Codes and Lattices in Cryptography Mini-Course **FCT-UNL, Spring 2023**

Notes 5: Information-theoretic cryptography

Lecturer: João Ribeiro

Introduction

These notes cover some more basic aspects of information-theoretic cryptography – protocols that are secure against computationally-unbounded adversaries.

5.1 Secret sharing

Consider the following scenario: We wish to distribute sensitive information s among n parties in a way that only select *authorized* subsets are able to reconstruct s, while *unauthorized* subsets of parties gain no information about the secret. For example, in order to ensure reliability and privacy, it could be useful to store a secret encryption key s across 10 servers in such a way that s can be reconstructed even if any 5 servers go offline, but also any powerful adversary that gains access into 4 or fewer servers learns no information about s. *Secret sharing*, a formalization of this concept, was introduced concurrently by Blakley [Bla79] and Shamir [Sha79].

A secret sharing scheme is composed of a probabilistic sharing procedure Share and a deterministic reconstruction procedure Rec. On input a secret s, we obtain the n shares of s

$$S = (S_1, \dots, S_n) \leftarrow \mathsf{Share}(s).$$

The reconstruction procedure receives as input a subset of shares, and attempts to reconstruct the secret. For a set $T \subseteq [n] = \{1, 2, \dots, n\}$, we define the projection $S_T = (S_i)_{i \in T}$. More formally, the following should hold.

Definition 5.1 (Threshold secret sharing scheme) *A pair* $(\mathsf{Share}, \mathsf{Rec})$ *is a* t-out-of-n secret sharing scheme *if the following two properties hold:*

- ***Correctness:*** *For any set of parties $T \subseteq [n]$ of size $|T| \geq t$ and any secret s we have that*
$$\Pr[\mathsf{Rec}(\mathsf{Share}(s)_T) = s] = 1.$$
- ***Privacy:*** *For any set of parties $T \subseteq [n]$ of size $|T| < t$ and any two secrets s and s' we have that $\mathsf{Share}(s)_T$ and $\mathsf{Share}(s')_T$ are identically distributed.*

Besides being interesting objects on their own, secret sharing schemes have also found important applications in other cryptographic tasks. For example, secret sharing is a fundamental building block of secure multiparty computation protocols. For more on this, see the excellent book of Cramer, Damgård, and Nielsen [CDN15].

5.1.1 Shamir's secret sharing scheme

Arguably the most well known and widely used secret sharing scheme is due to Shamir [Sha79] and is based on polynomial interpolation. Suppose that we wish to share a secret among n parties. Fix a prime power $q > n$ and n non-zero points $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q \setminus \{0\}$, where $\mathbb{F}_q$ denotes the finite field of order q. Fix also a threshold t, specifying the number of parties required to reconstruct the secret. On input a secret $s \in \mathbb{F}_q$, the sharing procedure $\mathsf{Share}(s)$ in Shamir's secret sharing scheme works as follows:

1. Sample coefficients $c_1, c_2, \dots, c_{t-1} \leftarrow \mathbb{F}_q$ and consider the polynomial
$$f(x) = s + \sum_{i=1}^{t-1} c_i x^i$$
over $\mathbb{F}_q$ of degree at most $t-1$;
2. Compute the i-th share S_i as the evaluation $S_i = f(\alpha_i)$ for $i \in [n]$.

To reconstruct the secret given t shares $S_{i_j} = f(\alpha_{i_j})$, perform Lagrange interpolation on these evaluations to recover f, and then compute $f(0)$ to recover the secret.

Theorem 5.1 *Shamir's scheme as described above is a t-out-of-n secret sharing scheme.*

Proof: Correctness follows from the fact that we can reconstruct the polynomial f, which has degree at most $t-1$, from any t evaluations via Lagrange interpolation. Then, we may recover $s = f(0)$. Regarding privacy, fix any $t-1$ shares $S_{i_1}, \dots, S_{i_{t-1}} \in \mathbb{F}_q$. It suffices to note that for each secret $s \in \mathbb{F}_q$ there is exactly one polynomial f_s of degree at most $t-1$ such that $f_s(0) = s$ and $f_s(\alpha_{i_j}) = S_{i_j}$ for all $j = 1, \dots, t-1$. Therefore, every secret is equally likely given $S_{i_1}, \dots, S_{i_{t-1}}$. ■

As we shall see next, Shamir secret sharing fits into a more general framework connecting certain linear codes to secret sharing schemes.

5.1.2 Secret sharing from MDS codes

In the previous notes we used Reed-Solomon codes to construct locally dense lattices with desirable properties. One useful property of Reed-Solomon codes is that they have optimal minimum distance for a given length and dimension – they are *maximum distance separable (MDS)*. We recall the general definition here.

Definition 5.2 (MDS code) *An $[n, r, d]_q$-code, where n is the length, r is the dimension, and d is the minimum distance, is said to be* maximum distance separable (MDS) *if $d = n - r + 1$.*

MDS codes satisfy the following useful property.

Lemma 5.1 *Suppose that $\mathcal{C}$ is an MDS $[n,r,d]_q$-code. Consider an arbitrary subset of coordinates $S \subseteq [n]$ of size r. Then,*

$$\mathcal{C}_S = \{c_S : c \in \mathcal{C}\} = \mathbb{F}_q^r.$$

In other words, for every vector $v \in \mathbb{F}_q^r$ and any set $S \subseteq [n]$ of size r there exists exactly one codeword $c \in \mathcal{C}$ such that $c_S = v$.

Proof: Fix an arbitrary ordered subset of coordinates $S = \{i_1, \dots, i_r\} \subseteq [n]$. For each vector $(\beta_1, \dots, \beta_r) \in \mathbb{F}_q^r$ we show that there exists exactly one codeword $c \in \mathcal{C}$ such that $c_{i_j} = \beta_j$ for all $j \in [r]$. Suppose not. Then, there exist distinct codewords $c, c' \in \mathcal{C}$ such that $(c - c')_{i_j} = 0$ for all $j \in [r]$. This implies that c and c' disagree on at most $n - r < d$ coordinates, contradicting the minimum distance of $\mathcal{C}$. ∎

We show how to build a secret sharing scheme from any MDS code, generalizing Shamir's secret sharing scheme. Suppose that we wish to share a secret $s \in \mathbb{F}_q$ among n parties with threshold t. Let $\mathcal{C}$ be an arbitrary MDS $[n+1, t, d]_q$-code, meaning that $d = (n+1) - t + 1$. We proceed as follows:

1. Sample a codeword $c = (c_0, \dots, c_n)$ uniformly at random from the subset

$$\mathcal{C}_s = \{c \in \mathcal{C} : c_0 = s\};$$

2. Set the i-th share as $S_i = c_i$ for $i \in [n]$.

To reconstruct the secret given shares S_{i_j} for $j \in [t]$, we find the unique codeword $c \in \mathcal{C}$ such that $c_{i_j} = S_{i_j}$ for all j. Then, we output c_0.

This approach to constructing secret sharing schemes is due to Massey [Mas95]. The proof of the following result follows the same lines as the proof of Theorem 5.1.

Theorem 5.2 *The scheme defined above is a t-out-of-n secret sharing scheme.*

Proof: Regarding correctness, it suffices to argue that revealing t symbols $S_{i_1}, \dots, S_{i_t}$ of a codeword determine the whole codeword. Indeed, if two distinct codewords $c, c' \in \mathcal{C}$ satisfy $c_{i_j} = c'_{i_j} = S_{i_j}$ for all $j \in [t]$, then c and c' differ in at most $n + 1 - t < d = n + 1 - t + 1$ coordinates, contradicting the minimum distance of $\mathcal{C}$.

To see privacy, fix any $t-1$ shares $S_{i_1}, \dots, S_{i_{t-1}}$. Since $\mathcal{C}$ has dimension t and is MDS, Lemma 5.1 guarantees that for every secret s there exists exactly one codeword $c \in \mathcal{C}$ such that $c_0 = s$ and $c_{i_j} = S_{i_j}$. This means that every secret is equally likely given $S_{i_1}, \dots, S_{i_{t-1}}$. ∎

Remark 5.1 We can obtain Shamir's secret sharing scheme as a special case of this MDS-based approach by choosing our MDS code $\mathcal{C}$ to be a Reed-Solomon code, which we defined and also used in the previous notes to construct locally dense lattices.

5.1.3 Important properties of these schemes

Besides satisfying the basic definition of secret sharing, the secret sharing schemes based on MDS codes above satisfy additional useful properties.

Linearity. Using the schemes above, we can perform linear operations on secrets by having each party locally perform operations on their shares. For example, suppose that $(S_1, \dots, S_n)$ are shares of s and $(S'_1, \dots, S'_n)$ are shares of s' for the same threshold t. Then, each party can locally compute $S''_i = S_i + S'_i$ over $\mathbb{F}_q$, and the resulting shares $(S''_1, \dots, S''_n)$ are a t-out-of-n sharing of $s'' = s + s'$.

Robustness. The schemes above allow correct reconstruction of the secret even when some parties act dishonestly and report incorrect shares, provided that enough shares are revealed and not too many shares are reported incorrectly

To see why, let $\mathcal{C}$ be an $[n+1, t, d]_q$-code and suppose that a codeword $c \in \mathcal{C}$ is corrupted by erasing n_e coordinates (where "e" stands for "erasures") and arbitrarily modifying n_f other coordinates (where "f" stands for "flips"), yielding a corrupted string z. Then, it is not hard to check that there is at most one codeword of $\mathcal{C}$ that could have originated z in this manner whenever $n_e + 2n_f < d$. In other words, in such a case we can correct the errors introduced in z and recover c.

Consider now a setting where $\mathcal{C}$ is MDS and t' parties reveal their shares, but $t'' \leq t'$ of them are dishonest and can reveal incorrect shares. This corresponds to the setting above with $n_e = n + 1 - t'$ and $n_f = t''$. Therefore, we correctly reconstruct the secret whenever

$$n_e + 2n_f = n + 1 - t' + 2t'' < d = n + 1 - t + 1.$$

This robustness constraint can be rewritten as

$$t'' < \frac{t' - t + 1}{2}.$$

A note on efficiency of robust reconstruction. While for every linear code there exists an efficient procedure that corrects erasures in codewords (i.e., missing shares), it is not straightforward to devise an efficient procedure that corrects flips for any given MDS code $\mathcal{C}$, which we would need to *robustly* reconstruct our secret in the secret sharing scheme defined by $\mathcal{C}$. But, for example, we do know of such efficient error-correcting procedures when $\mathcal{C}$ is a Reed-Solomon code (see [GRS22, Chapter 17]).

5.2 Randomness extraction and applications

High-quality sources of randomness are a fundamental resource not only in cryptography but also in many other settings in computer science, from running probabilistic algorithms to machine learning. When designing probabilistic algorithms or cryptographic protocols, we almost always assume access to an endless stream of independent and uniformly random bits. However, this assumption may not be reasonable. On the one hand, physical sources of randomness (such as rolling a dice or measuring temperature fluctuations and emissions of electromagnetic radiation) produce biased and correlated bits. On the other hand, even if we do start off with a stream of independent and uniformly random bits, an adversary may compromise this stream, either lowering the quality of the randomness being produced, or, at the very least, learning some information about the randomness.

These scenarios motivate the theory of randomness extraction, which studies efficient ways of generating independent and uniformly distributed random bits starting from defective or adversarial sources of randomness. Research in randomness extraction goes as far back as work of von Neumann [vN51] and has flourished into a deep theory of pseudorandomness with many connections to coding theory, combinatorics, and graph theory. For much more on this, see the excellent monograph of Vadhan [Vad12], from which these notes draw inspiration. Besides their initial motivation, randomness extractors have found many other applications, most notably in cryptography – we will see one such application.

5.2.1 Min-entropy and statistical distance

Before we discuss randomness extractors more formally we introduce some basic concepts from probability theory.

Definition 5.3 (Min-entropy and entropic sources) *Given a random variable X supported on a finite set, the* min-entropy of X, *denoted by $H_\infty(X)$, is defined as*

$$H_\infty(X) = -\log\left(\max_x \Pr[X = x]\right).$$

We say that X is a k-source *if $H_\infty(X) \geq k$.*

If X is a k-source, then no adversary can guess X with probability better than 2^{-k}. The notion of k-sources, which is quite general and especially useful for cryptography, was introduced in seminal work of Chor and Goldreich [CG88]. For intuition, it is okay to think of a k-source $X \in \{0,1\}^n$ as being a stream of n independent and uniformly random bits about which an adversary managed to collect $n-k$ bits of information – but we do not know exactly what has been collected. For example, the adversary may have learned $(X_i)_{i\in T}$ for some secret set $T \subseteq [n]$ of size $n-k$ of their choice.

Definition 5.4 (Statistical distance) *Given two random variables X and Y supported on a finite set, the* statistical distance between X and Y, *denoted by $\Delta(X;Y)$, is defined as*

$$\Delta(X;Y) = \sup_{\mathcal{S}} |\Pr[X \in \mathcal{S}] - \Pr[Y \in \mathcal{S}]| = \frac{1}{2}\sum_z |\Pr[X = z] - \Pr[Y = z]|.$$

When $\Delta(X;Y) \leq \varepsilon$ we say that X and Y are ε-close*, in which case we may write $X \approx_\varepsilon Y$.*

The statistical distance is a metric, and it is also known as *total variation distance.* Sometimes it is also useful to see it as (a scaling of) the ℓ_1 norm between the probability mass functions of X and

Y when represented as real-valued vectors. It has a nice characterization in terms of distinguishing games: We have that $X \approx_\varepsilon Y$ if and only if for any computationally-unbounded probabilistic algorithm D it holds that

$$|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| \leq \varepsilon,$$

i.e., no algorithm can distinguish between a sample from X and a sample from Y with probability better than $\frac{1+\varepsilon}{2}$. This is the information-theoretic analogue of the notion of computational indistinguishability that we have seen in previous notes.

5.2.2 Randomness extractors

There are many interesting types of randomness extractors. At a high-level, these are deterministic algorithms which are fed weak sources of randomness and whose respective output is close to uniformly random with respect to statistical distance. The dream randomness extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ would receive as input an arbitrary k-source X over $\{0,1\}^n$ for appropriately large min-entropy k and the output would satisfy $\mathsf{Ext}(X) \approx_\varepsilon U_m$ for small ε, where U_m is the uniform distribution over $\{0,1\}^m$. However, this object does not exist even in the extreme scenario where $k = n - 1$, $\varepsilon = 0.999$, and $m = 1$. To see this we look into the preimages of Ext. Without loss of generality, we may assume that $|\mathsf{Ext}^{-1}(0)| \geq 2^{n-1}$. Now, consider X uniformly distributed over $\mathsf{Ext}^{-1}(0)$. Then, X is a $(k = n - 1)$-source, but $\Pr[\mathsf{Ext}(X) = 0] = 1$, and so the output is clearly not close to uniformly random at all.

The argument above shows that we need to consider alternative randomness extraction models. One of the most basic and useful such models is *seeded randomness extraction*, where we receive not only a weak k-source X but also a uniformly random seed.

Definition 5.5 (Strong seeded extractor) *A function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is said to be a* (k, ε)-strong seeded extractor *if for any k-source X over* $\{0,1\}^n$ *it holds that*

$$\mathsf{Ext}(X, S), S \approx_\varepsilon U_{m+d}$$

where the seed S is uniformly distributed over $\{0,1\}^d$ *and U_{n+d} is uniformly distributed over* $\{0,1\}^{n+d}$.

A strong seeded extractor converts d bits of perfect randomness plus an n-bit weak source into $m + d \gg d$ bits of perfect randomness. These objects were first studied in this form by Nisan and Zuckerman [NZ96], but some implicit constructions of seeded extractors actually predate this work.

How good can seeded extractors be? A standard application of the probabilistic method shows that a uniformly random function $F : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is, with high probability, a (k, ε)-strong seeded extractor with output length $m = k - 2\log(1/\varepsilon) - O(1)$ and seed length $d = \log(n - k) + 2\log(1/\varepsilon) + O(1)$. It is also known that these parameters are optimal up to the $O(1)$ terms. For more on this, see [Vad12, Section 6.1.3].

Running probabilistic algorithms on weak sources of randomness. One may wonder what strong seeded extractors are that useful for, given that they already require a potentially short but already uniformly random seed to work. First, it turns out that such objects have important applications in cryptography even when the seed length is large, as we will see shortly. Second, if the seed is short enough then we may do away with it completely in certain settings! A classical example is when we wish to run probabilistic algorithms on weak sources of randomness.

Let A be an efficient probabilistic that yields the wrong answer on some input w with probability at most $\gamma = 0.9$ when it has access to m independent and uniformly random bits. Suppose that we also have a $(k, \varepsilon = 0.01)$-strong seeded extractor Ext computable by an efficient algorithm with nearly optimal seed length $d = O(\log n)$. Consider the modified algorithm A' that on input w and a k-source X computes the $2^d = \mathsf{poly}(n)$ candidate random strings $y_s = \mathsf{Ext}(X, s)$ for each seed $s \in \{0,1\}^d$ and outputs the majority of $(A(w, y_s))_{s\in\{0,1\}^d}$ (with ties broken arbitrarily). Then, it is not hard to see that A' is wrong on input w with probability at most $2(\gamma + \varepsilon) = 0.22$, it is only a $\mathsf{poly}(n)$-factor slower than A, *and it runs on an arbitrary k-source instead of independent and uniformly random bits!* For a generalization of this result, see [Vad12, Proposition 6.15].

5.2.3 Extractors from universal hashing

We proceed to construct a strong seeded extractor which has very small error ε and optimal output length m, but requires large seed length d. This extractor is based on families of functions satisfying a weak form of pairwise independence.

Let $\mathcal{H}$ be a family of functions $h : \{0,1\}^n \to \{0,1\}^m$ (such functions are usually called *hash* functions).

Definition 5.6 (2-universal hash function family) *A family of hash functions $\mathcal{H}$ as above is said to be* 2-universal *if for any distinct strings $x_1, x_2 \in \{0,1\}^n$ and a random function $H \leftarrow \mathcal{H}$ it holds that*

$$\Pr[H(x_1) = H(x_2)] \leq 2^{-m}.$$

An example of a 2-universal hash function family $\mathcal{H}$ is the set of all binary $m \times n$ matrices $A \in \mathbb{F}_2^{m\times n}$. This family has the desirable properties that (i) we can efficiently sample a uniformly random function $h(x) = Ax$ from $\mathcal{H}$, and (ii) given a description of h (the matrix A), we can efficiently compute $h(x)$ for any given x. To see why this family is 2-universal, note that

$$\Pr[Ax_1 = Ax_2] = \Pr[A(x_1 - x_2) = 0] = 2^{-m}$$

for any distinct $x_1, x_2 \in \mathbb{F}_2^n$ when $A \leftarrow \mathbb{F}_2^{m\times n}$.

One of the most widely useful results in the randomness extraction literature is the *Leftover Hash Lemma*, which states that the function $\mathsf{Ext} : \{0,1\}^n \times \mathcal{H} \to \{0,1\}^m$ given by

$$\mathsf{Ext}(x, h) = h(x)$$

with $\mathcal{H}$ an appropriate 2-universal hash function family is a strong seeded extractor with great parameters. An early form of this result was proved by Bennett, Brassard, and Robert [BBR88], and the current lemma statement is due to Impagliazzo, Levin, and Luby [ILL89]. For example, the Leftover Hash Lemma is key for arguing the security of Regev's LWE-based public-key encryption scheme, as detailed in previous notes. We will also see another application of this result.

Lemma 5.2 (Leftover Hash Lemma) *Fix integers n and $k \leq n$ and any $\varepsilon > 0$. Let $\mathcal{H}$ be a 2-universal hash function family with input length n and output length $m = k - 2\log(1/\varepsilon)$. Then, the function $\mathsf{Ext} : \{0,1\}^n \times \mathcal{H} \to \{0,1\}^m$ given by $\mathsf{Ext}(x,h) = h(x)$ is a $(k, \varepsilon/2)$-strong seeded extractor.*

Proof: This proof follows that of [Vad12, Theorem 6.18]. Fix an arbitrary k-source X on $\{0,1\}^n$. We wish to show that

$$H, H(X) \approx_{\varepsilon/2} H, U_m, \tag{5.1}$$

where H is uniformly distributed over $\mathcal{H}$ and U_m is uniformly distributed over $\{0,1\}^m$ and independent of H.

Denote the probability mass functions of $(H, H(X))$ and (H, U_m) by $P_{H,H(X)}$ and P_{H,U_m}, respectively. Then, Equation (5.1) is equivalent to

$$\|P_{H,H(X)} - P_{H,U_m}\|_1 \leq \varepsilon, \tag{5.2}$$

where $P_{H,H(X)}$ and P_{H,U_m} are seen as real-valued vectors. In order to prove Equation (5.2), we first replace the ℓ_1 norm by the ℓ_2 norm via Cauchy-Schwarz. Then, we relate the ℓ_2 norm to the *collision probability* of $(H, H(X))$, and use the 2-universal property of $\mathcal{H}$ to bound this collision probability. By Cauchy-Schwarz, we have that

$$\|P_{H,H(X)} - P_{H,U_m}\|_1 \leq \sqrt{|\mathcal{H}| \cdot 2^m}\|P_{H,H(X)} - P_{H,U_m}\|_2. \tag{5.3}$$

We expand the ℓ_2 norm as

$$\begin{aligned}
\|P_{H,H(X)} - P_{H,U_m}\|_2^2 &= \sum_{h,y} |\Pr[H = h, H(X) = y] - \Pr[H = h] \cdot \Pr[U_m = y]|^2 \\
&= \sum_{h,y} \left|\Pr[H = h, H(X) = y] - \frac{1}{|\mathcal{H}| \cdot 2^m}\right|^2 \\
&= \sum_{h,y} \left(\Pr[H = h, H(X) = y]^2 - \frac{2 \cdot \Pr[H = h, H(X) = y]}{|\mathcal{H}| \cdot 2^m} + \frac{1}{(|\mathcal{H}| \cdot 2^m)^2}\right) \\
&= \sum_{h,y} \Pr[H = h, H(X) = y]^2 - \frac{1}{|\mathcal{H}| \cdot 2^m}. \qquad (5.4)
\end{aligned}$$

Now, notice that

$$\sum_{h,y} \Pr[H = h, H(X) = y]^2 = \Pr[(H, H(X)) = (H', H'(X'))], \tag{5.5}$$

where (H', X') are identically distributed to but independent of (H, X). In other words, the quantity $\sum_{h,y} \Pr[H = h, H(X) = y]^2$ is the *collision probability* of $(H, H(X))$. Observe that

$(H,H(X))=(H',H'(X'))$ holds if and only if $H'=H$ and either $X'=X$ or $X'\neq X$ but $H(X')=H(X)$. Therefore,

$$\Pr[(H,H(X))=(H',H'(X'))]=\Pr[H'=H,X'=X]+\Pr[H'=H,X'\neq X,H(X')=H(X)]. \tag{5.6}$$

We bound each term on the right hand side separately. First, note that

$$\Pr[H'=H,X'=X]=\Pr[H'=H]\cdot\Pr[X'=X]\leq\frac{1}{|\mathcal{H}|\cdot 2^k}=\frac{\varepsilon^2}{|\mathcal{H}|\cdot 2^m}, \tag{5.7}$$

where we have used the fact that $\Pr[X'=X]=\sum_x\Pr[X=x]^2\leq 2^{-k}$ since X is a k-source and the choice of $m=k-2\log(1/\varepsilon)$ for the rightmost equality. Second, we have that

$$\Pr[H'=H,X'\neq X,H(X')=H(X)]\leq\frac{1}{|\mathcal{H}|}\cdot\Pr[H(X')=H(X)|X'\neq X]\leq\frac{1}{|\mathcal{H}|\cdot 2^m}, \tag{5.8}$$

where the righmost inequality holds because $\mathcal{H}$ is 2-universal. Combining Equations (5.5) to (5.8) yields

$$\sum_{h,y}\Pr[H=h,H(X)=y]^2\leq\frac{1+\varepsilon^2}{|\mathcal{H}|\cdot 2^m}.$$

Plugging this bound into Equation (5.4) shows that

$$\|P_{H,H(X)}-P_{H,U_m}\|_2^2\leq\frac{\varepsilon^2}{|\mathcal{H}|\cdot 2^m}.$$

Finally, combining this bound with Equation (5.3) and recalling that it is sufficient to show Equation (5.2) concludes the proof. ■

Parameters of the Leftover Hash Lemma extractor. The seed length d of the Leftover Hash Lemma extractor discussed above corresponds to the description size of the hash function family $\mathcal{H}$. For our example above where $\mathcal{H}=\{h(x)=Ax : A\in\mathbb{F}_2^{m\times n}\}$, the resulting seed length would be $d=m\cdot n$. Other choices of $\mathcal{H}$ lead to smaller seed length d while retaining efficient sampling and computation of the hash output $h(x)$. One possibility is to consider $\mathcal{H}$ to be the family of functions $h_s(x)=[s\cdot x]_m$, where $s\in\mathbb{F}_2^n$ and $s\cdot x$ is computed over $\mathbb{F}_{2^n}$, and $[s\cdot x]_m$ denotes truncation to the first m bits of $s\cdot x$ when seen as a vector in $\mathbb{F}_2^n$. Choosing this family leads to seed length $d=n$, which is optimal among 2-universal hash function families. While the seed length is necessarily large (compare with the optimal seed length $d\approx\log n+2\log(1/\varepsilon)$), the statistical error ε of the extractor can be made very small and the output length $m=k-2\log(1/\varepsilon)$ is optimal up to a constant term.

Seeded extractors beyond the Leftover Hash Lemma. As the Leftover Hash Lemma extractor necessarily requires large seed length $d\geq n$, a line of work has studied efficient constructions of strong seeded extractors with much smaller seed length, culminating in the nearly-optimal construction of Guruswami, Umans, and Vadhan [GUV09] based on Parvaresh-Vardy codes. Generally speaking, there is a close connection between seeded extractors and *list-decodable codes*, and some works such as [Tre01, TZ04, GUV09] have exploited this connection. In particular, it is known [TZ04] that strong seeded extractors with output length $m=1$ are equivalent to list-decodable codes

5.2.4 Privacy amplification

Consider the following scenario: Alice and Bob hold a pre-shared n-bit secret key $SK \in \{0,1\}^n$ that they wish to use to communicate securely over a public channel – for example, using the one-time pad encryption scheme we have seen before. However, an eavesdropper, Eve, managed to get some limited partial information about Alice's and Bob's secret key SK. Maybe Eve managed to peek into the key exchange procedure, or perhaps Eve gained some form of access to the Alice's and Bob's devices storing the key.

We do not know exactly what Eve learned about the secret key SK. A reasonable way to model this is to assume that Eve learned side information $f(SK)$ for some bounded-output function $f : \{0,1\}^n \to \{0,1\}^\ell$ of their choice, for some parameter ℓ. For example, f could output any choice of ℓ bits of SK, but more sophisticated functions are also allowed. The secret key SK was originally uniformly distributed over $\{0,1\}^n$, but this is no longer the case from Eve's viewpoint after they learn $f(SK)$. What is still true, however, is that, with high probability, SK remains *hard to guess* even given $f(SK)$, provided that ℓ is not extremely large. In other words, SK conditioned on the value of $f(SK)$ still has decent min-entropy (with high probability over the value of $f(SK)$). Can Alice and Bob hope to recover from this situation and generate a uniform secret key via the public communication channel only, without the need to schedule a private meeting once again to share a new secret key?

The setting above corresponds to the classical cryptographic problem of *privacy amplification*, first studied by Bennett, Brassard, and Robert [BBR88]. Alice and Bob share an arbitrary k-source $X \in \{0,1\}^n$ (i.e., $H_\infty(X) \geq k$) and wish to generate a shared uniformly random secret key Z. To this end, Alice is allowed to send a message S to Bob via the public channel, and both Alice and Bob compute their new m-bit secret key as $Z = F(X,S)$ for some function $F : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$. Alice and Bob wish to ensure that Eve learns (almost) nothing about Z by seeing the message S being sent over the public channel – otherwise, the new key Z is useless. This can be formalized by requiring that

$$F(X,S), S \approx_\varepsilon U_m, S, \tag{5.9}$$

where U_n is uniformly random over $\{0,1\}^m$ and independent of S and ε is a small statistical error. In words, Eve cannot, even with knowledge of S, distinguish between the actual key $F(X,S)$ and an independent uniformly random string.

Not coincidentally, Equation (5.9) suggests a straightforward application of strong seeded extractors! Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (k,ε)-strong seeded extractor. Then, consider the following privacy amplification protocol:

1. Alice samples $S \leftarrow \{0,1\}^d$ and sends S to Bob through the public channel;
2. Alice and Bob both compute the new secret key $Z = \mathsf{Ext}(X,S)$.

Then, a direct application of the definition of a strong seeded extractor shows that Equation (5.9) holds, as desired.

Note that such privacy amplification protocols are interesting even when the seed length d of Ext is large. But, nevertheless, reducing the communication complexity of these protocols (i.e., the seed length d) provides yet another motivation for designing seeded extractors with small seed length.

References

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, April 1988.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[GRS22] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*. 2022. Draft available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *J. ACM*, 56(4), jul 2009. Preliminary version in CCC 2007.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC 1989)*, page 12–24, New York, NY, USA, 1989. Association for Computing Machinery.

[Mas95] James L. Massey. Some applications of coding theory in cryptography. *Codes and Ciphers: Cryptography and Coding IV*, pages 33–47, 1995.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. Preliminary version in STOC 1993.

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

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

[TZ04] Amnon Ta-Shma and David Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2004. Preliminary version in STOC 2001.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012. Draft available at https://people.seas.harvard.edu/~salil/pseudorandomness/.

[vN51] John von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 12(36):768–770, 1951.