0368.4162 Foundations of Cryptography November 14, 2019

Lecture 3: Pseudorandomness and One-Way Functions

Lecturer: Nir Bitansky

1 Previously on Foundations of Crypto

Starting from an encryption scheme for messages of length $\ell(n) = n+1$ we constructed a scheme for messages of any polynomial length $\ell'(n) = n^{O(1)}$. This fits within our goal of reducing cryptography, and in particular encryption, to the a few basic primitives that we could hopefully base on a host of assumptions. For now, 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 $G(s)$ cannot be truly random or even statistically close to random (even when s is chosen at random) . As was the case for encryption all that we ask is that $G(s)$ fools bounded algorithms into thinking that its random.

Definition 2.1 (Pseudorandom Generator (PRG))**.** *Let U_k denote the uniform distribution on k-bit strings. 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 won't focus 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 ensemble notation we saw last lecutre):

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

(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:

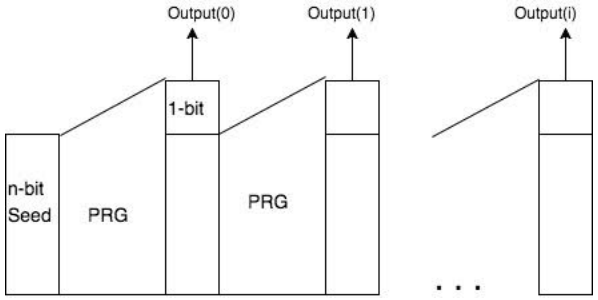
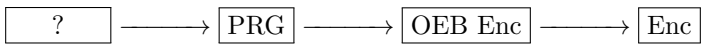


Figure 1: PRG-stretching illustration

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:



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.

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 $n \in \mathbb{N}$,*

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

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 $f(r) = x \leftarrow S(1^n; r)$.[1] □

[1] 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.

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 input lengths so that one-wayness also extends (you'll do this in your homework).

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 $\varepsilon = \varepsilon(n)$, we can construct an adversary A with polynomial related running time that distinguishes encryptions of a random message from encryptions of 0^{2n}:

$$\Delta_A((E_{sk}(m), m), (E_{sk}(0^{2n}), m) \geq \varepsilon - 2^{-n} \ .^2$$

$A(ct, m)$ 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. $f(sk', m', r') = (ct, m)$.

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 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 $m \leftarrow \{0,1\}^{2n}, sk \leftarrow \{0,1\}^n, ct \leftarrow E_{sk}(0^{2n})$.

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

  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 $\{0,1\}^{2n}$.

□

The Power of OWFs. We've argued that OWFs capture necessary properties for cryptography. The surprising part and one of the great achievements of modern cryptography is that *one-way functions are also sufficient!* at least for everything we've seen so far (and for more that's to come), which falls under the category of *secret-key crypto*.

Theorem 3.5 ([HILL99])**.** *OWFs imply PRGs!*

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

The full proof of this amazing theorem is out of the scope of this course. Nevertheless, in the next lecture, we'll develop some of the central concepts and tools that lead to this result (which will be interesting on their own), and use them to prove a weaker version.

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

4 Examples of OWFs

Before we go deeper into the above theorem, let's try to get a sense of what one-way functions could look like. Indeed, OWFs can be based on a large variety of hard computational problems. For a comprehensive list of examples from different areas see this assay by Boaz Barak, where he makes the point that if you throw a rock at random you're likely to hit a OWF. Here, we'll give just three examples from different area:

- **Factoring:**
  - Instance: $N = pq$, for two n-bit prime numbers p, q sampled at random.
  - Solution: p.

  Today the best (classical) algorithms for this problem run in time $\approx 2^{n^{\Omega(1)}}$. The problem can be solved efficiently by quantum algorithms.

- **Subset Sum:**
  - Instance: $x_1, \ldots, x_n, T$, where $x_i \leftarrow [2^n]$, and $T = \sum_{i \in I} x_i$ for a random subset $I \leftarrow 2^{[n]}$.
  - Solution: I' such that $T = \sum_{i \in I'} x_i$.

  Today the best (classical) algorithms for this problem run in time $\approx 2^{\Omega(n)}$.

- **Learning Parity with Noise:**
  - Instance: $(a_1, \langle a_1, s\rangle + e_1), \ldots, (a_{2n}, \langle a_{2n}, s\rangle + e_{2n})$, where $s \leftarrow \mathbb{F}_2^n$, $a_i \leftarrow \mathbb{F}_2^n$, e_i is 1 w.p. 0.1.
  - Solution: s (uniquely defined with overwhelming probability).

  Today the best (classical) algorithms for this problem run in time $\approx 2^{\Omega(n)}$.

5 Toward Pseudorandom Generators from One-Way Functions

Let's start by developing some intuition on how we could conceivably transform the hardness given by a OWF f into randomness. Recall that all that we need to do is stretch a random seed s by a single bit. We are going to focus on an easier case where f is a (length-preserving) permutation $f : \{0,1\}^n \to \{0,1\}^n$. [3]

In what sense is it easier? it suggests the following natural direction to constructing a PRG:

$$PRG(s) = f(s), \text{"some bit } s_i \text{ that's hard to predict from } f(s)\text{"} \ .$$

Why should there be such a bit? well if we can predict all bits of s, then we can also invert $f(s)$. While this intuition turns out to be correct for some specific candidate functions f, it is not true in general. The main problem with this intuition is that one-wayness only guarantees that we cannot simultaneously predict all bits $s_1, \ldots, s_n$. This perhaps says that some individual bit cannot be predicted w.p. (very close to) 100%, but it may be that every individual bit s_i could be predicted w.p. 99%, thus the probability of predicting *all bits simultaneously* is still negligible (try to come up with an example).

It turns out, however, that this intuition can be made correct if we slightly generalize what we mean by "a bit of s".

Definition 5.1 (Hardcore Bit (HCB))**.** *A poly-time computable function $B : \{0,1\}^* \to \{0,1\}$ is a hardcore bit of f if*

$$f(U_n), B(U_n) \approx_c f(U_n), U_1' \ .^{4}$$

One could prove that being pseudorandom is equivalent to being unpredictable:

[3] For example, such one-way permutations can be based on the hardness of the discrete logarithm problem.

[4] Above U_n, U_1' are independent. We often use a different number of $'$s to indicate independent samples from same distribution.

Definition 5.2 (Equivalent Definition)**.** *A poly-time computable function $B : \{0,1\}^* \to \{0,1\}$ is a hardcore bit of f if for every n.u. PPT A there exists a negligible μ such that:*

$$\Pr\left[A(f(U_n)) = B(U_n)\right] \leq \frac{1}{2} + \mu(n) \ .$$

Given a one-way permutation f with such a hardcore bit B, it is now clear how to get a PRG:

$$G(s) = f(s), B(s) \ .$$

So do all OWFs a HCB? Not exactly... but it turns out they do have a slightly more general version of a randomized HCB.

Definition 5.3 (Randomized Hardcore Bit (RHCB))**.** *A poly-time computable function $B : \{0,1\}^* \to \{0,1\}$ is randomized hardcore bit of f if*

$$f(U_n), B(U_n; U'_n), U'_n \approx_c f(U_n), U''_1, U'_n \ .$$

or equivalently, for every n.u. PPT A there exists a negligible μ such that:

$$\Pr\left[A(f(U_n), U'_n) = B(U_n; U'_n)\right] \leq \frac{1}{2} + \mu(n) \ .$$

Assuming one-way permutations, the above is good enough for PRGs.

Claim 5.4. *Let f be a one-way permutation and B a randomized hardcore bit for f, then*

$$G(s, s') := f(s), B(s, s'), s'$$

is a PRG.

Proof. Note that:

$$G(U_{2n}) \equiv f(U_n), B(U_n, U'_n), U'_n \approx_c f(U_n), U''_1, U'_n \equiv U_{2n+1} \ .$$

The left relation follows by the definition of G; the middle one follows by the security guarantee of hardcore predicates; and the right relation follows from the fact that $f(U_n) \equiv U_n$ for any permutation f.

This PRG is defined over inputs of length $2n$, but as in the case of OWFs, can be extended using padding. □

Remark: If you've encountered the notion of randomness extractors, you can think of such a HCB as a strong computational extractor. Given $f(s)$, s still has some form of computational entropy (even if no information-theoretic entropy), and B extracts it. It does so using a public seed s', and pseudorandomness is guaranteed even given this seed in the clear.

References

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