0368.4162 Foundations of Cryptography November 9, 2017

Lecture 3: Pseudorandomness and One-Way Functions

Lecturer: Nir Bitansky *Scribes: Eliran Kachlon, Ron Mokady*

1 Previously on Foundations of Crypto

Starting from an encryption scheme (E, D) for messages of length $n + 1$ we constructed a scheme (E', D') for messages of any polynomial length $\ell'(n) = n^{O(1)}$, where

$$E'_{sk}(x) \qquad = \qquad E_{sk=sk_0}(sk_1, x_1) \qquad E_{sk_1}(sk_2, x_2) \qquad \dots \qquad E_{sk_{\ell'-1}}(sk_{\ell'}, x_{\ell'})$$

We wanted to show that this is secure. We'll give a slightly different proof from last week. The proof is based on exactly the same idea, but will be lighter on notation.

Recall that we need to show that for any two sequences of ℓ'-bit messages $x = \{x_n\}_{n\in\mathbb{N}}$ and $y = \{y_n\}_{n\in\mathbb{N}}$, it is the case that

$$\{E'_{sk}(x_n) \mid sk \leftarrow \{0,1\}^n\}_{n\in\mathbb{N}} \approx_c \{E'_{sk}(y_n) \mid sk \leftarrow \{0,1\}^n\}_{n\in\mathbb{N}} \ .$$

Another convenient notation for this (which will save us some double subscripts) is

$$\{E'_{sk}(x) \mid sk \leftarrow \{0,1\}^n\}_{\substack{n\in\mathbb{N}\\ x,y\in\{0,1\}^{n+1}}} \approx_c \{E'_{sk}(y) \mid sk \leftarrow \{0,1\}^n\}_{\substack{n\in\mathbb{N}\\ x,y\in\{0,1\}^{n+1}}} .^{1}$$

First, note that it's sufficient to show that there exists a distribution ensemble $S = \{S_n\}_n$ such that

$$\{E'_{sk}(x) \mid sk \leftarrow \{0,1\}^n\}_{\substack{n\in\mathbb{N}\\ x\in\{0,1\}^{n+1}}} \approx_c \{S_n\}_{\substack{n\in\mathbb{N}\\ x\in\{0,1\}^{n+1}}} .$$

This is actually an equivalent definition for secure encryption, similar to the one we had for perfect security, and we can prove it using the fact that computational indistinguishability is transitive:

$$\{E'_{sk}(x) \mid sk \leftarrow \{0,1\}^n\}_{\substack{n\in\mathbb{N}\\ x,y\in\{0,1\}^{n+1}}} \approx_c \{S_n\}_{\substack{n\in\mathbb{N}\\ x,y\in\{0,1\}^{n+1}}} \approx_c \{E'_{sk}(y) \mid sk \leftarrow \{0,1\}^n\}_{\substack{n\in\mathbb{N}\\ x,y\in\{0,1\}^{n+1}}} .$$

So to prove the security of (E', D'), it suffices to show such an ensemble S and prove that it's indistinguishable from encryptions of arbitrary messages. We define S as follows;

$$S_n \qquad = \qquad E_{sk=sk_0}(0^{n+1}) \qquad E_{sk_1}(0^{n+1}) \qquad \dots \qquad E_{sk_{\ell'-1}}(0^{n+1})$$

Assume toward contradiction that there is a non-uniform PPT attacker A' and polynomial $p(\cdot)$ such that, for infinitely many n, there exists $x \in \{0,1\}^{\ell'(n)}$ such that:

$$\Delta_{A'}(E'_{sk}(x), S_n) := |\Pr[A'(E'_{sk}(x)) = 1] - \Pr[A'(S_n) = 1]| \geq 1/p(n) \ .$$

We will turn it into a non-uniform PPT attacker A that (for the same infinitely many n) will break (E, D) (see figure 1).

Fix such n and x, we define the following hybrid distributions: for every $0 \leq i \leq \ell'$:

$$H_i := E_{sk_0}(0^{n+1}), \dots, E_{sk_{i-1}}(0^{n+1}), E_{sk_i}(sk_{i+1}, x_{i+1}), \dots, E_{sk_{\ell'-1}}(sk_{\ell'}, x_{\ell'}) \ .$$

[1] This notation may seem weird at first. The distribution on the left doesn't depend on y and that on the right doesn't depend on x. It makes sense when we look at the entire indistinguishability and interpret it as follows. For any non-unfiorm PPT A there exists a negligible μ such that for any $n \in \mathbb{N}, x, y \in \{0,1\}^{n+1}$: $\Delta_A(E'_{sk}(x), E'_{sk}(y)) \leq \mu(n)$.

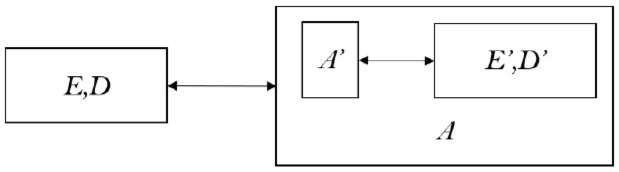

Figure 1: An efficient adversary A' against (E', D') can be translated to an efficient adversary A against (E, D).

According to the previous definition we have:

$$\begin{array}{rlllllll}
 & H_0 & = E_{sk_0}(sk_1, x_1) & E_{sk_1}(sk_2, x_2) & \dots & & & E_{sk_{\ell'-1}}(sk_{\ell'}, x_{\ell'}) \\
 & H_1 & = E_{sk_0}(0^{n+1}) & E_{sk_1}(sk_2, x_2) & \dots & & & E_{sk_{\ell'-1}}(sk_{\ell'}, x_{\ell'}) \\
 & \vdots & \quad\vdots & \vdots & \vdots & \vdots & & \vdots \\
\hline
i-1 & H_{i-1} & = E_{sk_0}(0^{n+1}) & E_{sk_1}(0^{n+1}) & \dots\ \boldsymbol{E_{sk_{i-1}}(sk_i, x_i)} & E_{sk_i}(sk_{i+1}, x_{i+1}) & \dots & E_{sk_{\ell'-1}}(sk_{\ell'}, x_{\ell'}) \\
i & H_i & = E_{sk_0}(0^{n+1}) & E_{sk_1}(0^{n+1}) & \dots\ \boldsymbol{E_{sk_{i-1}}(0^{n+1})} & E_{sk_i}(sk_{i+1}, x_{i+1}) & \dots & E_{sk_{\ell'-1}}(sk_{\ell'}, x_{\ell'}) \\
\hline
 & \vdots & \quad\vdots & \vdots & \vdots & \vdots & & \vdots \\
 & H_{\ell'} & = E_{sk_0}(0^{n+1}) & E_{sk_1}(0^{n+1}) & \dots & & & E_{sk_{\ell'-1}}(0^{n+1})
\end{array}$$

Note that $H_0 = E'_{sk=sk_0}(x)$ and $H_{\ell'} = S_n$.

Claim 1.1. *There exists $i \in [\ell']$ such that $\Delta_{A'}(H_{i-1}, H_i) \geq \frac{1}{p(n) \cdot \ell'(n)}$.*

Proof.

$$\frac{1}{p(n)} \leq \Delta_{A'}(E'_{sk}(x), S_n) = \Delta_{A'}(H_0, H_{\ell'}) \leq \sum_{i \in [\ell']} \Delta_{A'}(H_{i-1}, H_i) \leq \max_{i \in [\ell']} \Delta_{A'}(H_{i-1}, H_i) \cdot \ell'(n) \ ,$$

where the first inequality follows from the definition of A' and the second inequality from the triangle inequality. □

Notice that H_{i-1} and H_i only differ on the ith encryption:

- In H_{i-1}, we have $E_{sk_{i-1}}(sk_i, x_i)$,
- In H_i, we have $E_{sk_{i-1}}(0^{n+1})$.

We now need to construct an adversary A that breaks E. More precisely, we will show that for a random $sk_i \leftarrow \{0,1\}^n$

$$\Delta_A((E_{sk}(sk_i, x_i), sk_i), (E_{sk}(0^{n+1}), sk_i)) \geq \frac{1}{\ell'(n) \cdot p(n)} \ .$$

(Make sure you understand why this is enough.)

Adversary A: Given a ciphertext ct, and sk_i, A will sample by himself all the ciphertexts, except for the ith one, and plant ct as the ith cipher, in the following way:

$$\underbrace{E_{sk_0}(0^{n+1})}_{ct_1} \quad \underbrace{E_{sk_1}(0^{n+1})}_{ct_2} \quad \dots \quad \underbrace{ct}_{ct_i} \quad \underbrace{E_{sk_i}(sk_{i+1}, x_{i+1})}_{ct_{i+1}} \quad \dots \quad \underbrace{E_{sk_{\ell'-1}}(sk_{\ell'}, x_{\ell'})}_{ct_{\ell'}}$$

It will then run A' on this sample and output the same bit.

Note that if $ct \leftarrow E_{sk}(sk_i, x_i)$ then the sample is distributed exactly like H_{i-1} and if $ct \leftarrow E_{sk}(0^{n+1})$ then it is distributed exactly like H_i. Crucially, in both hybrids $ct_1, \dots, ct_{i-1}, ct_{i+1}, \dots, ct_{\ell'}$ are completely independent of sk_{i-1} and have the same distribution. (Notice that to sample ct_{i+1}, we use the input sk_i.)

It is left to observe that:

$$\Delta_A((E_{sk}(sk_i, x_i), sk_i), (E_{sk}(0^{n+1}), sk_i)) = \Delta_{A'}(H_{i-1}, H_i) \geq \frac{1}{\ell'(n) \cdot p(n)} \ .$$

Indeed, when $ct \leftarrow E_{sk}(sk_i, x_i)$, the emulated A' sees exactly the distribution H_{i-1} and when $ct \leftarrow E_{sk}(0^{n+1})$ it sees exactly H_i. □

Remember that our goal is to reduce crypto, and in particular encryption, to the a few basic primitives that we could hopefully base on a host of assumptions. So after proving our first theorem, we have already reduced (secret-key) encryption for arbitrarily long messages to "one-extra-bit encryption", i.e. encryption for messages of size $n+1$. *But where would OEB encryption come from?*

We now wish to go further down the ladder of reductions and find "the right primitive(s)" that on one hand would be sufficient, and on the other can be constructed under various assumptions. A key concept in this endeavor is *pseudorandomness.*

2 Pseudorandomness

Our basic goal here is to take few random bits and use them to generate more random bits. That is, we want a procedure G that takes n random bits and maps them to $n + \ell$ bits, for some $\ell(n) > 0$. It's not hard to see that $G(U_n)$ cannot be truly random or even statistically close to random. As was the case for encryption all that we ask is that $G(U_n)$ fools bounded algorithms into thinking that its random.

Definition 2.1 (Pseudorandom Generator (PRG))**.** *A pseudorandom generator with stretch $\ell(\cdot)$ is a (deterministic) polynomial-time algorithm G that maps n bits to $n + \ell(n)$ bits and such that its output is pseudorandom:*

$$\{G(U_n)\}_n \approx_c \left\{U_{n+\ell(n)}\right\}_n \ .$$

The random input of the generator is called a **seed.**

Remark: Other Flavors of Pseudorandomness. In a wider context, PRGs as above are called cryptographic pseudorandom generators. They are quite ambitious in the sense that the same PRG should fool a very large class of algorithms — all efficient ones. As such, we'll see that it also requires crypto assumptions. Pseudorandomenss is widely studied beyond cryptography, mostly in the context of derandomization. In this context, we may gain a lot — in terms of assumptions, or seed length — if we're only interested in fooling restricted classes of algorithms, such as algorithms with some apriori bounded running time, algorithms with bounded space, only linear functions, etc. In this course, we probably won't touch non-cryptographic PRGs, but we should keep this distinction in mind.

In crypto, the purpose of PRGs is not to derandomize algorithm (in the sense of making them completely deterministic), but rather to allow cryptographic keys to be short, or more generally to *recycle randomness.* The first thing, we would like to understand is their relation to OEB encryption.

Theorem 2.2. *There exist PRGs with one bit stretch ($\ell = 1$) iff there exist OEB encryption.*

Corollary 2.3 (of theorem from previous lecture)**.** *If there exists PRGs with one bit stretch, then there exists encryption for messages of arbitrary polynomial length $\ell(n) = n^{O(1)}$ (and key size n).*

We will only prove the first direction saying the PRGs imply OEB. We will not prove the second direction for now, we'll restate it a bit later, and prove a weaker version of it.

PRGs with one-bit stretch imply OEB. The key for the encryption will be a random seed $sk := s \leftarrow \{0,1\}^n$. Encryption for a message $m \in \{0,1\}^{n+1}$ is given by:

$$E_{sk=s}(m) = m \oplus G(s) \ .$$

Decryption of a ciphertext ct is given by

$$D_{sk=s}(ct) = ct \oplus G(s) \ .$$

This construction obeys the intuition that a pseudorandom string is as good as a random string, in particular, we can use it as a one-time pad. Let's prove that this is secure (using the similar ensemble notation as we did in the beginning of this lecture):

$$\{E_{U_n}(m)\}_{\substack{n\in\mathbb{N}\\ m\in\{0,1\}^{n+1}}} = \{m \oplus G(U_n)\}_{\substack{n\in\mathbb{N}\\ m\in\{0,1\}^{n+1}}} \approx_c \{m \oplus U_{n+1}\}_{\substack{n\in\mathbb{N}\\ m\in\{0,1\}^{n+1}}} \equiv \{U_{n+1}\}_{\substack{n\in\mathbb{N}\\ m\in\{0,1\}^{n+1}}} \ .$$

(Make sure you understand why the 2nd and the 3rd ensembles are indistinguishable). □

Longer Stretch. Not surprisingly, given a PRG with stretch $\ell = 1$ we can get a PRG with arbitrary polynomial stretch $\ell'(n) = n^{O(1)}$. The construction and its proof are very similar to extending encryption. In a picture it looks as follows:

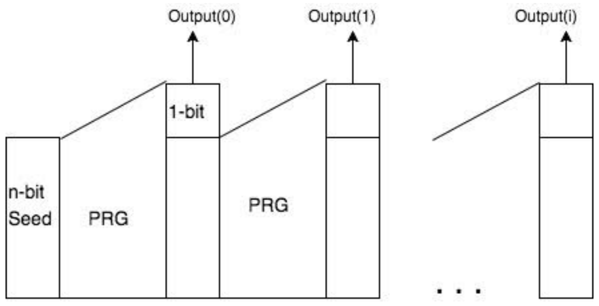

Figure 2: PRG-stretching illustration

Given a random seed $s_0 := s$, we apply it to the PRG and get an output $G(s) = \sigma_1 s_1$ where $|\sigma_1| = 1$ and $|s_1| = n$. We apply s_1 to the PRG again to get $G(s_1) = \sigma_2 s_2$, and repeat this process for $\ell'(n)$ iterations. We then output $\sigma_1 \ldots \sigma_{\ell'}$. Security follows from an hybrid argument.

2.1 Where Do PRGs Come From?

Our web of reductions is starting to build:

$$\boxed{\ \ ?\ \ } \longrightarrow \boxed{\text{PRG}} \longrightarrow \boxed{\text{OEB Enc}} \longrightarrow \boxed{\text{Enc}}$$

Remember that as the source for this graph of primitives, we would like to have a primitive that we can base on a variety of hard problems. PRGs bring us closer to this, and already have direct (candidate) constructions:

1. **Practical Designs:** there exist many so called practical constructions for PRGs (see this list for example). They are mainly designed to be *fast*, and typically we do not know how to reduce their security to a better assumption than "they are secure". Sometimes, we can show that they resist certain specific attacks. Sometimes they're broken. In the context of this course, this falls short of what we want.

2. **PRGs from Well-Studied Computational Assumptions:** there are quite a few computational assumptions that some specific distribution (ensemble) is pseudorandom. Examples include the Decision Diffie-Hellman (DDH) assumption, Learning with Errors (LWE), Learning Parity with Noise (LPN), which we might see later in the course. These are typically problems that have been long studied by the CS/Math community.

3 One-Way Functions: A Minimal Cryptographic Primitive

We are going to go down even further in the ladder of reductions, with the aim of extending even further the collection of problems on which we can base PRGs, and in turn Encryption.

Definition 3.1 (One-Way Function (OWF))**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is one way if it is computable in polynomial time and for any non-uniform PPT $A = \{A_n\}_n$ there exists a negligible $\mu(\cdot)$ such that for all $n \in \mathbb{N}$:*

$$\Pr\left[A_n(f(x)) \in f^{-1}(f(x)) \;\middle|\; x \leftarrow \{0,1\}^n\right] \leq \mu(n) \ .$$

That is, a one-way function is easy to compute in the forward direction, but for random inputs is hard to invert in the reverse direction.

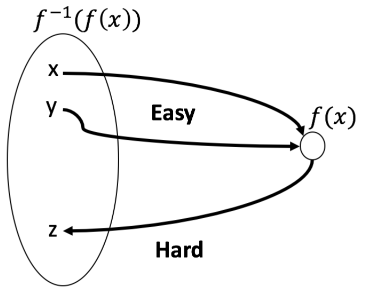

Figure 3: One-Way Function

Hard-on Average-NP Problems with Solved Instances — What OWFs Capture. Somewhat differently from the definitions we've seen so far, OWFs may seem like an odd creature. What do they capture? The answer is that OWFs capture the minimal form of computational hardness necessary for crypto. Let's try to understand in what sense. We've already seen that if P =NP, we can break encryption. However, as already hinted, it seems that we need more:

- **Average-Case Hardness:** Usually when we talk about NP hardness we mean *in the worst case.* That is every efficient algorithm would fail to solve the problem on some "worst-case instance". However, in cryptography we do not know ahead of time who is going to be our adversary, and thus need to sample hard on average problems that *all efficient algorithms* will fail to solve (except perhaps with negligible probability).
- **Solved Instances:** It's not enough that we can simply generate hard instances. The good guys must have some advantage over the bad guys — they should be able to solve the hard problems that they generate. For example, in encryption, they should be able to decrypt. In other words, we want to sample hard on average problems together with their solutions.

Definition 3.2 (Hard on Average NP Problem with Solved Instances)**.** *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be an NP relation. We say that R is hard on-average with solved instances if there exists a PPT sampler S such that:*

- S **samples solved instances:** *for any $n \in \mathbb{N}$,*

$$\Pr\left[(x,w) \in R \mid (x,w) \leftarrow S(1^n;r)\right] = 1 \ \ .$$

- S **samples hard instances:** *for any (non-uniform) PPT $A = \{A_n\}_n$ there is a negligible $\mu(\cdot)$, such that for any $n \in \mathbb{N}$,*

$$\Pr\left[w' \leftarrow A_n(x) \text{ and } (x,w') \in R \mid (x,w) \leftarrow S(1^n)\right] \le \mu(n) \ \ .$$

The following is a relatively easy exercise (make sure you know how to solve it).

Claim 3.3. *OWF exist iff hard-on-average NP problems with solved instances do.*

Proof Sketch. Given a OWF f, define $S(1^n)$ to sample $x \leftarrow \{0,1\}^n$ and output $(f(x), x)$. Given $S(1^n;r)$ for hard on average NP problem with solved instances, define $f(r) = x \leftarrow S(1^n;r)$.[2] □

As we shall see, one-way functions will indeed be necessary for essentially any crypto task we'll encounter in this course. In particular:

Theorem 3.4. *Secret-key encryption implies OWFs.*

Proof Sketch. Let (E, D) be an encryption scheme for messages of length $2n$, with keys of length n, and assume E uses $\ell = \ell(n)$ bits of randomness. We define $f : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^\ell \to \{0,1\}^*$ as follows:

$$f(sk, m, r) = (E_{sk}(m;r), m) \ \ .$$

This function is defined for inputs of length $3n + \ell$, and we'll show it is one-way for such inputs. This is enough, as we can extend the function to all inputs length by padding so that one-wayness also extends (think why).

The intuition is that if an adversary could invert this function, it must also be able to do it had we replaced the encryption of the random message m with an encryption of an independent message say 0^{2n}. However, except with negligible probability there cannot exist an encryption of 0^{2n} that decrypts to the random message m, simply because there are many more messages than keys.

Formally, we can show that given an adversary A' that inverts f, on inputs of length $3n + \ell$, with probability $\varepsilon = \varepsilon(n)$, then we can construct an adversary A with polynomial related running time that distinguishes encryptions of a random message from encryptions of 0^{2n}:

$$\Delta_A((E_{sk}(m), m), (E_{sk}(0^{2n}), m) \ge \varepsilon - 2^{-n} \ \ .$$[3]

$A(ct, m)$ given a ciphertext and message m applies $A'(ct, m)$, to obtain an (alleged) preimage (sk', m', r'), it will then output 1 iff A' successfully inverted, i.e. $f(sk', m', r') = (ct, m)$ and $m' = m$.

Indeed, note that:

- When $ct \leftarrow E_{sk}(m;r)$, then A' gets exactly a random image under the function $(ct, m) = f(sk, m, r)$, and will thus successfully invert and with probability ε.
- We'll show that when $ct \leftarrow E_{sk}(0^{2n};r)$, it can invert with probability at most 2^{-n}. In what follows, it will always be the case that $m \leftarrow \{0,1\}^{2n}, sk \leftarrow \{0,1\}^n, ct \leftarrow E_{sk}(0^{2n})$.

$$\begin{array}{lclc}
\Pr\left[\begin{array}{l}(sk', m', r') \leftarrow A'(ct, m)\\ f(sk', m', r') = (ct, m)\end{array}\right] & = & \Pr\left[\begin{array}{l}(sk', m', r') \leftarrow A'(ct, m)\\ (E_{sk'}(m';r'), m') = (ct, m)\end{array}\right] & \le \\
\Pr\left[\ \exists sk' \in \{0,1\}^n : D_{sk'}(ct) = m\ \right] & \le & \sum_{sk'} \Pr\left[D_{sk'}(ct) = m\right] & \le \ \ 2^n \cdot 2^{-2n} \ \ .
\end{array}$$

[2] This notation means: sample an instance with a solution from $S(1^n;r)$ with r as the random string, and output only the instance.

[3] As we mentioned earlier, this is enough to break encryption (you need to convince yourself why).

where the first equality follows from definition; the first inequality follows since the claim "$\exists sk' \in \{0,1\}^n : D_{sk'}(ct) = m$" follows from the claim "$(E_{sk'}(m'; r'), m') = (ct, m)$"; the second inequality follows from union bound; the last inequality follows because for any ct and sk, we have $\Pr[D_{sk}(ct) = m | m \leftarrow \{0,1\}^{2n}] \leq 2^{-2n}$, since m is distributed uniformly over $\{0,1\}^{2n}$.

$\square$