hash functions $\mathcal{H}$ mapping $\mathbb{F}_2^k$ to $\mathbb{F}_2^l$ which can be computed in time linear in k.*

There is a large corpus of work dealing with linear time encodable codes, starting with the seminal work of Spielman [25]. To the best of our knowledge, the currently best known parameters can be obtained using a family of codes by Guruswami and Indyk [12].

Theorem 3 (Guruswami-Indyk [12]). *For every rate R and every sufficiently small ϵ (depending on R) there exists an infinite family of m-folded $\mathbb{F}_2$-linear codes $\{\mathsf{C}_n\}$ of rate R, where $m = O\left(\frac{\log(1/\epsilon)}{\epsilon^4 R}\right)$, such that the codes from the family can be encoded in linear time and also decoded in linear time from an $1-R-\epsilon$ fraction of erasures.*

We will now instantiate the the linear secret sharing scheme $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ of Construction 1 with the codes from Theorem 3 and universal hash functions from Theorem 2.

Lemma 5. *For all constants $0 < \tau < \sigma < 1$ and every rate $\rho < \sigma - \tau$ there exists an infinite family of $\mathbb{F}_2$-linear secret schemes $\{\mathsf{LSSS}_n^1\}$ with τn-privacy, σn-reconstructability and rate ρ. The shares of LSSS_n^1 have size m bits, where $m > 0$ is a constant. Furthermore, $\mathsf{LSSS}_n^1.\mathsf{ShareRandom}$ and $\mathsf{LSSS}_n^1.\mathsf{Reconstruct}$ can be computed in linear time. Moreover, such a scheme LSSS_n^1 can be constructed randomly with success-probability $1-2^{-\eta m n}$ (for some constant $\eta > 0$ depending on τ, σ and ρ).*

Proof. We will instantiate the linear secret sharing scheme $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ from Construction 1 with a linear code C_n from the family $\{\mathsf{C}_n\}$ of $\mathbb{F}_2$-linear codes from Theorem 3 and a function h from the family $\mathcal{H}$ of $\mathbb{F}_2$-linear universal hash functions from Theorem 2. We now show how to choose the parameters for this instantiation.

By Lemma 4, in order to obtain a secret sharing scheme with τn privacy, we need to select an m-folded code C_n from the above family of length n and rate R such that $R \geq \rho + \eta + \tau + H(\tau)/m$ for an arbitrarily small constant η. Moreover, as by Lemma 2 we ne need to be able to correct a $1-\sigma$ fraction of erasures to

have σn reconstruction, we need to choose C_n such that $1-\sigma \leq 1-R-\epsilon$, equivalently $R \leq \sigma - \epsilon$. Both constraints together yield

$$\sigma - \tau - \rho \geq \epsilon + \eta + \frac{H(\tau)}{m}. \tag{1}$$

Since $\sigma > \tau$ and $\sigma - \tau > \rho$, the left hand side of Inequality 1 is a constant greater than 0. It is clear from Theorem 3 that we can choose the folding parameter m as an arbitrarily large constant, thereby also decreasing ϵ. Consequently, the terms ϵ and $\frac{H(\tau)}{m}$ become arbitrarily small and we can choose a sufficiently small $\eta > 0$ such that the inequality is satisfied. Setting $R = \sigma - \epsilon$ we found admissible constants $R, m, \eta, \epsilon > 0$ such that $R \geq \rho + \eta + \tau + H(\tau)/m$. Now let C_n be a code of length n from the above family that matches these constants. By Theorem 3 such a code exists for all constants $R, m, \epsilon > 0$. Now let $\mathcal{H}$ be a the family of universal hash functions from 2 mapping $\mathbb{F}_2^{Rmn}$ to $\mathbb{F}_2^{\rho mn}$ obtained by Theorem 2. By Lemma 4, choosing the universal hash function H randomly from $\mathcal{H}$ yields that $\mathsf{LSSS}_{\mathsf{C},\mathsf{H}}$ has τ privacy, except with probability $2^{-\eta mn}$.

Notice that the computational overhead per shared bit in LSSS_n^1 is constant for $\mathsf{ShareRandom}$. We will now bootstrap the scheme LSSS_n^1 given by Lemma 5 into a secret sharing scheme with linear time sharing and reconstruction algorithms.

Construction 2 *Let C' be a (folded) $\mathbb{F}_q$-linear $[n, l]$ code with encoding and decoding procedures $\mathsf{C}'.\mathsf{Encode}$ and $\mathsf{C}'.\mathsf{Decode}$ and let LSSS^1 be a $\mathbb{F}_q$-linear secret sharing scheme with a sharing procedure $\mathsf{LSSS}_1.\mathsf{ShareRandom}()$ for random secrets. The secret sharing scheme LSSS^2 is given by the following sharing and reconstruction procedures.*

$\mathsf{Share}(\mathbf{s})$:
- $(\mathbf{s}', \mathbf{c}) \leftarrow \mathsf{LSSS}^1.\mathsf{ShareRandom}()$
- $\mathbf{d} \leftarrow \mathsf{C}'.\mathsf{Encode}(\mathbf{s}' + \mathbf{s})$
- Parse $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$ and $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_n)$
- Output $\mathbf{z} = ((\mathbf{c}_1, \mathbf{d}_1), \ldots, (\mathbf{c}_n, \mathbf{d}_n))$

$\mathsf{Reconstruct}(\tilde{\mathbf{z}})$:
- Parse $\tilde{\mathbf{z}} = ((\tilde{\mathbf{c}}_1, \tilde{\mathbf{d}}_1), \ldots, (\tilde{\mathbf{c}}_n, \tilde{\mathbf{d}}_n))$
- $\tilde{\mathbf{c}} \leftarrow (\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_n)$
- $\tilde{\mathbf{d}} \leftarrow (\tilde{\mathbf{d}}_1, \ldots, \tilde{\mathbf{d}}_n)$
- $\mathbf{s}' \leftarrow \mathsf{LSSS}^1.\mathsf{Reconstruct}(\tilde{\mathbf{c}})$
- $\mathbf{y} \leftarrow \mathsf{C}'.\mathsf{Decode}(\tilde{\mathbf{d}})$
- If $\mathbf{s}' = \perp$ or $\mathbf{y} = \perp$
  - Output $\perp$
- Otherwise
  - Output $\mathbf{y} - \mathbf{s}'$

Lemma 6. *Assume that LSSS^1 provides t-privacy and r-reconstructability, and $\mathsf{LSSS}^1.\mathsf{ShareRandom}$ is linear time computable. Assume further that C' is linear time encodable and that $\mathsf{C}'.\mathsf{Decode}$ can decode from r-erasures. Then LSSS^2 also has t-privacy and r-reconstructability and $\mathsf{LSSS}^2.\mathsf{Share}$ is linear time computable. Furthermore, if both $\mathsf{LSSS}^1.\mathsf{Reconstruct}$ and $\mathsf{C}'.\mathsf{Decode}$ are linear time computable, then $\mathsf{LSSS}^2.\mathsf{Reconstruct}$ is also linear time computable.*

Proof. Linear time computability of $\mathsf{LSSS}^2.\mathsf{Share}$ and $\mathsf{LSSS}^2.\mathsf{Reconstruct}$ follows straightforwardly from the linear time computability of $\mathsf{LSSS}^1.\mathsf{ShareRandom}$ and $\mathsf{C}'.\mathsf{Encode}$ as well as $\mathsf{LSSS}^1.\mathsf{Reconstruct}$ and $\mathsf{C}'.\mathsf{Decode}$ respectively.

To see that LSSS^2 has r-reconstructability, observe that $\mathsf{LSSS}^1.\mathsf{Reconstruct}(\tilde{\mathbf{c}})$ recovers $\mathbf{s}'$ from t as long as $\tilde{\mathbf{c}}$ contains at most r erasures. Likewise, $\mathsf{C}'.\mathsf{Decode}(\tilde{\mathbf{d}})$ recovers $\mathbf{x} = \mathbf{s} + \mathbf{s}'$ from $\tilde{\mathbf{d}}$ as long as $\tilde{\mathbf{d}}$ contains at most r erasures. r-reconstructability of LSSS^2 follows.

To see that LSSS^2 has t-privacy, let $\mathbf{z} = ((\mathbf{c}_1, \mathbf{d}_1), \ldots, (\mathbf{c}_n, \mathbf{d}_n))$ be a vector of shares generated by $\mathsf{LSSS}^2.\mathsf{Share}(\mathbf{s})$. For any $A \subseteq \{1, \ldots, n\}$ of size at most t, it holds by the t-privacy of LSSS^1 that $\mathbf{s}'$ is

distributed uniformly at random given the shares $\mathbf{c}_A$. Thus, the $(\mathbf{c}_A, \mathbf{s}+\mathbf{s}')$ is distributed independently of $\mathbf{s}$. But the same holds for $(\mathbf{c}_A, \mathbf{d}_A)$, as $\mathbf{d}_A$ can be computed from $\mathbf{s}+\mathbf{s}'$. Consequently, t-privacy of LSSS^2 follows.

Finally, plugging the linear secret sharing scheme LSSS^1_n obtained in Lemma 5 into Construction 2, we obtain the main result for this section. For the sake of simplicity, as code C' in Construction 2 we can choose the same code C as in Lemma 5 and match its rate to the rate of LSSS^1_n. We conclude the following theorem.

Theorem 4. *For all constants $0 < \tau < \sigma < 1$ and every rate $\rho < \sigma - \tau$ there exists an infinite family of $\mathbb{F}_2$-linear secret scheme $\{\mathsf{LSSS}^2_n\}$ with τn-privacy, σn-reconstructability and rate ρ. The shares of LSSS^2_n have size m, where $m > 0$ is a constant. $\mathsf{LSSS}^2_n.\mathsf{Share}$ and $\mathsf{LSSS}^2_n.\mathsf{Reconstruct}$ can be computed in linear time. Moreover, such a scheme LSSS^2_n can be constructed randomly with success-probability $1 - 2^{-\eta m n}$ (for some constant $\eta > 0$ depending on τ, σ and ρ).*

6 Robust Secret Sharing with Constant Size Shares

In this section, we show how our generic construction of LSSSs from codes gives rise to new *robust* secret sharing schemes, i.e., to schemes where the secret can be correctly reconstructed even if some of the shares provided are incorrect.

The idea behind our new scheme is to instantiate Construction 1 with a *highly list-decodable* code C. When confronted with the task of reconstructing the secret in the presence of faulty shares, this allows us to narrow down the list of candidate secrets to a small set. To single out the right secret, we will *precode* it using an AMD code, as introduced in [7]. Informally, an AMD code is a (key-less) code that is resilient towards certain — namely algebraic — manipulations.

This construction is similar to the construction of Cramer, Damgård and Fehr [5]. However, the fact that our construction allows us to use a list-decodable code (whereas [5] uses standard Shamir secret sharing [24]) makes our scheme computationally efficient, in contrast to the robust reconstruction procedure of [5], which involves a brute-force search over all subsets of size t. Furthermore, in the regime we consider, namely when t/n is bounded away from $\frac{1}{2}$, we get better parameters than previous work.

6.1 Formal Definitions and Building Blocks

We start by formalizing the notion of a robust secret sharing scheme.

Definition 3. *A linear secret sharing scheme LSSS is (t, δ)-robust if there exists an additional algorithm $\mathsf{LSSS.RobustReconstruct}$ with the property that for every secret $\mathbf{s}$ and for every subset $A \subset \{1, \ldots, n\}$ of size $|A| = t$, the following holds. If $\mathbf{c} = \mathsf{LSSS.Share}(\mathbf{s})$, and $\tilde{\mathbf{c}}$ is such that $\tilde{\mathbf{c}}_{\bar{A}} = \mathbf{c}_{\bar{A}}$ and $\tilde{\mathbf{c}}_A$ only depends on $\mathbf{c}_A$, then $\mathsf{LSSS.RobustReconstruct}(\tilde{\mathbf{c}}) = \mathbf{s}$ except with probability δ.*

In the range $n/3 \leq t < n/2$, robust secret sharing is only possible if we allow a non-zero error probability δ, and we append some additional "checking data" to the actual shares. The goal is to optimize the trade-off between this overhead in the share size and δ. As outlined above, our construction is based on using a list-decodable code in our general construction of LSSS from codes.

Definition 4. *An m-folded $\mathbb{F}_q$-linear $[n, k]$ code C is said to be (t, ℓ)-list decodable if there exists an efficient algorithm $\mathsf{C.ListDecode}$ such that for any codeword $\mathbf{c} \in \mathsf{C}$ and any error pattern $\mathbf{e} \in (\mathbb{F}_q^m)^n$ of weight at most t, $\mathsf{C.ListDecode}(\mathbf{c}+\mathbf{e})$ produces a list of all elements $\mathbf{x} \in \mathbb{F}_q^k$ with $\mathsf{d}(\mathsf{C.Encode}(\mathbf{x}), \mathbf{c}+\mathbf{e}) \leq t$. Furthermore, the size of the list is at most ℓ.*

We will now state two results for highly list-decodable codes. The first one is due to Guruswami and Rudra [13] as well as Guruswami and Wang [14] and states that m-folded Reed Solomon codes are highly list-decodable. The second result, due to Guruswami and Xing [15], states that certain m-folded algebraic geometric codes are highly list-decodable.

Theorem 5 (List-decodability of Folded Reed Solomon Codes [13,14]). *For any rate $0 < R < 1$ and $\epsilon > 0$, any large enough integer $m > 0$ (depending on R and ϵ) and for any integer $n > 0$ there exist a prime power $q = q(n) = O(n)$ and an m-folded $\mathbb{F}_q$-linear code C of length n and rate R, such that C is efficiently $(\tau n, \ell)$-list decodable with $\tau = 1 - R - \epsilon$ and $\ell = \mathsf{poly}(n)$. The list decoder has runtime $\mathsf{poly}(n, m)$.*

Theorem 6 (List-decodability of Folded Algebraic Geometric Codes [15]). *For any rate $0 < R < 1$ and $\epsilon > 0$, and for any large enough integer $m > 0$ (depending on R and ϵ) there exist a constant prime power q and an infinite family of m-folded $\mathbb{F}_q$-linear codes $\{\mathsf{C}_n\}$, such that the rate of C_n is R, and C_n is efficiently $(\tau n, \ell)$-list decodable with $\tau = 1 - R - \epsilon$ and $\ell = \mathsf{poly}(n)$. The list decoder has runtime $\mathsf{poly}(n, m)$.*

Notice that in both constructions the runtime of the list decoder is polynomial in both the code length n and the folding parameter m. This means that we can choose the folding parameter super constant and still have efficient list decodability. Additionally, we make use of AMD codes (restricting ourselves to $\mathbb{F}_q$-linear spaces for simplicity).

Definition 5 (Algebraic Manipulation Detection Codes [7]). *Let q be a prime-power, $l > k$ be integers and $\delta > 0$. A (q^k, q^l, δ)-AMD code AMD consists of a probabilistic encoding algorithm $\mathsf{AMD.Encode} : \mathbb{F}_q^k \to \mathbb{F}_q^l$ and a (deterministic) decoding algorithm $\mathsf{AMD.Decode} : \mathbb{F}_q^l \to \mathbb{F}_q^k \sqcup \{\perp\}$, such that the following holds for every $\mathbf{x} \in \mathbb{F}_q^l$.*

- *Correctness:* $\mathsf{AMD.Decode}(\mathsf{AMD.Encode}(\mathbf{x})) = \mathbf{x}$ *with probability 1.*
- *Manipulation detection: for every offset $\mathbf{\Delta} \in \mathbb{F}_q^l$, and for $\mathbf{c}$ generated as $\mathbf{c} \leftarrow \mathsf{AMD.Encode}(\mathbf{x})$, it holds that $\mathsf{AMD.Decode}(\mathbf{c} + \mathbf{\Delta}) \in \{\perp, \mathbf{x}\}$ except with probability at most δ.*

A simple example AMD code is given by $\mathsf{AMD.Encode} : \mathbb{F}_q^k \to \mathbb{F}_q^k \times \mathbb{F}_q^k \times \mathbb{F}_q^k$, $s \mapsto (s, r, sr)$, where r is uniformly random from $\mathbb{F}_q^k$, and the multiplication sr is given by fixing an isomorphism of $\mathbb{F}_q$-vector spaces $\mathbb{F}_q^k \to \mathbb{F}_{q^k}$, and with the obvious decoding: checking the multiplicative relation. It is not hard to show that this AMD code has error probability $\delta = q^{-k}$. In our construction, we use a slightly more sophisticated AMD code, due to [7], given by

$$\mathsf{AMD.Encode} : \mathbb{F}_q^d \to \mathbb{F}_q^d \times \mathbb{F}_q \times \mathbb{F}_q,\ s \mapsto \left(s, r, r^{d+2} + \sum_{i=1}^{d} s_i r^i\right)$$

where r is uniformly random from $\mathbb{F}_q^k$, $\mathrm{char}(\mathbb{F}_q) \nmid d + 2$, and with the obvious decoding. This construction gives rise to the following claim.

Lemma 7 ([7]). *For any prime power q and integers $l > 2\kappa > 0$, there exists a $\left(q^{l-2\kappa}, q^l, (l - 2\kappa + 1)/q^\kappa\right)$-AMD code.*

6.2 The Construction

In order to have a modular exposition, we first introduce the notion of a list reconstructible secret sharing scheme. In a nutshell, list reconstructible secret sharing is a weak version of robust secret sharing. Instead of requiring reconstruction of the correct secret (in the presence of faulty shares), we merely require reconstruction of a short *list* of possible candidates of which one is the correct secret. In addition to that, we require some linearity property.

Definition 6. *We say that a linear secret sharing scheme LSSS is (t, ℓ)-list reconstructible, if there exists an efficient algorithm $\mathsf{LSSS.ListReconstruct}()$, such that for all $\mathbf{e}$ of weight at most t, the following holds. $\mathsf{LSSS.ListReconstruct}(\mathbf{e})$ outputs a list of length ℓ containing $\mathbf{0}$, and for any secret $\mathbf{s}$ and its share vector $\mathbf{c}$ we have*

$$\begin{aligned}\mathsf{LSSS.ListReconstruct}(\mathbf{c} + \mathbf{e}) &= \mathbf{s} + \mathsf{LSSS.ListReconstruct}(\mathbf{e}) \\ &= \{\mathbf{s} + \mathbf{w} \mid \mathbf{w} \in \mathsf{LSSS.ListReconstruct}(\mathbf{e})\},\end{aligned}$$

We now show that, not very surprisingly, using a list-decodable code in Construction 1 results in a list reconstructable secret sharing scheme.

Lemma 8. *Let* C *be an* m*-folded* $\mathbb{F}_q$*-linear* $[n,k]$ *code and* $\mathsf{h} : \mathbb{F}_q^k \to \mathbb{F}_q^l$ *an* $\mathbb{F}_q$*-linear function, and let* $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ *be the linear secret sharing scheme resulting from Construction 1. If* C *is* (t,ℓ)*-list-decodable then* $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ *is* (t,ℓ)*-list reconstructible.*

Proof. $\mathsf{ListReconstruct}$ simply works by running $\mathsf{C.ListDecode}$ and applying h to all the elements in the list output by $\mathsf{C.ListDecode}$. In order to show that Definition 6 is satisfied, due to the linearity of h it is sufficient to show that

$$\mathsf{C.ListDecode}(\mathbf{c}+\mathbf{e}) = \mathbf{m} + \mathsf{C.ListDecode}(\mathbf{e}),$$

for any $\mathbf{m} \in \mathbb{F}_q^k$, $\mathbf{c} = \mathsf{C.Encode}(\mathbf{m})$, and any error vector $\mathbf{e}$ of weight at most t.

First of all, the bound on the size of the list, and that it contains $\mathbf{0}$, are obvious. For any $\mathbf{w} \in \mathsf{C.ListDecode}(\mathbf{e})$, we have that $\mathsf{d}(\mathsf{C.Encode}(\mathbf{w}), \mathbf{e}) \leq t$. By linearity, it holds that

$$\mathsf{C.Encode}(\mathbf{m}+\mathbf{w}) = \mathbf{c} + \mathsf{C.Encode}(\mathbf{w}).$$

Therefore,

$$\mathsf{C.Encode}(\mathbf{m}+\mathbf{w}) - (\mathbf{c}+\mathbf{e}) = \mathsf{C.Encode}(\mathbf{w}) - \mathbf{e}$$

from which follows that

$$\mathsf{d}(\mathsf{C.Encode}(\mathbf{m}+\mathbf{w}), \mathbf{c}+\mathbf{e}) = \mathsf{d}(\mathsf{C.Encode}(\mathbf{w}), \mathbf{e}) \leq t,$$

i.e. $\mathbf{m}+\mathbf{w} \in \mathsf{C.ListDecode}(\mathbf{c}+\mathbf{e})$. Similarly, for any $\mathbf{w} \in \mathsf{C.ListDecode}(\mathbf{c}+\mathbf{e})$, we have that

$$\mathsf{C.Encode}(\mathbf{w}-\mathbf{m}) = \mathsf{C.Encode}(\mathbf{w}) - \mathbf{c},$$

and consequently

$$\mathbf{e} - \mathsf{C.Encode}(\mathbf{w}-\mathbf{m}) = (\mathbf{c}+\mathbf{e}) - \mathsf{C.Encode}(\mathbf{w}).$$

This proves that the two lists, $\mathsf{C.ListDecode}(\mathbf{c}+\mathbf{e})$ and $\mathbf{m} + \mathsf{C.ListDecode}(\mathbf{e})$, contain exactly the same elements, which was to be proven.

Instantiating the above with the list decodable codes from Theorem 5 and Theorem 6 respectively yields the following Lemma.

Lemma 9. *For any* $\tau < \frac{1}{2}$*, any* $\tau < \sigma < 1-\tau$*, any* $\rho < \sigma - \tau$ *and any sufficiently large* $m > 0$ *we have that:*

- *For every integer* $n > 0$ *there exists a* $q = O(n)$ *and an* n*-player* $\mathbb{F}_q$*-linear secret sharing scheme* LSSS_n *with* τn*-privacy,* σn*-reconstruction and* $(\tau n, \mathsf{poly}(n))$*-list reconstruction. Furthermore, the rate of* LSSS_n *is* ρ *and the shares of* LSSS_n *are elements of* $\mathbb{F}_q^m$*. The list reconstruction algorithm has runtime* $\mathsf{poly}(n,m)$*.*
- *There exists a constant prime power* q *and an infinite family of* $\mathbb{F}_q$*-linear secret sharing schemes* $\{\mathsf{LSSS}_n\}$*, such that* LSSS_n *is an* n*-player scheme with* τn*-privacy,* σn*-reconstruction and* $(\tau n, \mathsf{poly}(n))$*-list reconstruction. Furthermore, the rate of* LSSS_n *is* ρ *and the shares of* LSSS_n *are elements of* $\mathbb{F}_q^m$*. The list reconstruction algorithm has runtime* $\mathsf{poly}(n,m)$*.*

Proof. We will instantiate the scheme $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ from Construction 1 enhanced to a list-reconstructible scheme by Lemma 8 with either the codes from Theorem 5 or Theorem 6 respectively and a suitable family $\mathcal{H}$ of universal hash functions.

By Lemma 4, in order to obtain a secret sharing scheme with τn privacy and rate ρ, we need to select an m-folded code C from one of the above families with rate R such that

$$R \geq \rho + \eta + \tau + \frac{H(\tau)}{m \log(q)}$$

for an arbitrarily small constant η. To get list reconstructability, we need to be able to list-decode a τ fraction of errors. Thus, we need to choose the constants R and ϵ for the above families such that $\tau \leq 1 - R - \epsilon$, which is equivalent to $R \leq 1 - \tau - \epsilon$. Together, these two constraints yield a new constraint

$$1 - 2\tau - \rho \geq \epsilon + \eta + \frac{H(\tau)}{m \log(q)}. \tag{2}$$

The left-hand side of Inequality 2 is a constant greater than 0, as $\rho < \sigma - \tau < 1 - 2\tau$ and $1 - 2\tau > 0$, as $\tau < \frac{1}{2}$. Thus, we can fulfill the constraints by choosing sufficiently small constants $\epsilon > 0$ and $\eta > 0$ and an m greater than a sufficiently large constant and setting $R = 1 - \tau - \epsilon$.

- Using the folded RS codes provided by Theorem 5 in Construction 1, we can conclude that for every $n > 0$ there exist a prime power $q = O(n)$ and an n-player $\mathbb{F}_q$-linear secret sharing scheme $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ with τn-privacy, σn-reconstruction, rate ρ, shares from $\mathbb{F}_q^m$ and which is $(\tau n, \mathsf{poly}(n))$-list reconstructible. Here, we use a linear universal hash function h chosen from a family $\mathcal{H}$ of $\mathbb{F}_q$-linear universal hash functions which maps $\mathbb{F}_q^{Rmn}$ to $\mathbb{F}_q^{\rho mn}$.
- Using the folded AG codes provided by Theorem 6 in Construction 1, we can conclude that there exists a constant prime power q and an infinite family of $\mathbb{F}_q$-linear secret sharing schemes $\{\mathsf{LSSS}_n\}$, such that LSSS_n is an n-player $\mathbb{F}_q$-linear secret sharing scheme $\mathsf{LSSS}_{\mathsf{C},\mathsf{h}}$ with τn-privacy, σn-reconstruction, rate ρ, shares from $\mathbb{F}_q^m$ and which is $(\tau n, \mathsf{poly}(n))$-list reconstructible. Here, we use a linear universal hash function h chosen from a family $\mathcal{H}$ of $\mathbb{F}_q$-linear universal hash functions which maps $\mathbb{F}_q^{Rmn}$ to $\mathbb{F}_q^{\rho mn}$.

This concludes the proof.

Construction 3 *Let* LSSS_1 *be an* n*-player linear secret sharing scheme with secret space* $\mathbb{F}_q^l$*, and say that it has* t*-privacy and* r*-reconstructability, as well as* (t, ℓ)*-list reconstructability. Let further* AMD *be a* (q^k, q^l, δ)*-AMD code. We define the secret sharing scheme* LSSS_3*, having message space* $\mathbb{F}_q^k$ *and share spaces equal to those of* LSSS_1*, by the following sharing and reconstruction procedures:*

```
Share(s):
    z ← AMD.Encode(s)
    c ← LSSS_1.Share(z)
    Output c = (c_1, ... c_n)
```

```
RobustReconstruct(c̃):
    L ← LSSS_1.ListReconstruct(c̃)
    For z̄ ∈ L:
        s̄ ← AMD.Decode(z̄)
        If s̄ ≠ ⊥
            Output s̄
    Output ⊥
```

Lemma 10. *The scheme* LSSS_3 *given above is a* $(t, \ell\delta)$*-robust linear secret sharing scheme with* t*-privacy and* r*-reconstructability.*

As for efficiency, the running time of Share is equal to the sum of the running times of $\mathsf{AMD.Encode}$ and $\mathsf{LSSS}_1.\mathsf{Share}$; the running time of $\mathsf{RobustReconstruct}$ is equal to he sum of the running time of $\mathsf{LSSS}_1.\mathsf{ListReconstruct}$ and ℓ times the running time of $\mathsf{AMD.Decode}$.

Proof. The fact that LSSS_3 has t-privacy and r-reconstruction follows immediately from the t-privacy and r-reconstruction of LSSS_1. We will now show that LSSS_3 can correctly reconstruct a secret from n shares where up to t are incorrect, except with probability at most $\ell\delta$. Let $\mathbf{c} = \mathsf{LSSS}_3.\mathsf{Share}(\mathbf{s})$ for some adversarially chosen secret $\mathbf{s}$. Assume the adversary $\mathcal{A}$ corrupts a set A of players, where $|A| \leq t$. Thus the corrupted share vector $\tilde{\mathbf{c}}$ can be written as

$$\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e},$$

where $\mathbf{e}$ is an additive error with support A. Since $\mathcal{A}$ computes $\tilde{\mathbf{c}}_A$ from $\mathbf{c}_A$, which is independent of $\mathbf{z} = \mathsf{AMD.Encode}(\mathbf{s})$ by the t-privacy of LSSS_1, $\mathbf{e}$ is independent of $\mathbf{z}$ (given $\mathbf{s}$). We will consider the error-probability of $\mathsf{RobustReconstruct}$, i.e. the probability that $\mathsf{RobustReconstruct}(\tilde{\mathbf{c}})$ outputs something different from $\mathbf{s}$.

Consider the list $E = \mathsf{LSSS}_1.\mathsf{ListReconstruct}(\mathbf{e})$. As the weight of $\mathbf{e}$ is at most t, it holds that $\mathbf{0} \in E$ and

$$L = \mathsf{LSSS}_1.\mathsf{ListReconstruct}(\tilde{\mathbf{c}}) = \mathbf{z} + E,$$

and thus in particular that $\mathbf{z} \in L$. Moreover, as the error $\mathbf{e}$ is independent of $\mathbf{z}$, it also holds that E is independent of $\mathbf{z}$. Hence, it holds for each $\mathbf{r} \in E$ that $\mathbf{r}$ is independent of $\mathbf{z}$. By the AMD property it thus holds for each $\mathbf{r} \in E\backslash\{\mathbf{0}\}$ that

$$\Pr[\mathsf{AMD.Decode}(\mathbf{z} + \mathbf{r}) \neq \perp] \leq \delta.$$

A union bound yields that

$$\Pr[\exists \mathbf{r} \in E\backslash\{\mathbf{0}\} : \mathsf{AMD.Decode}(\mathbf{z} + \mathbf{r}) \neq \perp] \leq \ell\delta.$$

Doing a change of variable, namely setting $\bar{\mathbf{z}} = \mathbf{z} + \mathbf{r}$, such that the quantification over $\mathbf{r} \in E\backslash\{\mathbf{0}\}$ becomes a quantification over $\bar{\mathbf{z}} \in \mathbf{z} + E\backslash\{\mathbf{0}\} = L\backslash\{\mathbf{z}\}$, gives us

$$\Pr[\exists \bar{\mathbf{z}} \in L\backslash\{\mathbf{z}\} : \mathsf{AMD.Decode}(\bar{\mathbf{z}}) \neq \perp] \leq \ell\delta.$$

Thus, every $\bar{\mathbf{z}} \in L\backslash\{\mathbf{z}\}$ will be rejected by $\mathsf{AMD.Decode}$, except with probability $\ell\delta$. Furthermore, $\mathbf{z} \in L$ will be accepted. Therefore, we can conclude that $\mathsf{LSSS}_3.\mathsf{RobustReconstruct}(\tilde{\mathbf{c}}) = \mathbf{s}$, except with probability $\ell\delta$. Consequently, LSSS_3 is $(t, \ell\delta)$-robust, which concludes the proof.

We will now state our main result for this section.

Theorem 7. *For any $\tau < \frac{1}{2}$, any $\tau < \sigma < 1 - \tau$, any $\rho < \sigma - \tau$ and any integer $\lambda > 0$ (the security parameter), we have that:*

- *For every $n > 0$ there exists an efficient n-player $(\tau n, 2^{-\lambda})$-robust secret sharing scheme LSSS with τn-privacy, σn-reconstructability and with rate ρ. The shares have size $\Theta(\log(n) + \lambda/n)$ and the secret has size $\Theta(n \cdot \log(n) + \lambda)$.*
- *There exists an infinite family $\{\mathsf{LSSS}_n\}$ of efficient n-player $(\tau n, 2^{-\lambda})$-robust secret sharing schemes with τn-privacy, σn-reconstructability and with rate ρ. The shares have size $\Theta(1 + \lambda/n)$ and the secret has size $\Theta(n + \lambda)$.*

We emphasize that even for non-robust ramp schemes, the rate ρ cannot be bigger than $\sigma - \tau$.

Proof. We shall instantiate Construction 3 with the list reconstructible secret sharing schemes provided by Lemma 9 and the AMD code given by Lemma 7. Let LSSS be an n-player $\mathbb{F}_q$-linear secret sharing scheme (from one of the two families in Lemma 9) with τn-privacy, σn-reconstructability and $(\tau n, \mathsf{poly}(n))$-list reconstructability, shares in $\mathbb{F}_q^m$ and rate ρ' with $\rho < \rho' < \sigma - \tau$. Recall that both constructions in Lemma 9 allow us to choose the parameter m arbitrarily large.

Now, we consider a $(q^{\rho' mn - 2\kappa}, q^{\rho' mn}, \delta')$-AMD code AMD, as provided by Lemma 7, with $\delta' = (\rho' mn - 2\kappa + 1)/q^{\kappa}$, where κ is to be determined later. By Theorem 7, this gives $(\tau n, \delta)$-robustness for

$$\delta \leq \mathsf{poly}(n) \cdot \delta' \leq \mathsf{poly}(n) \frac{\rho' mn}{q^{\kappa}} = \frac{m \cdot \mathsf{poly}(n)}{q^{\kappa}}.$$

Setting $\kappa = \lambda / \log(q) + \log(m \cdot \mathsf{poly}(n))$ gives $\delta \leq 2^{-\lambda}$. Finally, the rate of the scheme is given by

$$\frac{\rho' mn - 2\kappa}{mn} = \rho' - 2\frac{\lambda}{mn \log(q)} - 2 \cdot \frac{\log(\mathsf{poly}(n))}{mn} - 2\frac{\log(m)}{mn}.$$

By choosing the parameter m large enough, i.e. $m = \Omega(1+\lambda/(n\cdot\log(q)))$, this becomes bigger than ρ. Finally, for the first family provided in Lemma 9, such an LSSS exists for every length n and we have $q = O(n)$. Thus we can choose the parameter m as $m = \Theta(1 + \lambda/(n \cdot \log(n))$ and the shares for this instantiation have size $m \cdot \log(q) = \Theta(\log(n) + \lambda/n)$, whereas the secret has size $\rho mn \cdot \log(q) = \Theta(n \cdot \log(n) + \lambda)$. The second family provided by Lemma 9 is an infinite family for a constant q. Thus, for this instantiation the shares have size $m \cdot \log(q) = \Theta(1 + \lambda/n)$, whereas the secret has size $\rho mn \cdot \log(q) = \Theta(n + \lambda)$. This concludes the proof.

7 Acknowledgement

We would like to thank the anonymous reviewers of Eurocrypt 2015 for their helpful comments on this work. We would also like to thank Ignacio Cascudo and Irene Giacomelli for helpful discussions and comments.

References

1. Cascudo, I., Damgård, I., David, B., Giacomelli, I., Nielsen, J.B., Trifiletti, R.: Additively homomorphic uc commitments with optimal computational overhead. Manuscript (2014)
2. Cascudo Pueyo, I., Chen, H., Cramer, R., Xing, C.: Asymptotically good ideal linear secret sharing with strong multiplication over *Any* fixed finite field. In: Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 466–486 (2009)
3. Cevallos, A., Fehr, S., Ostrovsky, R., Rabani, Y.: Unconditionally-secure robust secret sharing with compact shares. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques. Lecture Notes in Computer Science, vol. 7237, pp. 195–208. Springer (2012)
4. Chen, H., Cramer, R., Goldwasser, S., De Haan, R., Vaikuntanathan, V.: Secure computation from random error correcting codes. In: Advances in Cryptology-EUROCRYPT 2007, pp. 291–310. Springer (2007)
5. Cramer, R., Damgård, I., Fehr, S.: On the cost of reconstructing a secret, or vss with optimal reconstruction phase. In: CRYPTO. pp. 503–523 (2001)
6. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. pp. 316–334 (2000)
7. Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings. pp. 471–488 (2008)
8. Damgård, I., David, B., Giacomelli, I., Nielsen, J.B.: Compact vss and efficient homomorphic uc commitments. Cryptology ePrint Archive, Report 2014/370 (2014), to appear in AsiaCrypt 2014
9. Druk, E., Ishai, Y.: Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In: ITCS. pp. 169–182 (2014)
10. Gammel, B.M., Mangard, S.: On the duality of probing and fault attacks. J. Electronic Testing 26(4), 483–493 (2010)
11. Garay, J.A., Ishai, Y., Kumaresan, R., Wee, H.: On the complexity of uc commitments. In: Advances in Cryptology–EUROCRYPT 2014, pp. 677–694. Springer (2014)
12. Guruswami, V., Indyk, P.: Linear-time encodable/decodable codes with near-optimal rate. IEEE Transactions on Information Theory 51(10), 3393–3400 (2005)
13. Guruswami, V., Rudra, A.: Explicit capacity-achieving list-decodable codes. In: STOC. pp. 1–10 (2006)
14. Guruswami, V., Wang, C.: Linear-algebraic list decoding for variants of reed-solomon codes. IEEE Transactions on Information Theory 59(6), 3257–3268 (2013)
15. Guruswami, V., Xing, C.: Optimal rate list decoding of folded algebraic-geometric codes over constant-sized alphabets. In: SODA. pp. 1858–1866 (2014)
16. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions (extended abstracts). In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA. pp. 12–24 (1989)

17. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007. pp. 21–30 (2007)
18. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: STOC. pp. 433–442 (2008)
19. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings. pp. 572–591 (2008)
20. Krawczyk, H.: Secret sharing made short. In: CRYPTO. pp. 136–146 (1993)
21. Mansour, Y., Nisan, N., Tiwari, P.: The computational complexity of universal hashing. In: Proceedings: Fifth Annual Structure in Complexity Theory Conference, Universitat Politècnica de Catalunya, Barcelona, Spain, July 8-11, 1990. p. 90 (1990)
22. Massey, J.L.: Some applications of coding theory in cryptography. In: Codes and Ciphers: Cryptography and Coding IV. pp. 33–47 (1995)
23. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, New York, NY, USA (2005)
24. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
25. Spielman, D.A.: Linear-time encodable and decodable error-correcting codes. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA. pp. 388–397 (1995)