Randomness in Cryptography May 2, 2013

Lecture 16: Special Purpose Extractors

Lecturer: Yevgeniy Dodis *Scribe: Zahra Jafargholi*

In previous lectures we studied extractors and strong extractors, today we study several randomness extractors, each with a special property. In this note $\mathcal{I} \in \{0,1\}^d$ is the uniform seed, $W \in \{0,1\}^n$ is the week source, $\mathbf{H}_\infty(W) \geq k$, $R \in \{0,1\}^m \approx_\varepsilon U_m$ is the extracted randomness from W.

1 Randomness extractors and Entropic security

In previous lectures we defined a randomness extractor as follow,

Definition 1 Function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is (k,ε)- *extractor* if for any distribution $W \in \{0,1\}^n$, with $\mathbf{H}_\infty(W) \geq k$

$$\mathsf{SD}(\mathsf{Ext}(W;\mathcal{I}), U_m) \leq \varepsilon.$$

$\diamondsuit$

If $(\mathcal{I}, \mathsf{Ext}(W;\mathcal{I}) \approx_\varepsilon (\mathcal{I}, U_m)$ we say Ext is a *strong* extractor. We saw that if $H = \{h_i : \{0,1\}^n \to \{0,1\}^m | \forall i \in \{0,1\}^d\}$ is a universal hash family and $m \leq k - 2\log(1/\varepsilon)$ then $\mathsf{Ext}(X;\mathcal{I}) = h_\mathcal{I}(X)$ is (k,ε)-strong extractor (LHL). We also saw the LHL is pretty robust. If $\mathcal{I}$ has entropy deficiency c then we can extract $m = k - c - 2\log(1/\varepsilon)$ bits.

Next we define (k,ε)-entropically secure and (k,ε)- indistinguishable maps:

Definition 2 The probabilistic map $S()$ *hides all functions of* W with leakage ε if for every adversary $\mathcal{A}$, there exists an adversary $\mathcal{A}_*$ such that for all functions $f : \{0,1\}^* \to zo^*$,

$$|\Pr[\mathcal{A}(S(W)) = f(W)] - \Pr[\mathcal{A}_*() = f(W)]| \leq \varepsilon$$

$\diamondsuit$

In other words, predicting $f(W)$ is nearly as hard with $S(W)$ as it is without $S(W)$. If $S()$ hides all functions of W, for all k -sources W, we say $S()$ is (k,ε)*-entropically secure.*

Definition 3 The probabilistic map $S()$ is (k,ε)*-indistinguishable* if for all pairs of k-sources W_1 and W_2, $S(W_1) \approx_\varepsilon S(W_2)$ $\diamondsuit$

In particular all such output distributions of S are ε-close to $Y(U_n)$, so all (k,ε)-extractors are$(k,2\varepsilon)$-indistinguishable.

Theorem 1 *If $S()$ is (k,ε)-entropically secure, then it is $(k-1,4\varepsilon)$- indistinguishable. Conversely if $S()$ is (k,ε)-indistinguishable then it is $(k+2,8\varepsilon)$-entropically secure.*

To give an intuition as why indistinguishablity implies entropic security , we give an example showing that if $S()$ is $(k-1,\varepsilon)$-indistinguishable then $S()$ is $(k,\varepsilon/2)$-entropically secure for all balanced predicates.

Suppose that $g()$ is a balanced predicate for distribution X, that is $\Pr[g(X)=0] = \Pr[g(X)=1] = 1/2$, and that $\mathcal{A}$ is an adversary contradicting entropic security for min-entropy $k = \mathbf{H}_\infty(X)$, that is $\Pr[\mathcal{A}(S(X)) = g(X)] = 1/2 + \varepsilon$. For $b \in \{0,1\}$, let X_b be the distribution of X conditioned on $g(X) = b$. The adversary's advantage over random guessing in distinguishing $S(X_0)$ and $S(X_1)$ is ε. However, the same advantage is also a lower bound for the statistical difference. We get,

$$\begin{aligned}\frac{1}{2}+\varepsilon &= \Pr[\mathcal{A}(S(X)) = g(X)]\\ &= \Pr[b \overset{r}{\leftarrow} \{0,1\} : \mathcal{A}(S(X_b)) = b] \leq \frac{1}{2} + \frac{1}{2}\mathsf{SD}(S(X_0), S(X_1)),\end{aligned}$$

and so the distance between $S(X_0)$ and $S(X_1)$ is at least 2ε. Note that since $g(X)$ is a balanced predicate, both X_0 and X_1 have min-entropy $k-1$ and $\mathsf{SD}(S(X_0), S(X_1)) \geq 2\varepsilon$ contradicts with $S()$ being $(k-1,\varepsilon)$ -indistinguishable.

Corollary 1 *(k,ε)-extractors hide all functions for sources with min-entropy $k+2$.*

This means to design an entropically secure map with some special functionality it is enough to design a special purpose extractor having this functionality.

2 Adding Invertability: Entropically Secure Encryption

As a first special property, we consider invertible extractors: namely, the extractors input W should be reproducible from its output $\mathsf{Ext}(W;\mathcal{I})$ and the seed $\mathcal{I}$ by some (efficient) procedure $\mathsf{Inv}(\cdot,\cdot)$. Notice that invertibility of the extractors and the fact that $\mathcal{I}$ is independent from W imply that the output length m of the extractor must be at least n. Since these $m \geq n$ bits are (nearly) uniform, and W has only k bits of entropy, we see that the remaining at least $n-k$ bits of the entropy must come from the seed $\mathcal{I}$. In fact, the lower bound on extractors easily implies that $|\mathcal{I}| = d \geq n - k + 2\log(1/\varepsilon) - O(1)$. We will shortly see how we can match this bound.

Also notice that an invertible extractor cannot be strong. Indeed, in this case the value W should be obtainable from an almost uniform string $\mathsf{Ext}(W;\mathcal{I}) \circ \mathcal{I}$, which means that W should be close to a fixed distribution $\mathsf{Inv}(U_m, \mathcal{I})$. However, this is impossible, since W could be any distribution of min-entropy k (and we assume $k < n$). However, some part $\mathcal{I}_p$ of the seed $\mathcal{I} = (\mathcal{I}_s, \mathcal{I}_p)$ could indeed be independent from the output. (We call such extractors semi-strong.) Unfortunately, an extension of the previous argument still implies that the "secret part" of the seed must be long: $|\mathcal{I}_s| \geq n - k + 2\log(1/\varepsilon) - O(1)$. However, having the public part $\mathcal{I}_p$ enables us to achieve simpler constructions.

Application: Entropically Secure Encryption. Now assume we have Ext which is a (k, ε)-extractor with invertability property. We can set $C = \mathsf{Ext}(W, \mathcal{I})$ to be the ciphertext for message W, since Ext is invertable we can recover W from C and $\mathcal{I}$ and as long as W has min-entropy k, C hides all functions of W which gives us the notion of (k, ε)-*entropically secure encryption*. This notion is weaker than Shanon's security but it will allow us to have shorter key length. Here we achieve key length of slightly more than $n - k$ bits. Intuitively, to hide the plaintext of n bits we need n bits of randomness, here we take k of those bits from the randomness of the message itself and the rest $(n - k)$ bits we get from the secret key. We also remark that "semi-strong" invertible extractors with seed $(\mathcal{I}_s, \mathcal{I}_p)$ corresponds to probabilistic encryption, where the parties only share the "secret part" $\mathcal{I}_s$, while the "public part" $\mathcal{I}_p$ is sent together with the ciphertext $\mathsf{Ext}(W; (\mathcal{I}_s, \mathcal{I}_p))$. Thus, here we only care about minimizing the secret part $\mathcal{I}_s$.

Constructions. One idea is to construct invertible extractors from good expander graphs, which in fact mix in one step. A bit more formally, assume we have a 2^d-regular expander graph G on 2^n vertices V with the property that for any subset T of 2^k vertices, picking a random vertex w of T and taking a random neighbor v, we obtain an almost uniform distribution on V (say, within distance ε from U_n). Since any k-source W is known to be a convex combination of uniform distributions on some subsets T of size 2^k, it is obvious that such extractor graphs immediately yield a $(n, k, m = n, \varepsilon)$-extractor, where the source W defines the original vertex v and the seed $\mathcal{I}$ specifies which neighbor v of w to take. To see the invertibility of this scheme, we need to ensure that it is possible to label the edges of G in such a way that knowing the index i and the i-th neighbor v of a vertex w under this labeling, we can recover w back. We call such natural labelings *invertible*. Luckily, Hall's marriage theorem implies that every 2^d-regular graph has an invertible labeling, although this labeling does not have to be efficient. In all our examples, however, the corresponding labeling will indeed be efficient. It remains to construct such extractor with the smallest degree 2^d, since d will translate to the length of the seed (i.e., the secret key).

Here we give three different invertible (n, k, n, ε)-extractors:

1. Ramanjun expander graphs. is obtained by using (optimal) Ramanujan expander graphs, which indeed have $d = n - k + 2\log(1/\varepsilon) + O(1)$. However, the constructions of such optimal graphs is relatively complex.

2. "Sparse One-Time Pad": Using $2\log(n/\delta)$ bits $(\mathcal{I})$, we sample δ-biased X of length n and we get $W \oplus X \approx_\varepsilon U_n$ where $\varepsilon = \delta 2^{(n-k)/2}$. This extractor can achieve seed length $d = n - k + 2\log(1/\varepsilon) + 2\log n + O(1)$ when using optimal δ-biased sets, but such sets are still non-trivial to construct.

3. "LHL-based semi-strong extractor": the seed $\mathcal{I}$ consists of a secret part $\mathcal{I}_s$ of length $\ell = n - k + 2\log(1/\varepsilon) + O(1)$, which is just a random point $x \in \{0,1\}^\ell$, and a public part $\mathcal{I}_p$, which samples a random hash function $h_\mathcal{J}$ from any family $\mathcal{H} = \{h_j : \{0,1\}^\ell \to \{0,1\}^n\}_j$ of XOR-universal hash functions. The extractor is $\mathsf{Ext}(W; (X, \mathcal{J})) = h_\mathcal{J}(X) \oplus W$.

To argue the correctness of the last construction we use a variant of LHL.

Lemma 2 ([6] LHL$'$: One-Time Pad Extractor) *If $\mathcal{H} = \{h_j : \{0,1\}^\ell \to \{0,1\}^n\}_j$ is XOR-universal and X and W are two independent distributions on $\{0,1\}^\ell$ and $\{0,1\}^n$, respectively, satisfying $\mathbf{H}_\infty(X)+\mathbf{H}_\infty(W) \geq n+2\log(1/\varepsilon)+1$, then $\mathsf{SD}(\mathcal{J}, h_{\mathcal{J}}(X) \oplus W), (\mathcal{J}, U_n) \leq \varepsilon$.* $\diamondsuit$

The third construction follows from this result by taking $\ell = n - k + 2\log(1/\varepsilon) + 1$ and having X to be uniform on $\{0,1\}^\ell$. Applied to the $a \cdot x$ XOR-universal family, we get the following very simple entropically-secure encryption scheme: view $GF[2^\ell]$ as a subset of $GF[2^n]$ (where XOR coincides with field addition) and encrypt message w by sending $(a, a \cdot x + w)$, where $a \in GF[2^n]$ is a public randomizer, and $x \in GF[2^\ell]$ is the secret key.

3 Adding Collision-Resistance: Perfectly One-Way Hash Functions

The next special property we consider is (computational) collision-resistance. We say that an extractor Ext is collision resistant, if it is (computationally) infeasible to find two inputs $(w, i) \neq (w, i)$ such that $\mathsf{Ext}(w; i) = \mathsf{Ext}(w; i)$. A small technicality here is that the definition of collision-resistance against non-uniform adversaries requires an extra key (key) to be generated at the beginning of the game, so we will do the same: i.e., our extractor will also have "collision-resistant" key, in addition to its seed i. Also, we will consider only strong extractors, which output their seed i as part of their output. This means that: (a) an output (z, i) can be verified by presenting w alone (by checking if $z = \mathsf{Ext}_{\mathsf{key}}(w; i)$); and (b) the definition of collision resistance states that, for a random key key, it is hard to find $w \neq w$ and a seed i such that $\mathsf{Ext}_{\mathsf{key}}(w; i) = \mathsf{Ext}_{\mathsf{key}}(w'; i)$. Combined, this means that the value (z, i) is a "commitment" to w, which can be "opened" by presenting w alone! Of course, the price we pay for such a nice decommitment procedure comes from a weaker privacy guarantee: since extractors are entropically secure, we can only say that the "commitment" value (z, i) hides all functions of w (for any key key, but for a random seed i), as long as w has high entropy. Thus, publishing (z, i) allows anybody to test (without either false positives or false negatives!) whether or not some input w is equal to the "correct" secret input w, and yet without leaking any particular function of w. We also notice that entropic security/indistinguishability is all we need in this application (i.e., we do not care if the extractors output is close to the uniform as opposed to some other fixed distribution). We call such (k, ε)-indistinguishable (but not necessarily extractor) maps, collision-resistant maps (k, ε)-privately binding.

Application: Perfectly one-way hash functions.
DEFINITION 4 ([3]) An ensemble of keyed randomized functions $\mathcal{H} = \{H_{\mathsf{key}} | \mathsf{key} \in \mathcal{K}_n, n \in N\}$with domain $\{0,1\}^n$ (where n is the security parameter), key space $\mathcal{K}_n$ and randomness space $\mathcal{R}_n$ is (k, ε)-*perfectly one-way* if there is a polynomial-time verification algorithm Ver such that

- For all keys $\mathsf{key} \in \mathcal{K}_n$, inputs $w \in \{0,1\}^n$, and strings $i \in \mathcal{R}_n$, $\mathsf{Ver}(\mathsf{key}, w, H_{\mathsf{key}}(w;i)) = \textsc{acc}$.

- For any efficient adversary $\mathcal{A}$, the probability over $\mathsf{key} \in K_n$ that $\mathcal{A}(k)$ outputs (w, w', y) satisfying $w \neq w'$ and $\mathsf{Ver}(\mathsf{key}, w, y) = \mathsf{Ver}(\mathsf{key}, w', y) = \textsc{acc}$ is negligible in n.

- For all keys $\mathsf{key} \in \mathcal{K}_n$, the randomized map $W \mapsto H_{\mathsf{key}}(W;\mathcal{I})$ is (t,ε)-entropically secure: for all t-sources W, the value $H_{\mathsf{key}}(W;\mathcal{I})$ hides all functions of W with leakage ε, when $\mathcal{I}$ is chosen at random from $\mathcal{R}_n$.

$\diamondsuit$

As we can see from Theorem 1, a $(k+2,\varepsilon)$-privately binding map E is (k,ε)-entropically secure, and thus immediately gives a (k,ε)-POWHF H of the following form: $H_{\mathsf{key}}(w;i)$ outputs $(E_{\mathsf{key}}(w;i), i)$, and $\mathsf{Ver}(\mathsf{key}, w, (z, i))$ accepts if and only if $E_{\mathsf{key}}(w;i) = z$. Here we construct a perfectly one-way hash function. We start from a yet another variant of the leftover hash lemma. It states that combining a pairwise independent hash function h with an arbitrary function f (of small enough output domain) yields an indistinguishable map: that is, the output may not be random, but it will look the same for all input distributions of sufficiently high entropy.

Lemma 3 ([7] LHL$''$: Composing with arbitrary function) *Let $f : \{0,1\}^N \to \{0,1\}^m$ be an arbitrary function. If $\mathcal{H} = \{h_i : \{0,1\}^n \to \{0,1\}^N\}_i$ is a family of pairwise independent hash functions and W is a k-source over $\{0,1\}^n$ with $tk \geq m + 2\log(1/\varepsilon) + 1$, then $\mathsf{SD}(I, f(h_{\mathcal{I}}(W))), (I, f(U_N)) \leq \varepsilon$.* $\diamondsuit$

Contrary to intuition, this statement does *not* follow directly from the usual LHL, since the hash function $h_{\mathcal{I}}$ might be length-increasing (i.e., there is no constraint on N), and thus the intermediate distribution $h_{\mathcal{I}}(W)$ might not be close to uniform. On the other hand, we do need a slightly stronger assumption on our hash family that universality: *pairwise independence*. Namely, for any $x \neq y$, the value $h_{\mathcal{I}}(x)$ and $h_{\mathcal{I}}(y)$ should be truly random and independent from each other (i.e., $(h_{\mathcal{I}}(x), h_{\mathcal{I}}(y)) \equiv (U_N, U_N)$). Constructively, one can turn the $a \cdot x$ construction of universal hash functions into that of pairwise independent hash function, by also sampling a random $b \in GF[2^{\max(n,N)}]$ together with a (i.e., $i = (a,b)$), and setting $h_i(x)$ to be the first N bits of $a \cdot x + b$.

We can now apply this lemma as follows. The function f will be a (computationally) collision-resistant hash function C_{key} whose output length $m \leq k - 2\log(1/\varepsilon) - 1$ (and whose choice will fix the collision-resistant key key). As for the family of pairwise independent hash functions, we will take a family of pairwise independent *permutations*: here $h_{\mathcal{I}}(x)$ and $h_{\mathcal{I}}(y)$ look like a pair of random *distinct elements*. Although they are technically not pairwise independent *functions*, they are 2^{-n}-close to them, which will not affect Lemma 3. In the $a \cdot x + b$ construction, this is achieved by restricting a to be non-zero.

We now get the following construction of a (k,ε)-privately binding map: $E_{\mathsf{key}}(w,i) = C_{\mathsf{key}}(h_i(w))$. Its (k,ε)-indistinguishability follows directly from Lemma 3, while its collision-resistance follows from that of C_{key} and the fact that h_i is a *permutation*, for any i. Also notice that if the collision-resistant function C_{key} is regular, i.e. $C_{\mathsf{key}}(U_N) \equiv U_\ell$, then we indeed get a collision-resistant randomness extractor. See [3] for a construction of such regular collision-resistant functions.

4 Adding locally computablity: Key derivation in Bounded storage model

In Bounded Storage Model, Alice and Bob share a short random key K. In this model a large random (or at least with high entropy) string X of length N is broadcast to both Alice and Bob. Eve is allowed to store any function $Z = f(X)$ of length γN, for some $\gamma < 1$. The fact that Eve cannot store the entire X, makes X to look imperfect to Eve. Here the goals are:

1. Key agreement: to extract randomness R from X, using K. In particular, we want R to be much longer than K and look random to Eve, regardless of the storage function f.
2. Key reuse: to Keep using K with new X.
3. Everlasting security: R should look random even if K is leaked later.

The first simple solution is to apply a strong extracter to X using K as the seed. The problem with this approach is that Alice and Bob need to read the entire X to extract from it, which is not possible since we are assuming X is too big even for Eve to read it.

In [14], K consists of k random indices $i_1, i_2, ..i_k \in [N]$ and we use these indices to sample k bits from X and let $w = X[i_1]...X[i_k]$ and then Let R be the parity of bits of w. [14] shows that R is secure as long as $\gamma < 1/5$ and k is large enough. Although this method is inefficient but it does achieve all three goals.

We next describe another method known as *sample-then- extract* [15]. The idea is to parse K into two keys K_s, K_e, sampling key and extraction key. Then we use K_s to sample a small subset of bits W from X and then use any good strong extractor with seed K_e to extract R from W. With a "good" K_s, W will have high entropy from Eve's point of view and therefore R will be secure. With an optimal sampler and extractor we can have $|K| = O(\log N + \log(1/\varepsilon))$ and extract m bits by reading $O(m)$ bits W from X.

5 Adding noise tolerance: Fuzzy extractors and Secure sketches

Fuzzy extractors were introduced in [5] to cope with keys derived from biometrics and other noisy measurements. The idea is to extract a random key R from the biometric W together with the error-correction information P, such that R is random even given P, but R *can be recovered from any noisy variant W' of W using P*. Equivalently, it gives a one-round secret key agreement protocol over a public channel, where the transmission of P allows the communicating parties to agree on the same key R, despite initially receiving different versions of some noisy data. Formally, assuming W lives in a metric space $\mathcal{M}$ equipped with a distance function $\mathsf{dist}(\cdot,\cdot)$, Definition 5 ([5]) An *$(\mathcal{M}, k, m, t, \varepsilon)$-fuzzy extractor* is a given by two efficient procedures $(\mathsf{Gen}, \mathsf{Rep})$.

1. Gen is a probabilistic generation procedure, which on input $w \in \mathcal{M}$ outputs an "extracted" string $R \in \{0,1\}^m$ and a public string P, such that for any k-source W, if $(R, P) \leftarrow \mathsf{Gen}(W)$, then $\mathsf{SD}(R, P), (U_m, P) \leq \varepsilon$.

2. Rep is a deterministic reproduction procedure which allows one to recover R from the corresponding public string P and any vector w' close to w: for all $w, w' \in \mathcal{M}$ satisfying $\mathsf{dist}(w, w') \leq t$, if $(R, P) \leftarrow \mathsf{Gen}(w)$, then we have $\mathsf{Rep}(w', P) = R$.

The *entropy loss* of a fuzzy extractor is defines as $k - m$. $\diamondsuit$

While the above definition is general enough to deal with arbitrary metrics $\mathcal{M}$, in the following we will restrict ourselves with $\mathcal{M} = \mathcal{F}^n$, where $\mathcal{F}$ is a finite set equipped with the usual Hamming metric: $\mathsf{dist}(w, w')$ is the number of positions i where $w_i \neq w'_i$. (See [5] for constructions over different metrics.) In this case we call the corresponding extractor $(n, k, m, t, \varepsilon)$-fuzzy. The binary case $\mathcal{F} = \{0,1\}$ will be of special importance.

Secure Sketch. Notice that in the "error-free" case ($t = 0$) strong extractors achieve this functionality, by setting $P = \mathcal{I}$. A natural way to extend strong extractors into fuzzy extractors is to publish, as part of P, some "error-correction information" S about W, which will allow to recover W from W' and S, after which we can apply a strong extractor to this recovered W. A formalization of this idea leads to a new primitive of independent interest called *secure sketch* [5].

Definition 6 ([5]) A *(n, k, k', t)-secure sketch* (over $\mathcal{F}$) is a pair of efficient (possibly randomized) maps $S : \mathcal{F}^n \to \{0,1\}^*$ and $\mathsf{Rec} : \{0,1\}^* \to \mathcal{F}^n$ such that:

- For all pairs of strings w, w' of distance at most t, we have $\mathsf{Rec}(w', S(w)) = w$ with probability 1.

- For all k-sources W, we have $\bar{\mathbf{H}}_\infty(W \mid S(W)) \geq k'$.

The *entropy loss* of a sketch is defined as $k - k'$. $\diamondsuit$ $\diamondsuit$

Intuitively, a secure sketch allows one to correct errors in W while giving up the smallest amount of entropy about W (which is exactly the entropy loss $k - k'$). Also notice that the most direct way to bound the entropy loss is to make the output length of the sketch as small as possible: indeed, it is easy to see that $k - k' \leq |S(W)|$. Bounding the length of the sketch is also important from the perspective of communication complexity for *information reconciliation*: if Alice wants to transmit her string W to Bob (who knows some noisy version W' of W), sending a shorter sketch will result in a more communication-efficient protocol.

Code-Offset Construction. The following well known construction builds secure sketches for the Hamming space $\mathcal{F}^n$, where $\mathcal{F}$ is a field. Recall that a linear $[n, \ell, d]$-code consists of a ℓ-dimensional subset C of the vector space $\mathcal{F}^n$, with the property that any two distinct vectors $x, y \in C$ have Hamming distance at least d (called the *minimal distance* of C).

A parity-check matrix H for C is any matrix whose rows generate the orthogonal space $C^{\perp}$. Fixing such a matrix, for any $v \in \mathcal{F}^n$ we can define the *syndrome* of v w.r.t. C as $\mathsf{syn}_C(v) \overset{\text{def}}{=} Hv$. I.e., the syndrome of a vector is its projection onto subspace that is orthogonal to the code, and can thus be intuitively viewed as the vector modulo the code. Note that $v \in C \Leftrightarrow \mathsf{syn}_C(v) = 0$. Note also that H is an $(n-\ell) \times n$ matrix, and thus $\mathsf{syn}_C(v)$ is $(n-\ell)$ field-elements long. Also, it is well known that any error vector e of Hamming weight less than $d/2$ is (in principle) uniquely determined from its syndrome $\mathsf{syn}_C(e)$. Moreover, efficiently decodable codes can recover this e in polynomial time from its syndrome.

Given an efficiently decodable $[n, \ell, d]$-code, where $d = 2t + 1$, we now define $S(w) = \mathsf{syn}_C(w)$. As for the recovery procedure, notice that if $\mathsf{dist}(w, w') \le t < d/2$, then $w - w'$ defines a vector e of Hamming weights less than $d/2$. Moreover, $\mathsf{syn}_C(e) = \mathsf{syn}_C(w) - \mathsf{syn}_C(w') = S(w) - \mathsf{syn}_C(w')$ can be recovered from $S(w)$ and w'. By efficient decodability of the code, this means we can recover e, and thus $w = w' + e$. Overall, we obtain a secure sketch for $\mathcal{F}^n$ with entropy loss at most $|S(w)| = (n-k)\log|\mathcal{F}|$. (This loss was shown to be nearly optimal in [5].) For example, in case $|\mathcal{F}| \ge n$, we can use Reed-Solomon codes which have $k = n - d + 1 = n - 2t$, obtaining (optimal) entropy loss $2t \log|\mathcal{F}|$.

Fuzzy Extractors From Secure Sketches. As noticed by [5], secure sketches naturally combine with the leftover hash lemma (more generally, with any strong extractor) to yield nearly optimal fuzzy extractors, whose entropy loss is that of the secure sketch plus $2\log(1/\varepsilon)$.

Lemma 4 (Fuzzy Extractors from Sketches [5]) *Assume* (S, Rec) *is an* (n, k, k', t)*-secure sketch, and let* Ext *be the* (n, k', m, ε)*-strong extractor based on universal hashing (in particular,* $m = k' - 2\log(1/\varepsilon)$*). Then the following* $(\mathsf{Gen}, \mathsf{Rep})$ *is a* $(n, k, m, t, \varepsilon)$*-fuzzy extractor:*

- $\mathsf{Gen}(w; (r, i))$*: set* $P = (S(w; r), i)$ *and* $R = \mathsf{Ext}(w; i)$*.*
- $\mathsf{Rep}(w', (s, i))$*: output* $R = \mathsf{Ext}(\mathsf{Rec}(w', s), i)$*.* $\diamondsuit$

6 Correcting errors without leaking partial information: Entropically Secure Sketches

We now combine the notions of error-correction and entropic security. For a motivation, we saw that secure sketches allow one to correct errors in W without significantly lowering its entropy. They do, however, leak information about W: for example, the syndrome construction revealed the entire syndrome of W. Can we build secure sketches which leak no information about W? Unfortunately, we know that secure sketches must leak "Shannon information" about W [2]; i.e., the entropy of W must drop given $S(W)$. Surprisingly enough, it was shown in [7] that (for the Hamming distance) it *is* nevertheless possible for the secure sketches to hide all functions of W; i.e., to be entropically secure! Put differently, it is possible to *correct errors in W without revealing a-priori information about W*.

Theorem 2 ([7]) (Binary Alphabet) *There exist efficient* (n, k, k', t)*-secure sketches for inputs in* $\{0,1\}^n$ *which are also* (k, ε)*-entropically secure, such that*

1. *the tolerated error t and residual entropy k' are $\Omega(n)$;*

2. *the information leakage ε is exponentially small in n,*

whenever the original min-entropy k is linear in n. That is, whenever $k = \Omega(n)$, we can find entropically secure sketches where t, k' and $\log(1/\varepsilon)$ are $\Omega(n)$.

(Large Alphabet) *If $|\mathcal{F}| = q > n$ and $k > 2t\log(q)$, there exist efficient (n, k, k', t)-entropically secure sketches with leakage ε over $\mathcal{F}^n$ such that $k' = k - 2t\log(q)$ and $\varepsilon = O(2^{-k'/2})$. Both of these parameters are optimal.* $\diamondsuit$

A few comments are in place. First, if an (n, k, k', t)-secure sketch is also (k, ε)-entropically secure, then k' is bounded below by $\log(1/\varepsilon)$ (roughly), since by the definition of entropic security the adversary's probability of predicting the identity function $f(W) = W$ is at most $\varepsilon + 2^{-k} \approx \varepsilon$. Thus, good entropic security automatically gurantees high residual min-entropy k'. Second, by Corollary 1, to demonstrate Theorem 2 it suffices to construct *randomness extractors* which are simultaneously secure sketches! In fact, [7] even constructed a *strong* randomness extractor (whose output included the seed) with this property. Namely, they constructed a strong extractor Ext such that w can be recovered from $\mathsf{Ext}(w; i)$, the seed i and any w' close to w. Unlike the standard rational for extractors, however, the objective of such "secure-sketch extractors" is to *minimize* their output length, since this length corresponds to the length of the secure sketch, which directly bounds the entropy loss of the sketch. In other words, the purpose of this extractor is the recovery of w using the minimal amount of information, and not the randomness extraction (which only serves as a convenient tool to argue privacy).

Finally, it is also instructive to compare such invertible extractors with the invertible extractors studied in Section 2. There we could also recover w from $\mathsf{Ext}(w; i)$ and the seed i, but without the string w' close to w. As a consequence, the output length such extractors had to be at least n. Here, by also giving a string w' close to w, the objective is to push the output length down as much as possible: not only below n, but also significantly below the min-entropy k!

Construction. The secure sketch/strong extractor construction of [7] used a special family $\{C_i\}_i$ of $[n, \ell, d = 2t + 1]$-codes (for "appropriate" ℓ), and set $S(w; i) = (i, \mathsf{syn}_{C_i}(w))$. The challenge was to obtain the largest possible dimension ℓ such that, for a random code C_m and for any k-source W, $(m, \mathsf{syn}_{C_m}(W))$ is close to uniform. We refer to [7] for the details on how to build such codes in order to prove Theorem 2, here only stating (without proof) the actual construction for the large alphabet case. We start from a fixed code C equal to the $[n, n - 2t, 2t + 1]$-Reed-Solomon code. Given an index $i = (a_1, \ldots a_n)$ consisting of *non-zero* elements of $\mathcal{F}$, we define $C_i = \{(a_1 \cdot c_1, \ldots, a_n \cdot c_n) \in \mathcal{F}^n \mid (c_1, \ldots, c_n) \in C\}$. (Restricting a_j's to be non-zero ensures that each C_i still has minimal distance $2t + 1$.) The resulting family is $\{C_{(a_1,\ldots,a_n)} \mid a_1 \neq 0, \ldots, a_n \neq 0\}$. Theorem 2 states that the resulting secure skecth matches the entropy loss of the regular, "entropically insecure" sketch presented in Section 5!

Application: Private Fuzzy Extractors. A $(n, k, m, t, \varepsilon_1)$-fuzzy extractor (see Definition 5) is called (k, ε)-*private*, if its generation procedure $\mathsf{Gen}(W) \to (R, P)$ is (k, ε)-indistinguishable (and, thus, $(k + 2, O(\varepsilon))$-entropically secure). Such extractors imply that no a-priori information about W is leaked both from the extracted randomness R and the public value P. Even stronger, such an extractor is called (k, ε)-*uniform* if $\mathsf{SD}(R, P), (U_{|R|}, U_{|P|}) \leq \varepsilon$. Namely, in the latter case $\mathsf{Gen}(W) \to (R, P)$ by itself could be viewed as a randomness extractor, whose first part of the output R could be the recovered from the second (independent!) part P and any string W' close to the source W.

It is easy to see that applying the construction from Lemma 4 to any (k, ε_2)-indistinguishable sketch S gives an $(k, \varepsilon_1 + \varepsilon_2)$-private fuzzy extractor. And if the sketch by itself is an extractor, we get a $(k, \varepsilon_1 + \varepsilon_2)$-uniform fuzzy extractor! Assuming $k = \Omega(n)$ and applying now the construction from Theorem 2, we get a (k, ε)-uniform fuzzy extractor all of whose parameters are optimal up to a constant factor: $m, t, \log(1/\varepsilon) = \Omega(n)$, and $|P| = O(n)$. Moreover, this fuzzy extractor is "strong" in a sense that P contains (together with other data) all the randomness m used by Gen.

Application: Fuzzy POWHFs. It was also observed in [7] that entropically secure sketches compose well with any ordinary POWHFs (see Definition 4), as long as the residual min-entropy of the secret given the sketch is higher than the min-entropy requirement for the POWHF. As a result, using the same notation as it Definition 4, [7] obtained a family of what we call (k, t, ε)-*fuzzy perfectly one-way* hash functions, satisfying the following three conditions:

- For all keys $\mathsf{key} \in \mathcal{K}_n$, inputs $w, w' \in \{0,1\}^n$ satisfying $\mathsf{dist}(w, w') \leq t$, and strings $i \in \mathcal{R}_n$, we have $\mathsf{Ver}(\mathsf{key}, w', H_k(w; i)) = \textsc{acc}$.

- For any efficient adversary $\mathcal{A}$, the probability over $k \in K_n$ that $\mathcal{A}(\mathsf{key})$ outputs a triple (w, w', y) such that $\mathsf{dist}(w, w') > 2t$ and $\mathsf{Ver}(\mathsf{key}, w, y) = \mathsf{Ver}(\mathsf{key}, w', y) = \textsc{acc}$ is negligible in n.

- For all keys $\mathsf{key} \in \mathcal{K}_n$, the randomized map $W \mapsto H_{\mathsf{key}}(W; \mathcal{I})$ is (k, ε)-entropically secure.

Thus, publishing $H_k(w, i)$ allows anybody to test (without either false positives or false negatives!) whether or not some input w' is *close* to the "correct" secret input w, and yet without leaking any particular function of w. Taking now the specific construction of entropically secure sketches from Theorem 2 and the construction of POWHFs from Section 3, we obtain, for any entropy level $t = \Omega(n)$, a (t, τ, ε)-fuzzy POWHF where both τ and $\log(1/\varepsilon)$ are $\Omega(n)$.

Application: Key Reuse in the Noisy BSM. Perhaps as the most surprising application of entropically secure sketches, [7] showed that they can be used to simultaneously achieve error correction, key reuse and "everlasting security" in the so called bounded storage model (BSM) [10]. This resolved the main open problem of [4]. We refer to [7] for more details and references regarding this application.

References

[1] Noga Alon, Oded Goldreich, Johan Håstad, René Peralta: Simple Constructions of Almost k-Wise Independent Random Variables. FOCS 1990: 544-553

[2] Gilles Brassard, Louis Salvail. Secret-Key Reconciliation by Public Discussion. In *Advances in Cryptology — EUROCRYPT 1993*, p. 410–423.

[3] R. Canetti, D. Micciancio, O. Reingold. Perfectly One-Way Probabilistic Hash Functions. In *Proc. 30th ACM Symp. on Theory of Computing*, 1998, pp. 131–140.

[4] Y.Z. Ding. Error Correction in the Bounded Storage Model. In *Theory of Cryptography Conference 2005*, pp. 578–599.

[5] Y. Dodis, L. Reyzin and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Advances in Cryptology — EUROCRYPT 2004*.

[6] Y. Dodis and A. Smith. Entropic Security and the Encryption of High-Entropy Messages. In *Theory of Cryptography Conference 2005*.

[7] Y. Dodis and A. Smith. Correcting Errors Without Leaking partial Information. *Proc. 37th ACM Symp. on Theory of Computing*, 2005, pp. 654–663.

[8] J. Håstad, R. Impagliazzo, L. Levin, M. Luby. A Pseudorandom generator from any one-way function. In *Proc. 21st ACM Symp. on Theory of Computing*, 1989.

[9] C.-J. Lee, C.-J. Lu, S.-C. Tsai and W.-G. Tzeng. Extracting Randomness from Multiple Independent Sources. In *IEEE Transactions on Information Theory (SCI)*, 51(6):2224–2227, 2005.

[10] U. Maurer. Conditionally-Perfect Secrecy and a Provably Secure Randomized Cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[11] J. Naor, M. Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. In *SIAM J. Comput.* 22(4): 838-856 (1993).

[12] N. Nisan, D. Zuckerman. Randomness is Linear in Space. In *JCSS*, **52**(1), pp. 43–52, 1996.

[13] A. Russell and Wang. How to Fool an Unbounded Adversary with a Short Key. In *Advances in Cryptology — EUROCRYPT 2002*.

[14] Y. Aumann and M.O. Rabin. Information theoretically secure communication in the limited storage model. In *Advances in Cryptology — CRYPTO 1999*.

[15] Salil P. Vadhan. Constructing Locally Computable Extractors and Cryptosystems in the Bounded-Storage Model. *J. Cryptology* 17(1): 43-77 (2004).