Introduction to Cryptography 02/13/2018

Lecture 9: Pseudorandomness III

Instructor: Vipul Goyal *Scribe: Vy Nguyen*

1 Poly-Stretch Pseudorandom Generators

Recall the construction of a **poly-stretch PRG** $G_P : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ from a 1-bit stretch PRG $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$. Given a random n-bit seed s, $G_P(s) = b_1 b_2 ... b_{l(n)}$ where

$$\begin{gathered} s_0 \leftarrow s \\ s_1 || b_1 \leftarrow G(s_0) \\ .. \\ .. \\ s_{l(n)} || b_{l(n)} \leftarrow G(s_{n-1}) \end{gathered}$$

Theorem 1 *If G is a one-bit stretch PRG, then G_P is a poly-bit stretch PRG.*

Before we proceed with the proof, let's quickly review the Hybrid Lemma.

Lemma 1 ***(Hybrid Lemma)*** *Let $X_1, ..., X_m$ be distribution ensembles for $m = poly(n)$. Suppose there exists a non-uniform PPT adversary D that can distinguish X_1 and X_m with an advantage ϵ. Then there exists $i \in \{1, ..., m-1\}$ such that D can distinguish X_i and X_{i+1} with an advantage $\geq \frac{\epsilon}{m}$.*

Now let's prove Theorem 1.

Proof. Suppose that G_P is not a poly-bit stretch PRG. Let D be a non-uniform PPT algorithm that can distinguish G_P and $U_{l(n)}$ with a noticeable advantage of at least ϵ. We will use the hybrid proof to show that this cannot be the case.

Let s be an n-bit seed selected uniformly random from $\{0,1\}^n$ and construct the hybrids as follows
-H_0: D is given $G_P(s) = b_1 b_2 ... b_{l(n)}$. Output $D(b_1 b_2 ... b_{l(n)})$.
-H_1: D is given $G_P(s) = u_1 b_2 ... b_{l(n)}$. Output $D(u_1 b_2 ... b_{l(n)})$.
-..
-H_i: D is given $G_P(s) = u_1 u_2 ... u_i b_{i+1} ... b_{l(n)}$. Output $D(u_1 u_2 ... u_i b_{i+1} ... b_{l(n)})$.
Specifically,

$$\begin{gathered} s_1 || u_1 \leftarrow \{0,1\}^{n+1} \\ .. \\ s_i || u_i \leftarrow \{0,1\}^{n+1} \\ s_{i+1} || b_{i+1} \leftarrow G(s_i) \\ .. \\ s_{l(n)} || b_{l(n)} \leftarrow G(s_{n-1}) \end{gathered}$$

-H_n: D is given $G_P(s) = u_1 u_2 ... u_{l(n)}$. Output $D(u_1 u_2 ... u_{l(n)})$.

Now, let's construct another adversary A which receives an input y that can come from either a PRG $G(s)$ or a uniform distribution U_{n+1}. A runs as follows

$$s_1||u_1 \leftarrow \{0,1\}^{n+1}$$
$$..$$
$$..$$
$$s_i||Z \leftarrow y$$
$$s_{i+1}||b_{i+1} \leftarrow G(s_i)$$
$$..$$
$$..$$
$$s_{l(n)}||b_{l(n)} \leftarrow G(s_{n-1})$$

After constructing the string, A then runs $D(u_1u_2...Zb_{i+1}...b_{l(n)})$. If the input was sampled from $G(s)$, the output of D is distributed identically to the output of H_i, whereas if the input was sampled from U_{n+1}, the output of D is distributed identically to the output of H_{i+1}. Thus the advantage of A breaking G is the same as that of D in distinguishing H_i and H_{i+1}. Since the advantage of A breaking G is less than some negligible function $\mathsf{negl(n)}$ and there are l hybrids, the advantage is at most $l \times \mathsf{negl(n)}$. This contradicts our initial assumption that the advantage is at least ϵ. Thus, G_P is a poly-bit stretch PRG. ■

2 Random Functions

Definition 1 *A function F_R is a* ***random function*** *if $F_R : \{0,1\}^n \rightarrow \{0,1\}^m$. For simplicity, we will assume $m = n$.*

It is useful to think of F_R as a table that maps $F_R(x_i) \rightarrow r_i$ where r_i is some random number.

x	$F_R(x)$
000...000	101...110
000...001	111...010
000...010	111...110
.	.
.	.
111.111	001...011

Note that this table is exponentially large with 2^n entries. Since each image of F_R takes up n bits, a bitwise representation of F_R would take up $n \times 2^n$ bits and there is a total of $2^{|F_R|} = 2^{n \times 2^n}$ such functions F_R that map n bits to n bits. It is not possible to store the full-table description of F_R because it would take exponential time. We need to come up with another way to define these random functions.

3 Pseudorandom Functions

An alternative approach is to construct a pseudorandom function (PRF). A PRF looks like a random function, but only requires a polynomial numbers of bits to describe. In other words, a PRF and a random function should be computationally indistinguishable. Let's attempt to provide a definition for a PRF.

Proposition 1 *For any non-uniform PPT distinguisher D, we require that there exists a negligible function $\mathsf{negl(n)}$ such that $\forall n \in N$*

$$\{\Pr[\text{D(table generated randomly)}] - \Pr[\text{D(table generated from PRF)}] = 0\} \leq \mathsf{negl(n)}$$

Problem: D is a PPT machine and cannot process a table that is exponentially large. Let's consider a different **game-based definition** of PRF.

Definition 2 *A function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a **pseudorandom function** if it is*

- *Easy to Compute: For any input $s, x, F(s, x)$ can be computed in polynomial time.*
- *Hard to Distinguish: For every non-uniform PPT D, there exists a negligible function negl(n) such that $\forall n \in N$*

$$Pr[D \text{ wins Guessing Game}] \leq \tfrac{1}{2} + \mathsf{negl(n)}$$

where the Guessing Game is defined as follows

1. *The game has two players: a **challenger** Ch and a **distinguisher** D.*
2. *The game begins with Ch choosing a random seed s and random bit b. If $b = 0$, then Ch will implement a PRF, Otherwise, it will implement a random function. Ch may not switch the function that it is using once the game has started.*
3. *D sends queries $x_1, x_2, ...$ to Ch and can repeat any number of times.*
4. *Ch applies whichever function it chose and responds with the result $F(x_i)$, i.e.*
   - *if $b = 0$, reply $PRF(s, x_i)$.*
   - *if $b = 1$,*
     * *keep a table T for previous answers.*
     * *if x_i is in T, return $T[x_i]$.*
     * *else, choose random $r_i \leftarrow \{0,1\}^n$, $T[x] = r_i$, return r_i and store it in T.*
5. *Game ends when D stops and outputs bit b'.*

D wins the game if $b' = b$.

4 Construction of a PRF

We will construct a PRF using a PRG. We will begin by building a PRF for just 1-bit inputs and generalize it to n-bit inputs later. The construction for the 1-bit input is as follows

Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG. Compute $G(s) = y_0||y_1$ where $|y_0| = |y_1| = n$. For any input $b \leftarrow \{0,1\}$,

$$F(b) = \begin{cases} y_0 & \text{if } b = 0 \\ y_1 & else \end{cases}$$

By constructing F in this way, we guarantee that for any 1-bit input b, $F(b)$ will always be the same and it will always look random since G always produces random looking output.

We can extend the same idea for n-bit inputs by using a decision tree. Whereas in the 1-bit case, the decision tree has depth 1; in the n-bit case, the tree would have depth n and each input x would take a unique path down the tree, eventually arriving at a leaf node. Note that the adversary can only see the leaf nodes and nothing else inside the tree.

More formally, for an n-bit input $x = x_1x_2...x_n$ and a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, the PRF F can be constructed as

$$F(s,x) = G_{x_n}(G_{x_{n-1}}(...(G_{x_1}(s))..))$$

Now let's try to prove the security of this PRF construction.

Theorem 2 ***(Goldrich-Goldwasser-Micali(GGM))*** *If pseudorandom generators exist, then pseudorandom functions exist.*

Proof. We will proceed with another hybrid proof. Notice that if we try to create hybrids on each possible leaf node in the tree, then we will have an exponential number of hybrids and the Hybrid Lemma will not hold. So we must find a way to construct a polynomial number of hybrids. Observe that any PPT adversary is only allowed to make polynomial queries. Since each query corresponds to a unique path, the total number of nodes being visited by all queries is $n \times poly(n) = poly(n)$. So we will construct a hybrid over an arbitrary path taken by some query x.

Suppose that F is not pseudorandom, then by definition, there exists some distinguisher D and noticeable function ϵ such that

$$Pr[D \text{ wins } \mathsf{Guessing\ Game}\ ] \geq \frac{1}{2} + \epsilon$$

In other words, D can distinguish between the output of a PRF and a random function. Now let's construct the hybrids H_i

Level 1 Hybrids:
- H_0: Level 0 is random, level $i > 0$ is pseudorandom (this represents the actual PRF).
- H_1: Level 0,1 are random, level $i > 1$ is pseudorandom.
- ..
- ..
- H_i: Level $\leq i$ are random, level $> i$ is pseudorandom.
- ..
- ..
- H_n: all levels are random (this represents a random function).

Since D can distinguish between a PRF and a random function, it can differentiate between H_0 and H_n with a noticeable advantage ϵ. Then, by the Hybrid Lemma, there exists an $i \leq n \ \in \mathbb{N}$ such that another distinguisher D_i' can distinguish between H_i and H_{i+1} with an advantage of at least $\frac{\epsilon}{n}$. Note that the only difference between H_i and H_{i+1} is that
- in H_i, level $i+1$ is pseudorandom.
- in H_{i+1}, level $i+1$ is random.

So we need to create another set of hybrids for H_i and H_{i+1}.

Level 2 Hybrids: (assuming that all nodes are in lexicographic order)
Suppose D makes q queries. Define $H_{i,q}$ as the same as H_i, except that all nodes at level $i+1$ that are children of nodes $\leq j$ are changed to random. Specifically,
- $H_{i,0}$: Same as H_i.
- $H_{i,1}$: Suppose the query asks for x_a. At level $i+1$, change the parent of x_a and all its siblings to random.
- $H_{i,2}$: Suppose the query asks for x_a and x_b. At level $i+1$, change the parents of x_a and x_b and all their siblings to random.
- ..
- $H_{i,q}$: Same as H_{i+1}.

By the Hybrid Lemma, given D'_i, there exists $D''_{i,j}$ that can distinguish between $H_{i,j}$ and $H_{i,j+1}$ with an advantage of at least $\frac{\epsilon}{qn} \geq \frac{1}{poly(n)}$. The only difference between $H_{i,j}$ and $H_{i,j+1}$ is that
- in $H_{i,j}$, node j at level $i+1$ is pseudorandom.
- in $H_{i,j+1}$, node j at level $i+1$ is random.

Now since $D''_{i,j}$ can distinguish between $H_{i,j}$ and $H_{i,j+1}$, we can construct another distinguisher A as follows:

1. A receives an input y as a random string or as $G(s)$ for some PRG G and a random string s. A interprets as $y_0||y_1$.

2. Substitute y_0 and y_1 for the children of node j and compute the remaining nodes that are affected by the adversary's queries by using random values.

3. If $D''_{i,j}$ says that the distribution of these nodes matches the distribution of $H_{i,j}$, then A decides that the input is pseudorandom. Otherwise, if it matches the distribution of $H_{i,j+1}$, then A decides that the input is random.

According to the above construction, A can distinguish between a random string and the output of a PRG. Since A violates the pseudorandom property of PRG, A does not exist and consequently, $D''_{i,j}, D'_i$, and D do not exists. Thus, the construction of PRF is indistinguishable. ■