Some Notes on Randomness Extraction

Christopher Smith

Last Updated: October 19, 2025

1 Defining Extractors

Informally, a (seeded) randomness extractor is an algorithm that takes as input a truly random seed and some imperfect source of randomness, and outputs bits that are statistically close to uniform. Before formally presenting randomness extractors in Definition 3, we present the prerequisite notions of statistical distance and min-entropy.

Definition 1 (Statistical Distance)**.** *Let X, Y be two random variables with common support U. Then the statistical distance between X and Y is*

$$\mathbf{SD}(X, Y) = \frac{1}{2} \sum_{u \in U} |\Pr[X = u] - \Pr[Y = u]| \tag{1}$$

Definition 2 (Min-Entropy)**.** *The min-entropy of a random variable W is defined as*

$$\mathbf{H}_\infty(W) = -\log \left(\max_w \Pr[W = w] \right) \tag{2}$$

We may now formally define strong randomness extractors. Strong extractors are typically desirable over weak extractors because as long as an initial seed is chosen uniformly at random, it can be publicly exposed and used for all future invocations of the extractor.

Definition 3 (Strong Extractor)**.** *Let $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ be a polynomial time computable function. We say that Ext is an efficient (n, m, ℓ, ϵ)-strong extractor if, for any random variable W on $\{0,1\}^n$ satisfying $\mathbf{H}_\infty(W) \geq m$, we have*

$$\mathbf{SD}((\mathsf{Ext}(W; U_d), U_d) \ , \ (U_l, U_d)) \leq \epsilon \tag{3}$$

where U_d is uniform on $\{0,1\}^d$.

Strong extractors as given in Definition 3 are also sometimes referred to as worst-case strong extractors to distinguish them from the average-case strong extractors of Definition 5. The motivation for average-case extractors (as given in [DORS08]) is that an adversary hoping to learn information about a random variable W may obtain some side information z (sampled from a random variable Z) such that $\mathbf{H}_\infty(W|Z = z)$ is unacceptably low. If we assume, however, that the adversary has no control over its side-information Z—as is often the case—then we can consider extractors based on a weaker notion of min-entropy that averages over all possible $z \leftarrow Z$.

Definition 4 (Average Conditional Min-Entropy [DORS08])**.** *Let (W, Z) be a pair of random variables. The average conditional min-entropy of W given Z is:*

$$\widetilde{\mathbf{H}}_\infty(W|Z) = -\log\left(\mathop{\mathbb{E}}_{z\leftarrow Z}\left[\max_w \Pr[W = w \;\middle|\; Z = z]\right]\right) = -\log\left(\mathop{\mathbb{E}}_{z\leftarrow Z}\left[2^{-\mathbf{H}_\infty(W|Z=z)}\right]\right) \tag{4}$$

Definition 5 (Average-Case Strong Extractor [DORS08])**.** *Let* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ *be a polynomial time computable function. We say that* Ext *is an efficient average-case (n, m, ℓ, ϵ)-strong extractor if, for any pair of random variables (W, Z) such that W is a random variable over $\{0,1\}^n$ satisfying $\widetilde{\mathbf{H}}_\infty(W|Z) \geq m$, we have*

$$\mathbf{SD}((\mathsf{Ext}(W; U_d), U_d, Z)\ ,\ (U_l, U_d, Z)) \leq \epsilon \tag{5}$$

2 Basic Results

Perhaps the most important—or at least the most well-known—result about extractors is that they can be constructed from universal hash functions. This famous result is known as the Leftover Hash Lemma. The term was coined by [IZ89], though the lemma originally appeared in [ILL89]. The version provided here is taken from [DORS08].

Definition 6 (Universal Hash Function [CW77])**.** *A keyed hash function, or, equivalently, a family of hash functions $H = \{H_s : A \to B \mid s \in S\}$ is called universal if, for every $x \neq y \in A$,*

$$|\{s \in S \mid H_s(x) = H_s(y)\}| \leq \frac{|H|}{|B|} = \frac{|S|}{|B|} \tag{6}$$

Or in other words,

$$\Pr_{s\leftarrow S}[H_s(x) = H_s(y)] \leq \frac{1}{|B|} \tag{7}$$

Lemma 7 (Leftover Hash Lemma [DORS08])**.** *Assume a family of functions $\left\{H_s : \{0,1\}^n \to \{0,1\}^\ell\right\}$ is universal. Then, for any random variable W,*

$$\mathbf{SD}((H_S(W), S)\ ,\ (U_\ell, S)) \leq \frac{1}{2}\sqrt{2^{-\mathbf{H}_\infty(W)}2^\ell} \tag{8}$$

In particular, universal hash functions are (n, m, ℓ, ϵ)-strong extractors whenever $\ell \leq m - 2\log(1/\epsilon) + 2$.

Lemma 8 (Relationship Between Worst-Case and Average-Case Strong Extractors [DORS08])**.** *For any $\delta > 0$, if* Ext *is a (worst-case) $(n, m - \log(1/\delta), \ell, \epsilon)$-strong extractor, then* Ext *is also an average-case $(n, m, \ell, \epsilon + \delta)$-strong extractor.*

The upshot of Lemma 8 is that we can instantiate any average-case (n, m, ℓ, ϵ)-strong extractor with a $(n, m - \log(1/\delta), \ell, \epsilon - \delta)$-strong extractor, for any $0 < \delta < \epsilon$. This is useful because we have many constructions of strong extractors. In particular, the Leftover Hash Lemma (Lemma 7) states that universal hash functions are strong extractors. As it turns out, however, we need not appeal to Lemma 8 if we want average-case extractors from universal hash functions, since the following generalized leftover hash lemma gives us this directly.

Lemma 9 (Generalized Leftover Hash Lemma [DORS08])**.** *Assume a family of functions* $\left\{H_s : \{0,1\}^n \to \{0,1\}^\ell\right\}$ *is universal. Then for any pair of random variables* W, Z*,*

$$\mathbf{SD}((H_S(W), S, Z)\ ,\ (U_\ell, S, Z)) \leq \frac{1}{2}\sqrt{2^{-\widetilde{\mathbf{H}}_\infty(W|Z)}2^\ell} \tag{9}$$

In particular, universal hash functions are average-case (n, m, ℓ, ϵ)*-strong extractors whenever* $\ell \leq m - 2\log(1/\epsilon) + 2$*.*

Potentially the simplest and most widely known example of a randomness extractor is the "inner product extractor", which is obtained via application of the leftover hash lemma to a hash family that hashes its input by taking the dot product of the input and the key. The following claim shows how the inner product hash family is universal.

Claim 10 (Inner Product Hashing is Universal)**.** *Let* $\mathbb{F}$ *be a finite field, and* $n \in \mathbb{N}$*. The "inner product" hash family* $H = \{H_s : \mathbb{F}^n \to \mathbb{F} \mid s \in \mathbb{F}^n\}$ *given by* $H_s(x) = \sum_i s_i x_i$ *is universal.*

Proof. Let $x \neq y \in \mathbb{F}^n$, so $\exists i^*$ s.t. $x_{i^*} \neq y_{i^*}$ We have:

$$\Pr_{s \xleftarrow{\$} \mathbb{F}^n}[H_s(x) = H_s(y)] = \Pr\left[s_{i^*} = -\sum_{i \neq i^*} s_i(x_i - y_i)/(x_{i^*} - y_{i^*})\right] = \frac{1}{|\mathbb{F}|}$$

□

3 Linear Extractors are Invertible

It is sometimes desirable in certain applications—such as wiretap protocols [CDS11] and leakage-resilient secret sharing [CKOS22]—to have extractors that are invertible. The work of [CKOS22] tells us that any linear extractor is invertible. An extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ is linear if for any seed $s \in \{0,1\}^d$, $\mathsf{Ext}(\cdot, s)$ is a linear function. That is, $\mathsf{Ext}(\cdot, s)$ is a linear map between $\{0,1\}^n$ and $\{0,1\}^\ell$ viewed as vector spaces. The following lemma due to [CKOS22] gives us a generic polynomial time procedure for finding preimages of any linear extractor. The lemma additionally requires that the linear extractor be "efficient", but we have already assumed this as Definitions 3 and 5 require extractors to be computable in polynomial time.

Lemma 11 (Linear Extractors are Invertible (Lemma 2, [CKOS22]))**.** *For every efficient linear extractor* Ext*, there exists an efficient randomized function* $\mathsf{InvExt} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^n \cup \{\perp\}$ *such that*

1. $U_n, U_d, \mathsf{Ext}(U_n; U_d) = \mathsf{InvExt}(\mathsf{Ext}(U_n; U_d), U_d), U_d, \mathsf{Ext}(U_n; U_d)$
2. *For each* $(y, s) \in \{0,1\}^\ell \times \{0,1\}^d$*,*
   (a) $\Pr[\mathsf{InvExt}(y, s) = \perp] = 1$ *iff* $\nexists w \in \{0,1\}^n$ *such that* $\mathsf{Ext}(w; s) = y$*.*
   (b) $\Pr[\mathsf{Ext}(\mathsf{InvExt}(y, s); s) = y] = 1$ *iff* $\exists w \in \{0,1\}^n$ *such that* $\mathsf{Ext}(w; s) = y$*.*

Proof. We only restate the construction of InvExt. For the rest of the proof, consult Lemma 2 of [CKOS22]. Recall that $\mathsf{Ext}(\cdot, s)$ is a linear map between vector spaces $\{0,1\}^n$ and $\{0,1\}^\ell$. Let $\mathcal{I}_s$ and $\mathcal{K}_s$ be the image and kernel of $\mathsf{Ext}(\cdot, s)$. We now define InvExt as follows:
$\underline{\mathsf{InvExt}(y \in \{0,1\}^\ell, s \in \{0,1\}^d) \to \{0,1\}^n \cup \{\perp\}:}$

- If $y \in \mathcal{I}_s$:
  - Let w be such that $\mathsf{Ext}(w; s) = y$
  - Sample z uniformly from $\mathcal{K}_s$
  - Output $w + z$
- Else output $\perp$.

InvExt is efficient because the bases for the linear subspaces $\mathcal{K}_s, \mathcal{I}_s$, and the preimage space on y can all be determined in polynomial time (*e.g.*, via Gaussian elimination). To be slightly more concrete, we could represent $\mathsf{Ext}(\cdot, s)$ as a matrix M, compute the LUP decomposition of M, use this decomposition to solve for an arbitrary w s.t. $Mw = y$ (or output $\perp$ if no solution exists) and a random z s.t. $Mz = 0$, then output $w + z$. □

4 Towards Optimal Extraction

In Sec. 2 we saw the cute example of the inner product extractor. Assuming $\mathbb{F} = \{0, 1\}$, observe this extractor has a seed length of $d = n$ bits but an output length of only $\ell = 1$ bit. Informally, short seed length is a desirable property because it represents a "thriftier" extractor that, all else fixed, extracts the same amount of randomness for the price of fewer truly random bits. Thus, given an (n, m, ℓ, ϵ)-strong extractor, it is natural to ask for lower bounds on the seed length d in terms of n, m, ℓ, ϵ. Of course, we also want to make sure our extractors still extract as much min-entropy from the source as possible. So, in addition to short seeds, we are interested in extractors where ℓ is close to m [1]. In short, to prove a "lower bound" on the "quality" of any extractor, one can lower bound d and upper bound ℓ.

Chris: TODO: lower bound on d and upper bound on ℓ from NZ93 randomness linear in space.

Chris: TODO: nonconstructive existence result Trevisan pseudorandomness chapter 6 extractors. These aren't efficient are they? Chris: rrv99 cites Tight bounds for depth-two superconcentrators for... both TODOs?

5 The Trevisan Extractor

The Trevisan extractor is an efficient, near-optimal construction in terms of its seed length and entropy loss. The original construction was given by Luca Trevisan [Tre99], and the extractor quality was subsequently improved to near-optimal factors by Raz, Reingold, and Vadhan [RRV99]. We focus on the improved version [RRV99] given by the following result, and adapt notation to match our extractor notation.

Theorem 12 (Theorem 4 [RRV99])**.** *For every $n, m \in \mathbb{N}$ and $\epsilon > 0$ such that $m \leq n$, there exist (efficient) $(n, m, m - \Delta, \epsilon)$-strong extractors, where the entropy loss $\Delta = 2\log(1/\epsilon) + O(1)$ is optimal up to additive constants, and*

1. *$d = O(\log^2(n/\epsilon) \cdot \log m)$, or*
2. *$d = O(\log^2(n) \cdot \log(1/\epsilon) \cdot \log m)$*

[1] Note $m - \ell > 0$ since we consider strong extractors where the seed can be public. The quantity $\Delta := m - \ell$ is called the *entropy loss*.

We are interested in the second extractor of Theorem 12, because the seed length does not have a quadratic dependence on $\log(1/\epsilon)$. Towards the goal of obtaining a concrete description of this extractor amenable to implementation, we unpack the relevant portion of the proof of Theorem 12, which says that this second extractor is obtained by applying Lemma 13 to the second (strong) extractor of Theorem 14.

Lemma 13 (Lemma 28 [RRV99]). *Let* $\mathsf{Ext}_1 : \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^{m-\Delta_1}$ *be any* $(n, m, m - \Delta_1, \epsilon/4)$*-strong extractor with entropy loss* Δ_1*. Then there exists a* $(n, m, m-\Delta, \epsilon)$*-strong extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{d_1+d_2} \to \{0,1\}^{m-\Delta}$ *such that*

1. Ext *has entropy loss* $\Delta = 2\lceil\log(1/\epsilon)\rceil + 5$ Chris: notice this is optimal up to additive constants
2. $d_2 = O(\Delta_1 + \log n)$
3. Ext *is poly-time computable with one oracle query to* Ext_1*.*

Theorem 14 (Theorem 2 [RRV99]). *For every* $n, m, \ell \in \mathbb{N}$ *and* $\epsilon > 0$ *such that* $\ell \leq m \leq n$*, there exist (efficient)* (n, m, ℓ, ϵ)*-extractors with*

1. $d = O(\frac{\log^2 n \cdot \log(1/\epsilon)}{\log(m/\ell)})$*, or*
2. $d = O(\log^2 n \cdot \log(1/\gamma) \cdot \log(1/\epsilon))$*, where* $1 + \gamma = m/(\ell - 1)$*, and* $\gamma < 1/2$*.*

Our task is now split in two parts: (a) understand the transformation of Lemma 13, and (b) understand the second extractor of Theorem 14.

5.1 Transformation of Lemma 13

Chris: $\mathsf{Ext}(x, (y_1, y_2)) = \mathsf{Ext}_1(x, y_1) \circ \mathsf{Ext}_2(x, y_2)$

5.2 Second Extractor of Theorem 14

Before presenting the construction of the extractor, we introduce the following prerequisite notions: weak designs, multilinear error-correcting codes, the Nisan-Wigderson PRG, and pairwise-independent hash functions.

5.2.1 Weak Designs

In combinatorics, a *design* is a collection of sets with small pairwise intersections. We focus specifically on so-called weak designs, adapting notation to clarify the relationship between design parameters and extractor parameters.

Definition 15 (Weak (ℓ, t, p, d)-Design [RRV99,HR03]). *A family of sets* $S_1, \ldots, S_\ell \subseteq [d]$ *is a weak* (ℓ, t, p, d)*-design if*

1. *For all* i*,* $|S_i| = t$
2. *For all* i*,* $\sum_{j<i} 2^{S_i \cap S_j} \leq p \cdot (t - 1)$

Constructing explicit weak designs is a non-trivial task. Indeed, the original weak designs used by Raz *et al.* [RRV99] were obtained by derandomizing a nonconstructive existence proof via Method of Conditional Expectations [Wik24]. While this construction is efficient in the sense that it runs in polynomial time and space, we will focus on the improved weak design constructions given by

the subsequent work of Hartman and Raz [HR03]. These constructions satisfy the following notion of "explicitness".

Definition 16 (Explicit Weak Design [HR03])**.** *A weak (ℓ, t, p, d)-design $S = S_1, \ldots, S_\ell \subset [d]$ is explicit if there exists an algorithm that, given an index $1 \leq i \leq \ell$, outputs the subset S_i in time $T = \mathsf{poly}(\log \ell, t)$ and space $S = O(\log \ell + \log t)$.*

Theorem 17 Chris: TODO: think we need theorem 4 as well guarantees explicit weak designs for general choices of parameters. The proof of this theorem relies on application of a certain transformation to the explicit weak designs guaranteed by Theorem 18, which fixes parameters $d = t^2$ and $p = e^2$. The transformation can be found in Lemma 5.1 of Hartman and Raz [HR03]. We focus instead on the construction underlying Theorem 18.

Theorem 17 (Theorem 3 [HR03])**.**

1. *For every $\ell, t \in \mathbb{N}$, such that $\ell = \Omega(t^{\log t})$, and constant $p > 1$ there exists an explicit weak (ℓ, t, p, d)-design, where $d = O(t^2)$.*
2. *For every $\ell, t \in \mathbb{N}$ and $p > 1$, such that for some constant $\alpha \geq 1, 2^t < \alpha\ell$, there exists an explicit weak (ℓ, t, p, d)-design, where $d = O(t^2/\ln p)$.*

Theorem 18 (Theorem 1 [HR03])**.** *Let $\ell, t \in \mathbb{N}$ and assume for simplicity that t is prime (if t is not prime, pick the smallest prime greater than t), and ℓ is a power of t. Then there exists an explicit weak (ℓ, t, p, d)-design, with $d = t^2$ and $p = e^2$ (where e denotes the base of the natural logarithm).*

Construction. Let $\mathbb{F}_t$ be the finite field of size t. Since $d = t^2$, we can identify each number in $[d]$ with an ordered pair of elements in $\mathbb{F}_t$ (e.g, $\mathbb{F}_t \times \mathbb{F}_t \to [d]$ by $(a, b) \mapsto ta + b$, interpreting a and b as integers). Thus, constructing the design is equivalent to constructing subsets of $\mathbb{F}_t^2$. The weak design S is defined as follows:

$$S \coloneqq \left\{ S_p \;\middle|\; p \in \mathbb{F}_t^{\leq (\log \ell / \log t) - 1}[x] \right\}$$
$$S_p \coloneqq \{(a, p(a)) \mid a \in \mathbb{F}_t\}$$

Given a subset index $1 \leq i \leq \ell$, we can output the set S_i in time $T = \mathsf{poly}(\log \ell, t)$ and space $S = O(\log \ell)$. View i as its length $\log \ell$ bit-wise representation, and divide this bitstring into $\log \ell / \log t$ parts. The k-th part represents the coefficient of the i-th polynomial p. Then, for every $a \in \mathbb{F}_t$, map $(a, p(a))$ to $[d]$. The collection of these mappings is S_i. For the argument that the above construction is a weak design, see the original proof [HR03].

References

[CDS11] Mahdi Cheraghchi, Fredric Didier, and Amin Shokrollahi. Invertible extractors and wiretap protocols. *IEEE Transactions on Information Theory*, 58(2):1254–1274, 2011.

[CKOS22] Nishanth Chandran, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Short leakage resilient and non-malleable secret sharing schemes. In *Annual International Cryptology Conference*, pages 178–207. Springer, 2022. `https://eprint.iacr.org/2022/216.pdf`.

[CW77] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112, 1977.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

[HR03] Tzvika Hartman and Ran Raz. On the distribution of the number of roots of polynomials and explicit weak designs. *Random Struct. Algorithms*, 23(3):235–263, 2003.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 12–24, New York, NY, USA, 1989. Association for Computing Machinery.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *30th Annual Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[RRV99] Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. *Electron. Colloquium Comput. Complex.*, TR99-046, 1999.

[Tre99] Luca Trevisan. Construction of extractors using pseudo-random generators (extended abstract). In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 141–148. ACM, 1999.

[Wik24] Wikipedia contributors. Method of conditional probabilities — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Method_of_conditional_probabilities&oldid=1214641630`, 2024. [Online; accessed 5-January-2025].