Some Notes on Randomness Extraction

Christopher Smith

Last Updated: November 26, 2024

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:*

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

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). *A keyed hash function, or, equivalently, a family of hash functions $\left\{H_s : \{0,1\}^n \to \{0,1\}^\ell\right\}_{s \in S}$ is called universal if, for every $x, y \in \{0,1\}^n$ with $x \neq y$,*

$$\Pr_{s \leftarrow S}[H_s(x) = H_s(y)] \leq 2^{-\ell} \tag{6}$$

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,*

$$\mathbf{SD}((H_S(W), S)\ ,\ (U_\ell, S)) \leq \frac{1}{2}\sqrt{2^{-\mathbf{H}_\infty(W)} 2^\ell} \tag{7}$$

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,*

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

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

3 Two Matrix Constructions of Universal Hash Functions

Claim 10 (Matrix hashing is universal). *Interpret the set $\mathcal{M} = \{0,1\}^{\ell \times n}$ as the set of all ℓ by n matrices over $\mathbb{F}_2$. Define a hash family $\left\{H_M : \{0,1\}^n \to \{0,1\}^\ell\right\}_{M \in \mathcal{M}}$ by $H_M(x) = Mx$, where x is interpreted as a length n vector over $\mathbb{F}_2$. This hash family is universal.*

Proof. Let $x \neq y \in \{0,1\}^n$. We wish to show that $\Pr_M[Mx = My] \leq 2^{-\ell}$. See that the event $Mx = My$ implies $M(x-y) = 0$. Because $(x-y) \neq 0$, there exists an index j^* such that $(x-y)_{j^*} = 1$. Let $M_i = (m_{i,j})_{j \in [n]}$ denote the i-th row of M. We have

$$\begin{aligned}
&\Pr_{M \leftarrow \mathcal{M}}[M(x-y) = 0] = \Pr_{M \leftarrow \mathcal{M}}[M_i(x-y) = 0\ \forall i \in [\ell]\ ] \\
&= \prod_{i \in [\ell]} \Pr_{M_i \leftarrow \{0,1\}^n}[M_i(x-y) = 0] && (M_i \text{ chosen independently}) \\
&= \prod_{i \in [\ell]} \Pr_{M_i \leftarrow \{0,1\}^n}\left[m_{i,j^*} = \sum_{j \neq j^*} m_{i,j}(x-y)_j\right] && ((x-y)_{j^*} = 1) \\
&= \prod_{i \in [\ell]} \Pr_{m_{i,j^*} \leftarrow \{0,1\}}[m_{i,j^*} = \text{arbitrary bit}] && (m_{i,j^*} \text{ uniform and independent of all } m_{i,j \neq j^*}) \\
&= 2^{-\ell}
\end{aligned}$$

□

Notice that the key/seed for matrix hashing is the description of the matrix M, which requires $n \cdot \ell$ bits. As observed in [HILL99], we can reduce this requirement to $n + l - 1$ bits by using Toeplitz matrices. A matrix $M = (m_{i,j}) \in \mathbb{F}_2^{\ell \times n}$ is Toeplitz if it is constant on its diagonals: for all i, j, $m_{i,j} = m_{i+1,j+1} = \alpha_{i-j}$, where $\alpha_{1-n}, \alpha_{2-n}, \ldots, \alpha_0, \alpha_1, \ldots, \alpha_{\ell-1}$ are the $n + \ell - 1$ constants.

Claim 11 (Toeplitz matrix hashing is universal). *Consider the same matrix hash family as in Claim 10, except we now require $\mathcal{M}$ to be the set of all Toeplitz matrices over $\mathbb{F}_2$. This hash family is universal.*

Proof. Let $x \neq y \in \{0,1\}^n$. We wish to show that $\Pr_M[Mx = My] \leq 2^{-\ell}$. See that the event $Mx = My$ implies $M(x-y) = 0$. Because $(x-y) \neq 0$, there exists an index j^* such that $(x-y)_{j^*} = 1$. Let $M_i = (m_{i,j})_{j \in [n]}$ denote the i-th row of M. We have

$$\begin{aligned}
&\Pr_{M \leftarrow \mathcal{M}}[M(x-y) = 0] = \Pr_{M \leftarrow \mathcal{M}}[M_i(x-y) = 0\ \forall i \in [\ell]\ ] \\
&= \Pr_{M \leftarrow \mathcal{M}}\left[m_{i,j^*} = \sum_{j \neq j^*} m_{i,j}(x-y)_j\ \forall i \in [\ell]\right] && ((x-y)_{j^*} = 1) \\
&= \Pr_{(m_{1,j^*}, \ldots, m_{\ell,j^*})}[m_{i,j^*} = \text{arbitrary bit}\ \forall i \in [\ell]\ ] && (m_{i,j^*} \text{ uniform and independent of all } m_{i,j \neq j^*}) \\
&= \prod_{i \in [\ell]} \Pr_{m_{i,j^*} \leftarrow \{0,1\}}[m_{i,j^*} = \text{arbitrary bit}] && (m_{i,j^*} \text{ independent of } m_{i' \neq i, j^*}) \\
&= 2^{\ell}
\end{aligned}$$

□

Remark 12. *Notice the proof of Claim 11 is almost the same as that of Claim 10, except we just had to be more careful about how we invoke independence. In Claim 10 we first appealed to independence* between rows *of a random matrix, and then to independence of elements* within a row. *In Claim 11 we cannot appeal to independence* between rows *since the rows of a random Toeplitz matrix are not independent. Instead, we first appealed to independence of elements* within a row *of a random Toeplitz matrix, and then we appealed to independence of elements* within a column *of a random Toeplitz matrix.*

4 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 13 (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. □

A more concrete description of InvExt**.** Because $\mathsf{Ext}(\cdot, s)$ is a linear map, we may assume that the seed s describes a matrix M such that $\mathsf{Ext}(w, s) = Mw$. For simplicity we additionally assume $M \in \mathbb{F}_2^{\ell \times n}$. Below we provide an equivalent but more concrete description of InvExt more amenable to implementation.

```
$\mathsf{InvExt}(y \in \{0,1\}^\ell, s \in \{0,1\}^d) \to \{0,1\}^n \cup \{\bot\}$:
  • Recover the matrix $M \in \mathbb{F}_2^{\ell \times n}$ from $s$
  • Let $[E|y'] \in \mathbb{F}_2^{\ell \times (n+1)}$ be the matrix obtained by performing row reduction on the augmented matrix $[M|y]$.
  • If $E$ has a zero row, say row $i$, but the element in the $i$-th row of $y'$ is non-zero, then $Mx = y$ has no solutions, so $y \notin \mathcal{I}_s$, and we output $\bot$
  • Else ($y \in \mathcal{I}_s$):
      – Use $[E|y']$ to back-solve for (arbitrary) $w$ such that $Mw = y$
      – Use $[E|y']$ to uniformly sample from $\mathcal{K}_s$.
      – Output $w + z$
```

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`.

[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.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[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.