Lecture 5: Complexity-theoretic hash-functions Extractors, Universal hash-functions and more

Christopher Brzuska

February 5, 2024

1 Complexity-theoretic hash-functions: theory and practice

In the first couple of weeks of this course, we looked into cryptographic hash-functions, and we saw that building collision-resistance hash-functions from standard one-way functions seems difficult, at least in a black-box way. In turn, we saw that given a normal one-way function f, we can build a pre-image resistance hash-function by applying an extractor to the input first:

$$(x, s) \mapsto (f(\mathsf{ext}(x, s)), s)$$

The extractor ensures that if x is drawn from a high-entropy distribution, then the output is almost uniformly distributed. And thus, afterwards, we can simply rely on the security of the one-way function.

Today, we are going to see how to construct such extractors. What I find exciting is that, in fact, we can *prove* that randomness extractors exist, very different from our typical crypto scenario where we always need to rely on assumptions. The approach will be to take a complexity-theoretic hash-function, known as a *2-universal hash-function* and show that it is a good extractor. This theorem (or lemma...) is also known as the *leftover hash-lemma* which is an important lemma both in complexity-theory and (theoretical) cryptography.

Now, is this all theory, or are extractors also used in practice? Interestingly, we all use extractors everyday when we check our eMails or visit a website using https. The underlying security protocol, Transport Layer Security (TLS), allows to rely on different types of key material, namely symmetric keys and Diffie-Hellman secrets. When both are available, then the protocol first extracts from them, and then combines them into a key together. The extraction process is very important, because TLS is so widely used, and one cannot really be sure how well the secret keys have been generated, since they come from some other application.

TLS also derives multiple keys from one key, and the approach used for this derivation is known as the *extract-then-expand* appraoch where TLS first runs an extractor and then uses a pseudorandom function (PRF) to expand the key into several. The extract-then-expand approach was proposed by Hugo Krawczyk, one of Oded Goldreich's first PhD students, and a solid researcher in theoretical computer science. Hugo wrote a paper `https://eprint.iacr.org/2010/264` and later an IETF standard `https://datatracker.ietf.org/`

doc/html/rfc5869 on *hashed key derivation functions* (HKDF), and HKDF is used as an essential building block in the (quite new) TLS 1.3 standard https://datatracker.ietf.org/doc/html/rfc8446.
Since TLS 1.3 is considered to be top state-of-the-art in terms of key derivations, the messaging layer security (MLS) standard also uses a very similar approach for key derivations https://datatracker.ietf.org/doc/draft-ietf-mls-protocol/.
(If you are curious about this, you can reach out. Together with colleagues, we analyzed both, the TLS 1.3 key derivations https://eprint.iacr.org/2021/467 and the MLS key derivations https://eprint.iacr.org/2021/137. The latter article is based on the master thesis of Eric Cornelissen who was a SECCLO student at Aalto.)
Now, TLS 1.3 does not use one of these complexity-theoretic extractors for extraction, but instead uses a cryptographic hash-function for extraction also. One of the reasons that one can consider for this is that in TLS 1.3., it is hard to ensure that the seed s used for extraction is independent from the symmetric-key and uniformly random. Thus, TLS 1.3. uses a *fixed* seed in some instances, and one of the heuristic arguments in favor of a cryptographic hash-function is that for a cryptographic hash-function, it is much harder to exploit relations between inputs and outputs than for complexity-theoretic extractors.
Let us now move to extractors—but before we can go to the proof, we'll need some technical tools, so we know them beforehand and don't need to jump to understanding the tools in the middle of the proof. This is what Section 2 is about.

2 Statistical Distance and Collision Probability

We recall from Pihla's lecture last week the definition of statistical distance (also called total variation distance or the distance induced by the $|.|_1$-norm). Statistical distance is useful to describe a statistical analogue of computational indistinguishability. I.e., let X and Y be PPT algorithms that get as input the security parameter 1^λ and output some bitstring. If the statistical distance between the induced distributions $X(1^\lambda)$ and $Y(1^\lambda)$ is a negligible function in λ, then no efficient or even inefficient adversary can distinguish between $X(1^\lambda)$ and $Y(1^\lambda)$ with more than negligible probability.

Definition 2.1 (Statistical Distance)**.** For two random variables X and Y, we define the statistical distance between X and Y as

$$\mathsf{SD}(X,Y) := \tfrac{1}{2} \sum_{z \in \mathrm{Supp}(X) \cup \mathrm{Supp}(Y)} |\Pr[X=z] - \Pr[Y=z]|,$$

where the support $\mathrm{Supp}(X)$ denotes the set of values z where $\Pr[X=z] > 0$.

Lemma 1. (Properties of statistical distance)
Triangle Inequality. For three random variables X, Y, and Z, we have

$$\mathsf{SD}(X,Z) \leq \mathsf{SD}(X,Y) + \mathsf{SD}(Y,Z).$$

Computational Distance. Let X and Y be PPT algorithms that get as input the security parameter 1^λ and output some bitstring. Then, for all probabilistic algorithms $\mathcal{A}$ (both efficient and inefficient), we have that

$$\left|\Pr\left[1 = \mathcal{A}(1^\lambda, X(1^\lambda))\right] - \Pr\left[1 = \mathcal{A}(1^\lambda, Y(1^\lambda))\right]\right| \leq \mathsf{SD}(X(1^\lambda), Y(1^\lambda)).$$

Often, it is difficult to directly compute the statistical distance, and there are many ways of relating statistical distance to other notions of distance. For example, it is often useful to determine how far a distribution is from the *uniform* distribution. In cryptography, this is particularly important, because we often need to construct (pseudo-)random strings as keys, and in the extractors which we construct today, we will also want to say that their output is close to uniform. One indirect way of assessing the statistical distance of a random variable from the uniform distribution is by looking at the *collision probability*, i.e., when sampling twice from X (let us denote by X' an independent copy from X), independently, from the same distribution, what is the probability that one yields the same sample?

$$\begin{aligned}
&\Pr_{z\leftarrow\$X, z'\leftarrow\$X'}[z = z'] \\
&= \sum_{x\in\mathrm{Supp}(X)} \Pr_{z\leftarrow\$X, z'\leftarrow\$X'}[x = z = z'] \\
&= \sum_{x\in\mathrm{Supp}(X)} \Pr[X = x]\cdot\Pr[X' = x] \\
&= \sum_{x\in\mathrm{Supp}(X)} (\Pr[X = x])^2
\end{aligned}$$

This is indeed the sum of squares of the probabilities.

Definition 2.2 (Collision Probability)**.** Let X be a random variable, then its collision probability is defined as

$$\mathsf{CP}(X) := \sum_{z\in\mathrm{Supp}(X)} (\Pr X = z)^2$$

The uniform distribution U_n over $\{0,1\}^n$ has collision-probability 2^{-n}, since

$$\sum_{x\in\{0,1\}^n} \Pr[U_n = x]^2 = \sum_{x\in\{0,1\}^n} (2^{-n})^2 = \sum_{x\in\{0,1\}^n} 2^{-2n} = 2^n\cdot 2^{-2n} = 2^{-n}.$$

In turn, *any other* distribution over $\{0,1\}^n$ will have higher collision probability.[1]

Lemma 2 (Bounding SD from uniform via CP)**.** Let X be a distribution over $\{0,1\}^n$, and let U_n denote the uniform distribution over $\{0,1\}^n$. Then, we have that

$$\mathsf{SD}(X, U_n) \leq \tfrac{1}{2} 2^{\frac{n}{2}} \sqrt{\mathsf{CP}(X) - 2^{-n}}$$

Proof. In this proof, we will view the distribution X over bitstrings $\{0,1\}^n$ as a vector with 2^n entries $p_x := \Pr[X = x]$, one for each $x \in \{0,1\}^n$ such that $\sum_{x\in\{0,1\}^n} p_x = 1$. In this view, we can write the statistical distance as a norm of the difference of two vectors, namely

$$\mathsf{SD}(X, U_n) = \tfrac{1}{2}||X - U_n||_1,$$

[1] To see this intuitively, observe that if $a+b = 1$, then a^2+b^2 is minimized when $a = b = \frac{1}{2}$, and maximized when $a = 1$ and $b = 0$. More rigorously, we can use that $f : x \mapsto x^2$ is a convex function.

where $||v||_1 := \sum_{x\in\{0,1\}^n}|v_x|$ is the 1-norm in a 2^n-dimensional vector space. We can then use the known relation between the 1-norm and 2-norm, namely $||v||_1 \leq \sqrt{2^n}\cdot\sqrt{||v||_2}$, where $||v||_2 := \sqrt{\sum_{x\in\{0,1\}^n}(v_x)^2}$. Thus,

$$\mathsf{SD}(X,U_n) = \tfrac{1}{2}||X-U_n||_1 \leq \tfrac{1}{2}2^{\frac{n}{2}}||X-U_n||_2. \tag{1}$$

Now,

$$\begin{aligned}
&(||X-U_n||_2)^2\\
=&\sum_{x\in\{0,1\}^n}(\Pr[X=x]-\Pr[U_n=x])^2\\
=&\sum_{x\in\{0,1\}^n}(\Pr[X=x])^2-2\Pr[X=x]\cdot\Pr[U_n=x]+\Pr[U_n=x]^2\\
=&\sum_{x\in\{0,1\}^n}(\Pr[X=x])^2+\sum_{x\in\{0,1\}^n}\Pr[U_n=x]^2-2\cdot 2^{-n}\cdot\sum_{x\in\{0,1\}^n}\Pr[U_n=x]\\
=&\mathsf{CP}(X)+\mathsf{CP}(U_n)-2\cdot 2^{-n}=\mathsf{CP}(X)-2^{-n}
\end{aligned} \tag{2}$$

Putting (2) and (1), we obtain

$$\mathsf{SD}(X,U_n)\leq\tfrac{1}{2}2^{\frac{n}{2}}\sqrt{\mathsf{CP}(X)-2^{-n}}$$

as required. □

Finally, let us recall the definition of min-entropy, i.e., the logarithm of the value with the highest probability (in absolute values).

Definition 2.3 (Min-entropy)**.** The min-entropy $H_\infty(X)$ of a random variable X is defined as

$$H_\infty(X) := \min_{x\in\mathrm{Supp}(X)}|\log_2(\Pr[X=x])|.$$

3 Extractors

3.1 Definition

We can now move to the formal definition of extractors. Intuitively, we want that if X is a source with min-entropy k, then an extractor extracts (almost) k (almost) uniformly random bits from X, i.e., when S is a uniformly random seed and U_m denotes the uniform distribution over $\{0,1\}^m$ (where m is just a little bit smaller than k), then the statistical distance of $(\mathsf{ext}(X,S),S)$ and (U_m,S) should be small.

Definition 3.1 (Strong extractor)**.** A function $\mathsf{ext}:\{0,1\}^n\times R_m\to\{0,1\}^m$ is an (k,ϵ)-strong extractor if for all random variable X with $H_\infty(X)\geq k$, we have that

$$\mathsf{SD}((\mathsf{ext}(X,S),S),(U_m,S))\leq\epsilon.$$

Here, the set R_m is a set of randomness which depends on m.

This definition is referred to as *strong* extractors since closeness is required even when the seed S is given while a weaker notion of extractor compares $\mathsf{ext}(S,X)$ alone to $U_{k'}$.

3.2 Construction

We will build an extractor based on a complexity-theoretic hash-function, namely a *2-universal* hash-function. The idea is that a hash-function should behave *random*. Again, as for the extractor case, we cannot expect a fixed function to behave random, so we sample a key S from a randomness set R_m, and we want that *any* distinct pair of inputs x and x' will be mapped to two independent output values, i.e., all pairs of output values in $\{0,1\}^m$ are equally likely and thus have probability 2^{-2m}.

Definition 3.2 (2-universal hash-function)**.** A function $h : \{0,1\}^n \times R_m \to \{0,1\}^m$ is a 2-universal hash-function if for all $z, z' \in \{0,1\}^m$ and for all distinct $x, x' \in \{0,1\}^n$, we have that

$$\Pr_{S \leftarrow\$ R_m}[h(x,S) = z \ \wedge \ h(x',S) = z'] = 2^{-2m}$$

2-universal hash-functions are also called *pairwise independent hash-functions*. The syntax of a 2-universal hash-function and an extractor is very similar, and indeed, we will see that a 2-universal hash-function is an extractor, and indeed a very good one. This fact is known as the *leftover hash lemma* which is a useful and popular statement in complexity theory and cryptography.

Lemma 3 (Leftover Hash Lemma)**.** Let $h : \{0,1\}^n \times R_m \to \{0,1\}^m$ be a 2-universal hash-function. Then h is also a strong (k,ϵ) extractor as long as $k \geq m + 2 \cdot \log_2(\frac{1}{\epsilon})$.

Proof. Let n, m, k and ϵ be such that $k \geq m + 2 \cdot \log_2(\frac{1}{\epsilon})$. And let X be a random variable over $\{0,1\}^n$ such that $H_\infty(X) \geq k$. Moreover, assume that $R_m = \{0,1\}^m$ (for simplicity).
In this proof, we want to bound the statistical distance between $(h(X,U_m)||U_m)$ and U_{2m}, where U_m denotes the uniform distribution over $\{0,1\}^m$ and U_{2m} denotes the uniform distribution over $\{0,1\}^{2m}$. In order to bound this statistical distance, we want to use Lemma 2, so if we can show that

$$\mathsf{CP}(h(X,U_m)||U_m) \leq c, \tag{3}$$

then Lemma 2 in dimension $2m$ implies that

$$\mathsf{SP}(h(X,U_m)||U_m), U_{2m}) \leq \tfrac{1}{2} 2^m \sqrt{c - 2^{-2m}}.$$

For our lemma, we need that

$$\tfrac{1}{2} 2^m \sqrt{c - 2^{-2m}} \leq \epsilon,$$

and for simplicity, let's just look at values c where it even holds that

$$2^m \sqrt{c} \leq \epsilon.$$

This holds for all values c such that

$$c \leq 2^{2m} \cdot \epsilon^2.$$

Thus, all remains to prove Lemma 3 is to prove Inequality 3 for $c \leq 2^{2m} \cdot \epsilon^2$. In the following, we denote by X' and independent copy of X and by U'_m and independent copy of U_m

$$\begin{aligned}
& \mathsf{CP}(h(X, U_m)||U_m) \\
\stackrel{\text{def}}{=} & \Pr_{x \leftarrow\$ X, x' \leftarrow\$ X', r \leftarrow\$ U_m, r' \leftarrow\$ U'_m}[h(x,r) = h(x',r') \wedge r = r'] \\
= & \Pr_{x \leftarrow\$ X, x' \leftarrow\$ X', r \leftarrow\$ U_m, r' \leftarrow\$ U'_m}[h(x,r) = h(x',r') \mid r = r'] \cdot \Pr_{r,r'}[r = r'] \\
= & 2^{-m} \Pr_{x \leftarrow\$ X, x' \leftarrow\$ X', r \leftarrow\$ U_m}[h(x,r) = h(x',r)] \\
= & 2^{-m} (\Pr_{x \leftarrow\$ X, x' \leftarrow\$ X', r \leftarrow\$ U_m}[h(x,r) = h(x',r) \mid x \neq x'] \cdot \Pr[x \neq x'] \\
& + \Pr_{x \leftarrow\$ X, x' \leftarrow\$ X', r \leftarrow\$ U_m}[h(x,r) = h(x',r) \mid x = x'] \cdot \Pr[x = x']) \\
= & 2^{-m} (\Pr_{x \leftarrow\$ X, x' \leftarrow\$ X', r \leftarrow\$ U_m}[h(x,r) = h(x',r) \mid x \neq x'] \cdot (1 - \mathsf{CP}(X)) \\
& + 1 \cdot \mathsf{CP}(X)) \\
\stackrel{\text{2-uni-hash}}{=} & 2^{-m} (2^{-2m} \cdot (1 - \mathsf{CP}(X)) + \mathsf{CP}(X))
\end{aligned}$$

□

4 Distributional One-Way Functions

Definition 4.1 (Distributional One-Way Function)**.** A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is a *distributional one-way function* if there exists a positive polynomial $p : \mathbb{N} \to \mathbb{R}_0^+$ such that for all efficient adversaries $\mathcal{A}$ and large enough λ, the following two distributions have statistical distance at least $1/p(\lambda)$:

Real(λ)	Adversarial(λ)
$x \leftarrow\$ \{0,1\}^\lambda$	$x \leftarrow\$ \{0,1\}^\lambda$
$y \leftarrow f(x)$	$y \leftarrow f(x)$
return (y, x)	$x' \leftarrow\$ \mathcal{A}(1^\lambda, y)$
	return (y, x')