COS 533: Advanced Cryptography — Princeton University
Lecture 2 (February 4, 2021)
Lecturer: Mark Zhandry — Scribe: Mark Zhandry

Notes for Lecture 2

1 Last Time

Last time, we defined formally what an encryption scheme is, as well as a pseudorandom function. We also saw how to build encryption from PRFs.

2 This Time

Starting today, and for the next couple lectures, we will show how to construct encryption and other cryptographic applications from weaker tools. In particular, we will show:

1. PRGs (pseudorandom generators) $\rightarrow$ PRFs
2. OWFs (one-way functions) $\rightarrow$ PRGs
3. OWFs, or tools implied by them, $\rightarrow$ digial signatures

We won't be able to fully complete all the arrows, but we will give an indication of how it might be done. Today, we will show the first step, namely building a PRF from a PRG.

3 PRFs

A PRF is a keyed function that looks like a random function if you never get to see the key. That is PRF is a deterministic polynomial time computable function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \rightarrow \{0,1\}^{m(\lambda)}$, with the following security property.

Let A be an adversary. Let $\mathsf{PRF\text{-}EXP}_b(A, \lambda)$ be the following experiment on A, parameterized by a bit b:

1. A interacts with a challenger, denoted Ch.
2. At first, if $b = 0$, Ch chooses a random key $k \xleftarrow{\$} \{0,1\}^\lambda$. If $b = 1$, Ch initializes an empty list L.

3. Next, A sends the challenger an input $x \in \{0,1\}^{n(\lambda)}$. Ch responds as follows
   - If $b = 0$, Ch responds with $y \leftarrow \mathsf{PRF}(k, x)$.
   - If $b = 1$, Ch looks for a pair (x, y) in L. If it finds an (x, y), it responds with y. Otherwise, it generates a random y and adds the pair (x, y) to L. Then it responds with y.
4. A can repeat step 3 as many times as it wishes. We will charge A one unit of time for every time it repeats step 3.
5. Finally, A outputs a guess b' for b. b' is the output of $\mathsf{PRF\text{-}EXP}_b(A, \lambda)$

Notice that in the $b = 1$ case, Ch is effectively providing A with a truly random function O where all outputs are chosen independently and uniformly at random. In the $b = 0$ case, Ch is providing A with the PRF on a random key k. A's goal is to distinguish the two cases.

Definition 1 *An PRF* PRF secure *if, for all PPT adversaries A, there exists a negligible function ϵ such that*

$$| \Pr[1 \leftarrow \mathsf{PRF\text{-}EXP}_0(A, \lambda)] - \Pr[1 \leftarrow \mathsf{PRF\text{-}EXP}_1(A, \lambda)] | < \epsilon(\lambda)$$

4 PRGs

Next, we turn to constructing PRFs from a weaker object called a pseudorandom generator, or PRG. A PRG is a deterministic polynomial time function $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+s(\lambda)}$ for some $s(\lambda) \geq 1$. This means that G expands its input. For security, we ask that outputs of G look as if they were random.

Definition 2 *A function G is a secure PRG if, for all PPT adversaries A, there exists a negligible function ϵ such that*

$$| \Pr[A(G(x)) = 1 : x \xleftarrow{\$} \{0,1\}^\lambda] - \Pr[A(y) = 1 : y \xleftarrow{\$} \{0,1\}^{\lambda+s(\lambda)}] | < \epsilon(\lambda)$$

Notice that G can only take on at most 2^λ outputs, smaller than the $2^{\lambda+s(\lambda)}$ points in the co-domain. Therefore, it is impossible for G's outputs to be truly random. Nonetheless, PRG security says that the outputs *look* random to any polynomial-time adversary.

5 Extending the Stretch of PRGs

As a first step toward understanding how to construct PRFs from PRGs, we will show how to extend the stretch ($s(\lambda)$). In particular, we will show that from a PRG with stretch $s = 1$, we can construct a PRG of any desired (polynomial) stretch.

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$. We will write $G(x) = (G_0(x), b(x))$ where $G_0(x)$ is λ bits, and $b(x)$ is 1 bit.

For any s, we now construct a new PRG $G' : \{0,1\}^\lambda \to \{0,1\}^{\lambda+s}$. $G'(x)$ works as follows:

- Define $x_0 = x$.
- For $i = 1, \ldots, s$, let $x_i = G_0(x_{i-1})$ and $o_i = b(x_{i-1})$.
- Output $(x_s, o_1, \ldots, o_s)$.

The construction clearly is deterministic and has the desired stretch. We now prove security.

Theorem 3 *If G is a secure PRG, then so is G'.*

We prove security via a hybrid argument. Let A be a supposed adversary running in polynomial time and ϵ a non-negligible function such that

$$| \Pr[A(G'(x)) = 1 : x \xleftarrow{\$} \{0,1\}^\lambda] - \Pr[A(y) = 1 : y \xleftarrow{\$} \{0,1\}^{\lambda+s(\lambda)}] | < \epsilon(\lambda)$$

Define the following hybrid experiments. **Hybrid 0** means $y = G'(x)$ for a random $x \in \{0,1\}^\lambda$. In hybrid **Hybrid 1**, we replace x_1 and o_1 with uniform random strings, rather than computing them as $(x_1, o_1) = G(x)$. All subsequent steps are identical to G', and the y fed to A is $x_s, o_1, \ldots, o_s$. More generally, in **Hybrid** i, we sample uniformly random x_i and $o_1, \ldots, o_i$. Then we continue to compute $x_j = G_0(x_{j-1}), o_j = b(x_{j-1})$ for $j = i+1, \ldots, s$. y is then set to $x_s, o_1, \ldots, o_s$.

Note that **Hybrid** s is the setting where y is uniformly random. Therefore, A distinguishes **Hybrid 0** from **Hybrid** s with advantage ϵ. This means there is some integer $i \in \{1, \ldots, s\}$ such that

$$| \Pr[1 \leftarrow \text{Hybrid}_i(A, \lambda)] - \Pr[1 \leftarrow \mathsf{Hybrid}_{i-1}(A, \lambda)] | \geq \frac{\epsilon(\lambda)}{s(\lambda)}$$

We now are basically done, as **Hybrid** $i-1$ and **Hybrid** i differ by a single application of G. In particular, we can construct an adversary B for G as follows: On input $y \in \{0,1\}^{\lambda+1}$, let x_i be the first λ bits of y, and o_i be the last bit. Let $o_1, \ldots, o_{i-1}$ be

random. Then compute x_j, o_j for $j > i$ as in G', finally producing $z = (x_s, o_1, \ldots, o_s)$. Then run $A(z)$, and output the output of A.

If y is uniformly random, then z is distributed exactly as in **Hybrid** i. On the other hand, if $y = g(x)$ for a random x, then z is distributed exactly as in **Hybrid** $i-1$. Therefore, the advantage of B in breaking G is exactly the advantage of A in distinguishing **Hybrid** $i-1$ and **Hybrid** i, namely at least ϵ/s. This completes the proof.

6 PRFs from PRGs

We will not formally define the actual PRF algorithm, but instead will describe it in words. We will assume G is length-doubling, meaning $s(\lambda) = \lambda$. First, let's forget efficiency for the moment. The PRF PRF works as follows. It takes the key $k \in \{0,1\}^\lambda$, and applies G. The result is a 2λ-bit string. PRF splits the string in half into two λ-bit strings. Then it applies G separately to both halves, obtaining a 4λ-bit string. PRF splits this into 4 λ-bits strings, and applies G to each string. It continues in this way for $n(\lambda)$ steps, for any desired polynomial λ.

The result is a $2^{n(\lambda)} \times \lambda$-bit string. PRF will interpret this string as $2^{n(\lambda)}$ separate λ-bit strings. The output of PRF on input x with be the xth string in this list. Therefore, PRF has inputs of length $n(\lambda)$ and outputs of length λ.

It will be useful to think of the PRF computation as a tree: at the root is the PRF key k. The children of a node containing x are the first and second half of $G(x)$. The tree has $n+1$ levels and 2^n leaves. The leaves are the outputs of the PRF.

As described, PRF runs in time roughly $2^{n(\lambda)}$, which is exponential.

Question: How to compute each block locally in time polynomial in n, without computing the entire list of outputs?

Theorem 4 *If G is a secure PRG, then the construction above is a secure PRF*

This PRF construction can be seen as another form of length extension for PRGs. One attempt to prove this theorem would be to try and adapt the length extension proof to this setting. One would gradually replace each PRG application with random, until the entire tree were entirely random. Unfortunately, this will not work. As there are exponentially-many PRG applications, the number of hybrids will end up being $2^{n(\lambda)}$, resulting in an exponential loss in the security reduction. The result is an adversary for B with exponentially-small advantage (even if the starting adversary A has high advantage), which would not give a contradiction. Next time, we will see a more careful proof that overcomes this difficulty.