CS 600.442 – Modern Cryptography September 23, 2015

Lecture 6: Pseudo-random Functions (PRFs)

Instructor: Abhishek Jain *Scribe: Rahanik Vora*

1 Recap from the previous lecture

PRGs do not behave like random functions, they output only polynomial random bits. We want to generate a lot many random bits than only polynomial number. Pseudo-random function is an efficiently computable function that can generate more than polynomial random bits.

Definition 1 (Pseudo-random functions) *A family of functions* $\{f_s : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}\}$ *is a pseudorandom function (PRF) if:*

- **Efficient computation:** *There exists a PPT* F *s.t.* $F(s,x)$ *efficiently computes the function* $f_s(x)$.
- **Indistinguishability:**

$$\{s \xleftarrow{\$} \{0,1\}^n : f_s\} \approx \{f \xleftarrow{\$} F_n : f\}$$

The output of a PRF should be computationally indistinguishable from that of the output of a random function. To achieve this, it is critical that the seed s to the PRF is not revealed.

Typically, $l(n)$ will be equal to n.

2 Construction of a PRF

We will construct a PRF $\{f_s : \{0,1\}^n \to \{0,1\}^n\}$ from a length-doubling pseudo-random generator (PRG) $G : \{0,1\}^n \to \{0,1\}^{2n}$

Construction of f_s:

- Define G_0 and G_1 to be the two parts of G, so that $G(x) = G_0(x) \| G_1(x)$, where $G_0, G_1 : \{0,1\}^n \to \{0,1\}^n$
- $f_s(x) := G_{x_n}(G_{x_n-1}(\dots G_{x_1}(s)\dots))$

Note that the evaluation of $f_s(x)$ corresponds to traversing down a binary tree. At each level i, we generate $2n$ bits (using PRG) and keep only one n bits (either the left part or the right part, depending upon whether the ith bit of the input is 0 or 1) of the output. The height of the tree corresponds to input length, i.e., n. The tree will contain 2^n leaves, which means that exponentially many pseudorandom outputs, each of length n bits, can be evaluated efficiently by a pseudo-random function. Furthermore, each leaf is reached from the root node by a unique path down the tree.

Proof. Suppose that the proposed construction is not secure, i.e., there exists a n.u. PPT distinguisher D who can distinguish with non-negligible probability whether it is given oracle access to f_s (for a randomly chosen s) or a truly random function $f \leftarrow F_n$.

We are going to prove the security of PRF using the security of PRG via a hybrid argument. Note that one approach would be to do a sequence of hybrids over all the leaves of the tree, where we successively replace each leaf with random distribution. The problem with this approach is that the number of leaves in the tree is exponential, and therefore the hybrid lemma does not apply.

To address this, we will adopt a different hybrid strategy where we replace successive levels of the tree with random distribution. We define hybrid distributions H_n^i as:

- All the nodes in the tree up to level i are replaced with truly random values.
- From level $i+1$ to n, the nodes are generated pseudorandomly using G_0 and G_1 in the same manner as in the tree-based construction of f_s.

Note that H_n^1 corresponds to $\{f_s : \{0,1\}^n \to \{0,1\}^n\}$ (i.e., where only the seed to the PRF is picked at random). Furthermore, H_n^n corresponds to $\{f \stackrel{\$}{\leftarrow} F_n : f\}$ (i.e., where all the nodes, including the leaves are randomly chosen).

Now, if D can distinguish between oracle access to $f \leftarrow F_n$ and f_s (for a randomly chosen s) with non-negligible probability ϵ, then we have that D also can distinguish between oracle access to $f_1 \leftarrow H_n^1$ and $f_n \leftarrow H_n^n$ with probability ϵ. By the Hybrid Lemma, it follows that there exists i s.t. D distinguisher between $f_i \leftarrow H_n^i$ and $f_{i+1} \leftarrow H_n^i$ with non-negligible probability $\frac{\epsilon}{n}$.

Observe that the main difference between H_n^i and H_n^{i+1} is that the nodes at level $i+1$ in H_n^i are generated pseudorandomly using G_0 and G_1, while they are generated randomly in H_n^{i+1}. From level $i+2$ onwards, the nodes are generated pseudorandomly in both the hybrid distributions.

We now need to get a contradiction using the distinguisher for $f_i \leftarrow H_n^i$ and $f_{i+1} \leftarrow H_n^i$. One approach would be to consider yet another sequence of intermediate hybrids between H_n^i and H_n^i where we replace the nodes at level $i+1$, one by one, with random values. However, the problem is that the number of nodes at level $i+1$ are 2^{i+1}, which depending upon the value of i, could be exponential. In this case, we wouldn't be able to use the Hybrid Lemma again.

To address this problem, we use the fact that the distinguisher D makes only polynomial number of queries to its oracle. Let $q(n)$ be this polynomial. Now, let t denote the number of nodes at level $i+1$ that are visited while answering the queries of D. We have that $t \leq q(n)$.

We now define hybrids $H_n^{i,0}, H_n^{i,1}, \ldots, H_n^{i,t}$ where in hybrid $H_n^{i,j}$, the jth node at level $i+1$ that is affected by D's queries is replaced with random. By definition, we have that $H_n^{i,0}$ is the same as H_n^i. Also, since the remaining $2^{i+1} - t$ nodes at level $i+1$ are unaffected by D's queries, we have that $H_n^{i,t} \approx H_n^{i+1}$. If D distinguishes between H_n^i and H_n^{i+1} with probability $\frac{\epsilon}{n}$, then we have that D distinguishes between $H_n^{i,0}$ and $H_n^{i,t}$ with probability $\frac{\epsilon}{n}$. From the Hybrid Lemma, it follows that there exists $j < t$ s.t. D distinguishes between $H_n^{i,j}$ and $H_n^{i,j+1}$ with probability at least $\frac{\epsilon}{n \cdot q(n)}$ (since $t \leq q(n)$). However, the only difference between these two hybrid distributions is that in $H_n^{i,j}$, the $(j+1)$th node at level $i+1$ affected by D's queries is computed pseudorandomly, while it is computed uniformly at random in $H_n^{i,j+1}$. It follows by the robustness property of computational indistinguishability that D contradicts the pseudorandom property of G.

3 Secret Key Encryption

Suppose that Alice wants to send a message m to Bob privately such that no eavesdropper who is listening on their communication channel can read m. How can we accomplish this?

Going forward, we will consider two different settings to address this problem: one where Alice and Bob have a priori met before and share a secret key, and another where Alice and Bob haven't communicated before. We start with the former setting, referred to as secret-key encryption.

Informally speaking, a secret-key encryption scheme should accomplish the following goals:

- Correctness: Alice can compute an "encoding" c of m using the secret s shared by Alice and Bob. Bob can decode m from c correctly using s.
- Security: A polynomial-time adversary should not be able to distinguish between encryptions of messages m and m'.

We start with the syntax of a secret-key encryption scheme. A secret-key encryption scheme consists of three PPT algorithms (Gen, Enc, Dec) defined as follows:

- **Key Generation:** $Gen(1^n) \to s$
- **Encryption:** $Enc(s, m) \to c$
- **Decryption:** $Dec(s, c) \to m$

We require a secret-key encryption scheme to satisfy a correctness property as well as a security property. Here we define the correctness property. The security property will be visited in the next lecture.

- **Correctness:** For every message m, $Dec(s, Enc(s, m)) = m$, where $s \xleftarrow{\$} Gen(1^n)$

Note that here, we are requiring perfect correctness. We can relax this to only require decryption to output the correct message with high probability (where the probability is taken over the randomness used by the key generation and encryption algorithms).