On Extractors, Error-Correction and Hiding All Partial Information

(Invited Paper)

Yevgeniy Dodis
Department of Computer Science
New York University
Email: `dodis@cs.nyu.edu`

***Abstract*— Randomness extractors [12] allow one to obtain nearly perfect randomness from highly imperfect sources randomness, which are only known to contain "scattered" entropy. Not surprisingly, such extractors have found numerous applications in many areas of computer science including cryptography. Aside from extracting randomness, a less known usage of extractors comes from the fact that they hide all deterministic functions of their (high-entropy) input [6]: in other words, extractors provide certain level of privacy for the imperfect source that they use. In the latter kind of applications, one typically needs extra properties of extractors, such as invertibility, collision-resistance or error-correction. In this abstract we survey some of such usages of extractors, concentrating on several recent results by the author [5], [6], [7]. The primitives we will survey include several flavors of randomness extractors, entropically secure encryption and perfect one-way hash functions. The main technical tools will include several variants of the leftover hash lemma, error correcting codes, and the connection between randomness extraction and hiding all partial information.**

Due to space constraints, many important references and results are not mentioned here; interested reader can find those in [5], [6], [7].

I. Randomness Extractors and Entropic Security

The main measure of entropy we use is *min-entropy*, which measures the difficulty of guessing a random variable A a-priori: $\mathbf{H}_\infty(A) = -\log(\max_a \Pr[A = a])$ (all logarithms are base 2 by default). A is called a t-source if $\mathbf{H}_\infty(A) \geq t$. The conditional min-entropy of A given B is $\bar{\mathbf{H}}_\infty(A \mid B) \stackrel{\text{def}}{=} -\log(\mathbb{E}_{b\leftarrow B}\left[2^{-\mathbf{H}_\infty(A|B=b)}\right])$. (This definition is not standard but is "right" for cryptographic purposes [5].) We let U_ℓ denote the uniform distribution on $\{0,1\}^\ell$, and define the *statistical difference* between two distributions A and B on the same space as $\mathbf{SD}(A,B) \stackrel{\text{def}}{=} \frac{1}{2}\sum_v \big|\Pr[A = v] - \Pr[B = v]\big|$. If $E(x;\mathcal{I})$ is a probabilistic algorithm taking input x and using randomness $\mathcal{I}$, we will often omit $\mathcal{I}$ and have $E(x)$ denote the random variable $E(x;\mathcal{I})$, where $\mathcal{I}$ is assumed to be sampled uniformly at random from its domain. If X is a random variable, then $E(X)$ also stands for $E(X;\mathcal{I})$.

Definition 1 A polynomial time probabilistic function $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^\ell$ with randomness $\mathcal{I}$ (called the *seed*) is a (n,t,ℓ,ϵ)-*extractor* if for all t-sources W on $\{0,1\}^n$, we have $\mathbf{SD}(\mathsf{Ext}(W;\mathcal{I}), U_\ell) \leq \epsilon$. Ext is a (n,t,ℓ,ϵ)-*strong extractor* if its extracted randomness is statistically independent from the seed $\mathcal{I}$. Namely, if $\mathsf{Ext}'(W;\mathcal{I}) = \mathsf{Ext}(W;\mathcal{I}) \circ \mathcal{I}$ is by itself an extractor: $\mathbf{SD}((\mathsf{Ext}(W;\mathcal{I}),\mathcal{I}),(U_\ell,\mathcal{I})) \leq \epsilon$. $\diamondsuit$

It is well known that one must have $\ell \leq t - 2\log\left(\frac{1}{\epsilon}\right) + O(1)$. On a constructive side, this bound can be achieved, and with very short seeds $\mathcal{I}$ (of length $O(\log n + \log\left(\frac{1}{\epsilon}\right))$). For most of our applications, however, minimizing the seed length will be of secondary importance, and the famous *leftover hash lemma* (LHL), which we describe below, will be more than sufficient.

Definition 2 A family $\mathcal{H} = \left\{h_i : \{0,1\}^n \to \{0,1\}^\ell\right\}_i$ is called a *universal* hash family if for every distinct $x, y \in \{0,1\}^n$ we have $\Pr_{\mathcal{I}}[h_{\mathcal{I}}(x) = h_{\mathcal{I}}(y)] = 2^{-\ell}$. $\diamondsuit$

A simple construction of such a family for general n and ℓ views both the input x and the randomness a (defining the index i) as elements of finite field $GF[2^{\max(n,\ell)}]$, and sets $h_i(x)$ to be the first ℓ bits of the field product $a \cdot x$. The LHL below states that such universal hash functions (strongly) extract all the randomness from any t-source.

Lemma 1 ([8] Leftover Hash Lemma): If $\ell \leq t - 2\log\left(\frac{1}{\epsilon}\right)$, then a universal family $\mathcal{H} = \left\{h_i : \{0,1\}^n \to \{0,1\}^\ell\right\}_i$ is a (n,t,ℓ,ϵ)-strong extractor. $\diamondsuit$

We remark that LHL is pretty robust. For example, its conclusions holds even if the collision probability of universal hash functions is relaxed to roughly $2^{-\ell}(1+\epsilon^2)$, and also even if the index $\mathcal{I}$ choosing the function $h_{\mathcal{I}}$ itself comes from an imperfect source [9]: if $\mathbf{H}_\infty(\mathcal{I}) \geq |\mathcal{I}| - c$ and $\ell \leq t - c - 2\log\left(\frac{1}{\epsilon}\right)$, then $\mathbf{SD}((h_{\mathcal{I}}(W),\mathcal{I}),(U_\ell,\mathcal{I})) \leq \epsilon$. It also generalizes nicely w.r.t. conditioning [5]: if W and S are such that $\bar{\mathbf{H}}_\infty(W|S) \geq t$, then $\mathbf{SD}((h_{\mathcal{I}}(W),S,\mathcal{I}),(U_\ell,S,\mathcal{I})) \leq \epsilon$.

Entropic Security. Entropic security first appeared in specific contexts of entropically secure encryption [13] and perfectly one-way hash functions (POWHF) [3]. Here we define the general concept following [6].

Definition 3 ([6]) The probabilistic map $Y()$ *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 $f : \{0,1\}^* \to \{0,1\}^*$,

$$\big|\Pr[\mathcal{A}(Y(W)) = f(W)] - \Pr[\mathcal{A}_*() = f(W)]\big| \leq \epsilon.$$

The map $Y()$ is called (t,ϵ)-*entropically secure* if $Y()$ hides all functions of W, for all t-sources W. $\diamondsuit$

Intuitively, entropic security of Y states that as long as the random variable W has high-enough min-entropy, it is nearly as hard to predict $f(W)$ given Y as it is without Y, regardless of the adversary's computing power. We remark that it should not be confused with a much stronger notion of Shannon security, where we say that Y and W are (almost) statistically independent. Entropic security only states that Y does not help in predicting $f(W)$, for any function f *specified in advance*. For example, it is entirely possible that after seeing any particular realization y of Y, one might learn many functions f_y of W (although no particular function f is likely to be one of them for an average y). Another unfortunate possibility is that Y_1 and Y_2 could be individually entropically secure, and yet W can be completely recovered from $Y_1(W)$ and $Y_2(W)$ (see [6]). Nonetheless, the information-theoretic guarantee of entropic security is still pretty strong and useful, and we will see examples where it can be applied, despite the fact that Shannon's security is not even achievable!

A seemingly weaker definition is that of (t, ϵ)-indistinguishability [6]: a (probabilistic) map $Y()$ is (t, ϵ)-*indistinguishable* if for all pairs of t-sources W_1, W_2, the distributions $Y(W_1)$ and $Y(W_2)$ are ϵ-close. In particular, all such distributions are ϵ-close to a fixed distribution $\tilde{Y} = Y(U_n)$. Notice that when $\tilde{Y}$ is equal (or at least close) to the uniform distribution, then we retrieve the definition of a randomness extractor! In particular, extractors for t-sources are trivially (t, ϵ)-indistinguishable. The following, perhaps surprising but very useful, result states that entropic security is essentially equivalent to indistinguishability.

Theorem 2 ([6]): If $Y()$ is (t, ϵ)-entropically secure, then it is $(t-1, 4\epsilon)$-indistinguishable. Conversely, if $Y()$ is (t, ϵ)-indistinguishable, then it is $(t+2, 8\epsilon)$-entropically secure. $\diamondsuit$

Corollary 3: Extractors for min-entropy t hide all functions for sources of min-entropy $t+2$. $\diamondsuit$

The above connection states that, to design an entropically secure map satisfying some special functionality, it is sufficient to design a "special purpose" randomness extractor having this functionality. From a different perspective, a randomness extractor with some special property immediately gives an entropically secure map with the same property.

II. Adding Invertibility: Entropically Secure Encryption

As a first special property, let us consider *invertible extractors*: namely, the extractor's 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 ℓ of the extractor must be at least n. Since these $\ell \geq n$ bits are (nearly) uniform, and W has only t bits of entropy, we see that the remaining at least $n-t$ bits of the entropy must come from the seed $\mathcal{I}$. In fact, the lower bound on extractors easily implies that $|\mathcal{I}| \geq n - t + 2\log\left(\frac{1}{\epsilon}\right) - O(1)$. 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 $\mathsf{Inv}(U_\ell, \mathcal{I})$. However, this is impossible, since W could be any distribution of min-entropy t (and we assume $t < n$). However, some part $\mathcal{I}_p$ of the index $\mathcal{I} = (\mathcal{I}_s, \mathcal{I}_p)$ could indeed independent from the output. (We call such extractors *"semi-strong"*.) Unfortunately, an extension of the previous argument still implies that the "secret part" $\mathcal{I}_s$ of the seed must be long: $|\mathcal{I}_s| \geq n - t + 2\log\left(\frac{1}{\epsilon}\right) - O(1)$. However, having the public part $\mathcal{I}_p$ we enable us to achieve simpler constructions.

Application: Entropically Secure Encryption. Assume now that the source W is the message, the seed $\mathcal{I}$ is the shared secret key, and the extracted randomness $C = \mathsf{Ext}(W; \mathcal{I})$ is the ciphertext. By invertibility of our extractor, the recipient can indeed recover the message W from the ciphertext C and the secret key $\mathcal{I}$. What about security? Corollary 3 immediately implies that if Ext is a $(t-2)$-source extractor, then the ciphertext C hides all functions about the message W, as long as W has min-entropy t. This is exactly the notion of (t, ϵ)-entropically secure encryption [13], [6]! Once again, this notion is weaker than that of Shannon's security, but it also allows us to have shorter secret keys than what is possible in the latter setting. While Shannon's famous impossibility results requires a secret key of length at least n, here we will achieve keys of length slightly more than $(n-t)$.

In fact, there is a nice connection between Shannon's security and entropic security: by considering a distribution W_m whose first $(n-t)$ bits are fixed to some message m and the remaining t bit are chosen uniformly at random, it is easy to see that entropic security of n-bit t-source W_m implies Shannon's security of $(n-t)$-bit message m (which is indeed recoverable from any possible value in the support of W_m). This connection also implies that entropically-secure schemes must have secret keys of length at least $(n-t)$.

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. The idea of [6] is to construct invertible extractors from good expander graphs, which in fact mix in one step! They call such graphs *"extractor graphs"*. A bit more formally, assume we have a d-regular expander graph G on 2^n vertices V with the property that for any subset T of 2^t 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 t-source W is known to be a convex combination of uniform distributions on some subsets T of size 2^t, it is obvious that such extractor graphs immediately yeild a $(n, t, \ell = n, \epsilon)$-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 uder this labeling, we can recover w back. We call such natural labelings *invertible*. Luckily, Hall's marriage theorem implies that every 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 d, since $\log d$ will translate to the length of the seed (i.e., the secret key). [6] give three such contructions, whose properties are stated below and then explained in more detail.

Theorem 4: There exists (three different) invertible (n,t,n,ϵ)-extractors with the following properties:

1) Optimal: The seed length $|\mathcal{I}| = n-t+2\log\left(\frac{1}{\epsilon}\right)+O(1)$.
2) "Sparse One-Time Pad": The seed length $|\mathcal{I}| = n-t+2\log\left(\frac{1}{\epsilon}\right)+2\log n+O(1)$, where the seed $\mathcal{I}$ consists of a random point s in some "special" set $S\subset\{0,1\}^n$, and $\mathsf{Ext}(W;s)=s\oplus W$.
3) "LHL-based semi-strong extractor": the seed $\mathcal{I}$ consists of a secret part $\mathcal{I}_s$ of length $k=n-t+2\log\left(\frac{1}{\epsilon}\right)+O(1)$, which is just a random point $x\in\{0,1\}^k$, and a public part $\mathcal{I}_p$, which samples a random hash function $h_{\mathcal{J}}$ from any family $\mathcal{H}=\left\{h_j:\{0,1\}^k\to\{0,1\}^n\right\}_j$ of XOR-universal hash functions. The extractor is $\mathsf{Ext}(W;(X,\mathcal{J}))=h_{\mathcal{J}}(X)\oplus W$. $\diamondsuit$

The first of these extractors is obtained by using (optimal) Ramanujan expander graphs, which indeed have $\log d = n-t+2\log\left(\frac{1}{\epsilon}\right)+O(1)$. However, the constructions of such optimal graphs is relatively complex. A simpler and much more intutive second construction (with only marginally worse parameters) results if we consider graphs induced by the so called δ-biased spaces [11]. A set S in $\{0,1\}^n$ is *δ-biased* if for all nonzero $\alpha\in\{0,1\}^n$, the binary inner product $\alpha\odot s$ is nearly balanced for s drawn uniformly in S: namely, $\Pr_{s\leftarrow S}[\alpha\odot s=0]\in\left[\frac{1-\delta}{2},\frac{1+\delta}{2}\right]$. Alon et al. [1] gave explicit constructions of δ-biased sets in $\{0,1\}^n$ with size $O(n^2/\delta^2)$. Given such a set S, we can construct a graphs G_S where two nodes $x,y\in\{0,1\}^n$ are connected if and only if $x\oplus y\in S$. It is well known (see [6]) that such graphs will be (t,ϵ)-expander graphs provided $\delta\le\epsilon 2^{(n-t-2)/2}$. Coupled with optimal constructions of δ-biased spaces, we get graphs with $\log d = n-t+2\log\left(\frac{1}{\epsilon}\right)+2\log n+O(1)$. Also notice that the encryption/decryption procedure here is indeed a "very sparse" one-time pad: a random neigbor of W is simply $W\oplus s$.

Still, δ-biased sets are relatively non-trivial to construct. To get a very simple construction, the last construction builds much simpler "semi-strong" extractors with optimally short secret part $\mathcal{I}_s$ and relatively short public part. In the graph terminology, we construct a family of d-regular graphs $\{G_j\}$ (for optimally small d) which are good "average-case" expanders: for any set T of size 2^t, a random graph $G_{\mathcal{J}}$ will be a good extractors graph for T, as discussed above. However, it is much easier to understand this last construction directly, since it is very related to the LHL. First, as a slightly generalization of universal hash functions from Definition 2, we say that a family $\mathcal{H}=\left\{h_j:\{0,1\}^k\to\{0,1\}^n\right\}_j$ is called a *XOR-universal* if for every distinct $x,y\in\{0,1\}^n$ and every $\Delta\in\{0,1\}^n$ we have $\Pr_{\mathcal{J}}[h_{\mathcal{J}}(x)\oplus h_{\mathcal{J}}(y)=\Delta]=2^{-n}$. In particular, restricting Δ to 0^n gives the usual definition of universal hash functions, but the $a\cdot x$ construction of universal hash functions described earlier is in fact XOR-universal already. The following variant of the LHL was proven in [6].

Lemma 5 ([6] LHL′: One-Time Pad Extractor): If $\mathcal{H}=\left\{h_j:\{0,1\}^k\to\{0,1\}^n\right\}_j$ is XOR-universal and X and W are two independent distributions on $\{0,1\}^k$ and $\{0,1\}^n$, respectively, satisfying $\mathbf{H}_\infty(X)+\mathbf{H}_\infty(W)\ge n+2\log\left(\frac{1}{\epsilon}\right)+1$, then $\mathbf{SD}\left((\mathcal{J},h_{\mathcal{J}}(X)\oplus W),(\mathcal{J},U_n)\right)\le\epsilon$. $\diamondsuit$

The last construction of Theorem 4 follows from this result by taking $k=n-t+2\log\left(\frac{1}{\epsilon}\right)+1$ and having X to be uniform on $\{0,1\}^k$. Applied to the $a\cdot x$ XOR-universal family, we get the following very simple entropically-secure encryption scheme: view $GF[2^k]$ 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 $x\in GF[2^k]$ is the secret key.

III. 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 k 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 k, 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 $z=\mathsf{Ext}_k(w;i)$); and (b) the definition of collision resistance states that, for a random key k, it is hard to find $w\neq w'$ and a seed i such that $\mathsf{Ext}_k(w;i)=\mathsf{Ext}_k(w';i)$. 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 k, but for 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 extractor's output is close to the uniform as opposed to some other *fixed* distribution). We call such (t,ϵ)-indistinguishable (but not necessarily extractor) collision-resistant maps *(t,ϵ)-privately binding*.

Application: Perfectly One-Way Hash Functions. (t,ϵ)-privately binding maps immediately give a construction of *(t,ϵ)-perfectly one-way hash functions* (POWHFs) [3], [6] (here we give a slightly stronger than usual definition, since it is simpler to state and we will be able to achieve it):

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

- For all keys $k \in \mathcal{K}_n$, inputs $w \in \{0,1\}^n$, and strings $i \in \mathcal{R}_n$, $\mathsf{Ver}(k, w, H_k(w;i)) = \text{ACC}$.
- For any efficient adversary $\mathcal{A}$, the probability over $k \in K_n$ that $\mathcal{A}(k)$ outputs (w, w', y) satisfying $w \neq w'$ and $\mathsf{Ver}(k,w,y) = \mathsf{Ver}(k,w',y) = \text{ACC}$ is negligible in n.
- For all keys $k \in \mathcal{K}_n$, the randomized map $W \mapsto H_k(W;\mathcal{I})$ is (t,ϵ)-entropically secure: for all t-sources W, the value $H_k(W;\mathcal{I})$ 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 2, a $(t+2,\epsilon)$-privately binding map E is (t,ϵ)-entropically secure, and thus immediately gives a (t,ϵ)-POWHF H of the following form: $H_k(w;i)$ outputs $(E_k(w;i), i)$, and $\mathsf{Ver}(k,w,(z,i))$ accepts if and only if $E_k(w;i) = z$.

CONSTRUCTION. Here we present a simple construction from [6], which slightly improves and simplifies the analysis of the original construction from [3]. 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 look random, but it will look the same for all input distributions of sufficiently high entropy.

Lemma 6 ([7] LHL$''$: Composing with arbitrary function): Let $f : \{0,1\}^N \to \{0,1\}^\ell$ be an arbitrary function. If $\mathcal{H} = \left\{h_i : \{0,1\}^n \to \{0,1\}^N\right\}_i$ is a family of pairwise independent hash functions and W is a t-source over $\{0,1\}^n$ with $t \geq \ell + 2\log\left(\frac{1}{\epsilon}\right) + 1$, then $\mathbf{SD}\left((\mathcal{I}, f(h_\mathcal{I}(W))), (\mathcal{I}, f(U_N))\right) \leq \epsilon$. $\diamondsuit$

Contrary to intuition, this statement does *not* follow directly from the usual LHL (Lemma 1), 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_k whose output length $\ell \leq t - 2\log\left(\frac{1}{\epsilon}\right) - 1$ (and whose choice will fix the collision-resistant key k). 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 par of random *distinct elements*. Although they are technically not pairwise independent *functions*, they are 2^{-n}-close to them, which will not affect Lemma 6. 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 (t,ϵ)-privately binding map: $E_k(w,i) = C_k(h_i(w))$. Its (t,ϵ)-indistinguishability follows directly from Lemma 6, while its collision-resistance follows from that of C_k and the fact that h_i is a *permutation*, for any i. Also notice that if the collision-resistant function C_k is regular, i.e. $C_k(U_N) \equiv U_\ell$, then we indeed get a collision-resistant randomness extractor. See [3] for a construction of such regular collision-resistant functions.

IV. ADDING ERROR-CORRECTION: 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 *$(\mathcal{M}, t, \ell, \tau, \epsilon)$-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 $R \in \{0,1\}^\ell$ and a public string P, such that for any t-source W, if $(R,P) \leftarrow \mathsf{Gen}(W)$, then $\mathbf{SD}\left((R,P),(U_\ell,P)\right) \leq \epsilon$.
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 $\mathsf{dist}(w,w') \leq \tau$, 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 $t - \ell$. $\diamondsuit$

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,t,ℓ,τ,ϵ)-fuzzy. The binary case $\mathcal{F} = \{0,1\}$ will be of special importance.

SECURE SKETCH. Notice that in the "error-free" case ($\tau = 0$) 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, t, t', τ)-*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 τ, we have $\mathsf{Rec}(w', S(w)) = w$ with probability 1.
- For all t-sources W, we have $\tilde{\mathbf{H}}_\infty(W \mid S(W)) \geq t'$.

The *entropy loss* of a sketch is defined as $t - t'$. ♢

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 $t - t'$). 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 $t - t' \leq |SS(W)|$. 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 $[n, k, d]$-code consists of a k-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 $\mathsf{syn}_C(v) \stackrel{\text{def}}{=} Hv$. 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 $(n-k) \times n$ matrix, and thus $\mathsf{syn}_C(v)$ is $(n-k)$ 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 $[n, k, d]$-code, where $d = 2\tau + 1$, we now define $S(w) = \mathsf{syn}_C(w)$. As for the recovery procedure, notice that if $\mathsf{dist}(w, w') \leq \tau < d/2$, 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 $|\mathcal{F}| \geq n$, we can use Reed-Solomon codes which have $k = n - d + 1 = n - 2\tau$, obtaining (optimal) entropy loss $2\tau \log|\mathcal{F}|$.

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 $2\log\left(\frac{1}{\epsilon}\right)$.

Lemma 7 (Fuzzy Extractors from Sketches [5]): Assume (S, Rec) is an (n, t, t', τ)-secure sketch, and let Ext be the (n, t', ℓ, ϵ)-strong extractor based on universal hashing (in particular, $\ell = t' - 2\log\left(\frac{1}{\epsilon}\right)$). Then the following $(\mathsf{Gen}, \mathsf{Rep})$ is a $(n, t, \ell, \tau, \epsilon)$-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 $R = \mathsf{Ext}(\mathsf{Rec}(w', s), i)$. ♢

V. CORRECTING ERRORS WITHOUT LEAKING PARTIAL INFORMATION

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 8 ([7]): **(Binary Alphabet)** There exist efficient (n, t, t', τ)-secure sketches for inputs in $\{0,1\}^n$ which are also (t, ϵ)-entropically secure, such that

1) the tolerated error τ and residual entropy t' are $\Omega(n)$;
2) the information leakage ϵ is exponentially small in n,

whenever the original min-entropy t is linear in n. That is, whenever $t = \Omega(n)$, we can find entropically secure sketches where τ, t' and $\log\left(\frac{1}{\epsilon}\right)$ are $\Omega(n)$.

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

A few comments are in place. First, if an (n, t, t', τ)-secure sketch is also (t, ϵ)-entropically secure, then t' is bounded below by $\log\left(\frac{1}{\epsilon}\right)$ (roughly), since by the definition of entropic security the adversary's probability of predicting the identity function $f(W) = W$ is at most $\epsilon + 2^{-t} \approx \epsilon$. Thus, good entropic security automatically gurantees high residual min-entropy t'. Second, by Corollary 3, to demonstrate Theorem 8 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 II. 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 t!

CONSTRUCTION. The secure sketch/strong extractor construction of [7] used a special family $\{C_i\}_i$ of $[n, k, d = 2\tau + 1]$-codes (for "appropriate" k), and set $S(w; i) = (i, \mathsf{syn}_{C_i}(w))$. The challenge was to obtain the largest possible dimension k suth that, for a random code $C_{\mathcal{I}}$ and for any t-source W, $(\mathcal{I}, \mathsf{syn}_{C_{\mathcal{I}}}(W))$ is close to uniform. We refer to [7] for the details on how to build such codes in order to prove Theorem 8, here only stating (without proof) the actual construction for the large alphabet case. We start from a fixed code C equal to the $[n, n-2\tau, 2\tau+1]$-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 $2\tau + 1$.) The resulting family is $\left\{C_{(a_1,\ldots,a_n)} \mid a_1 \neq 0, \ldots, a_n \neq 0\right\}$. Theorem 8 states that the resulting secure skecth matches the entropy loss of the regular, "entropically insecure" sketch presented in Section IV!

APPLICATION: PRIVATE FUZZY EXTRACTORS. A $(n, t, \ell, \tau, \epsilon_1)$-fuzzy extractor (see Definition 5) is called (t, ϵ)-*private*, if its generation procedure $\mathsf{Gen}(W) \to (R, P)$ is (t, ϵ)-indistinguishable (and, thus, $(t+2, O(\epsilon))$-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 (t, ϵ)-*uniform* if $\mathbf{SD}\left((R, P), (U_{|R|}, U_{|P|})\right) \leq \epsilon$. 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 7 to any (t, ϵ_2)-indistinguishable sketch S gives an $(t, \epsilon_1 + \epsilon_2)$-private fuzzy extractor. And if the sketch by itself is an extractor, we get a $(t, \epsilon_1 + \epsilon_2)$-uniform fuzzy extractor! Assuming $t = \Omega(n)$ and applying now the construction from Theorem 8, we get a (t, ϵ)-uniform fuzzy extractor all of whose parameters are optimal up to a constant factor: $\ell, \tau, \log\left(\frac{1}{\epsilon}\right) = \Omega(n)$, and $|P| = O(n)$. Moreover, this fuzzy extractor is "strong" in a sense that P contains (together with other data) all the randomness $\mathcal{I}$ 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 (t, τ, ϵ)-*fuzzy perfectly one-way* hash functions, satisfying the following three conditions:

- For all keys $k \in \mathcal{K}_n$, inputs $w, w' \in \{0,1\}^n$ satisfying $\mathsf{dist}(w, w') \leq \tau$, and strings $i \in \mathcal{R}_n$, we have $\mathsf{Ver}(k, w', H_k(w; i)) = \text{ACC}$.
- For any efficient adversary $\mathcal{A}$, the probability over $k \in K_n$ that $\mathcal{A}(k)$ outputs a triple (w, w', y) such that $\mathsf{dist}(w, w') > 2\tau$ and $\mathsf{Ver}(k, w, y) = \mathsf{Ver}(k, w', y) = \text{ACC}$ is negligible in n.
- For all keys $k \in \mathcal{K}_n$, the randomized map $W \mapsto H_k(W; \mathcal{I})$ is (t, 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 8 and the construction of POWHFs from Section III, we obtain, for any entropy level $t = \Omega(n)$, a (t, τ, ϵ)-fuzzy POWHF where both τ and $\log\left(\frac{1}{\epsilon}\right)$ 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.

Acknowledgments: The author is grateful to Leonid Reyzin and Adam Smith for collaborating on the research presented in this abstract.

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*.