CS 7880 Graduate Cryptography September 20, 2017

Lecture 5: Pseudorandom Generators

Lecturer: Daniel Wichs *Scribe: Willy Quach*

1 Topic Covered

- Computational Security
- Pseudorandom Generators

2 Computational Security

We previously defined the notion of (one-time) *computational security* for encryption schemes, which formalizes the intuition that no *efficient* attacker can distinguish the encryptions of two different messages, even if those messages are chosen maliciously. This is a weaker notion compared to perfect security and ε-security for negligible ε. In particular, the One Time Pad is (one-time) computationally secure (as we saw it is secure against computationally unbounded attackers).

Thus, it is natural to ask whether computational security gives us more flexibility than statistical notions of security to build encryption schemes. More precisely, recall that for ε-security for small ε, we necessarily have $|\mathcal{K}| \geq |\mathcal{M}|$. Does there exist a computationally secure encryption scheme such that $|\mathcal{K}| < |\mathcal{M}|$? It turns out that there is no hope if $P = NP$:

Theorem 1 *Suppose $P = NP$. Then for any n, there is no One-time computationally secure encryption scheme with $\mathcal{K} = \{0,1\}^n$ and $\mathcal{M} = \{0,1\}^{n+1}$.*

Proof: Let $(\mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme (with key space $\mathcal{K}$ and message space $\mathcal{M}$), and let $\mathcal{A}$ be an attacker for the One-time computational security game, defined as follows:

- In the first phase, $\mathcal{A}$ chooses two random messages m_0, m_1 uniformly and independently in $\mathcal{M} = \{0,1\}^{n+1}$.
- After receving a ciphertext c from the Challenger (which encrypts m_b in $OneSec^b$ for $b \in \{0,1\}$), $\mathcal{A}$ checks if:
$$\exists k \in \mathcal{K}, \mathsf{Dec}(k, c) = m_1, \tag{1}$$
and outputs 1 if that is the case, and 0 otherwise.

We first claim that $\mathcal{A}$ is efficient. To do so, define $\mathcal{L} := \{(c, m) \mid \exists k \in \mathcal{K}, \mathsf{Dec}(k, c) = m\}$. Then $\mathcal{L} \in NP$ (as there is some k being a valid witness for any $(c, m) \in \mathcal{L}$ by construction). But by assumption, $\mathcal{A}$ can test efficiently if $(c, m_1) \in \mathcal{L}$, and can therefore perform the check (1) efficiently.

Next, we prove that $\mathcal{A}$ distinguishes games $OneSec^0$ and $OneSec^1$ with constant probability. Note that if $\mathcal{A}$ plays game $OneSec^1$, he always outputs 1 by correctness of the scheme (the key chosen by the Challenger being a valid k for (1)):

$$\Pr[OneSec^1_{\mathcal{A}} = 1] = 1.$$

Now suppose that $\mathcal{A}$ plays game $OneSec^0$. Then, for any c, the set $S := \{\mathsf{Dec}(k, c), k \in \mathcal{K}\}$ has size at most $|\mathcal{K}| = 2^n$. Note that $\mathcal{A}$ outputs 1 if and only if $m_1 \in S$ (where c is computed by the Challenger). As $\mathcal{A}$ picked m_1 independently of m_0, m_1 does not depend on c, so that:

$$\Pr[OneSec^0_{\mathcal{A}} = 1] = \Pr[m_1 \in S] \leq |\mathcal{K}|/|\mathcal{M}| \leq 1/2,$$

which concludes the proof. □

3 Pseudorandom Generators (PRG)

A *Pseudorandom Generator* is a (family of) function that stretch a random input string (the *seed*) and outputs a longer string which looks uniform. More formally:

Definition 1 [Pseudorandom Generator]

A family of deterministic and efficient to compute functions $G : \left(\{0,1\}^n \to \{0,1\}^{\ell(n)}\right)_{n\in\mathbb{N}}$ such that $\forall n, \ell(n) > n$ is a *Pseudorandom Generator* if:

$$G(U_n) \approx U_{\ell(n)}.$$

♢

Remark 1 *The set $G(\{0,1\}^n)$ has size at most 2^n, whereas $\{0,1\}^{\ell(n)}$ has size greater than 2^{n+1}. So an unbounded adversary can easily distinguish between the two distributions. However if G is a PRG, then no efficient adversary can do the same.*

The existence of PRGs suffices to build non-trivial One-Time computationally secure encryption. Define:

$$\mathsf{Enc}(k, m) := G(k) \oplus m,$$

$$\mathsf{Dec}(k, c) := G(k) \oplus c,$$

where $\mathcal{K} = \{0,1\}^n$ and $\mathcal{M} = \{0,1\}^{\ell(n)}$ (recall that $\ell(n) > n$ by definition, so that $|\mathcal{K}| < |\mathcal{M}|$).

In other words, this scheme uses the output of G as a One-Time Pad key.

Theorem 2 *Suppose $G : \left(\{0,1\}^n \to \{0,1\}^{\ell(n)}\right)_{n\in\mathbb{N}}$ is a PRG. Then $(\mathsf{Enc}, \mathsf{Dec})$ defined above is One-Time computationally secure.*

Proof: The proof proceeds with a sequence of hybrids, using the hybrid argument introduced in the previous lecture.

Let $OneSec^b, b \in \{0,1\}$ be the game defining One-time computationally security.

Define the following intermediate games (which only differ in the way c is computed by the Challenger):

- Game $OneSecR^b, b \in \{0,1\}$: this is almost the same game as $OneSec^b$, but the Challenger here computes c as $c = R \oplus m_b$ for a uniformly random $R \leftarrow \{0,1\}^{\ell}(n)$.
- Game $OneSecR^*$: here the Challenger picks c uniformly in $\{0,1\}^{\ell}(n)$.

Note that $OneSec^b \approx OneSecR^b$ for any $b \in \{0,1\}$. This follows from a reduction from the security of the PRG: we show that a distinguisher D between $OneSec^b$ and $OneSecR^b$ (for any b) implies a distinguisher R between $G(U_n)$ and $U_{\ell(n)}$. Indeed, the reduction R receives samples z from $\{0,1\}^{\ell(n)}$, and forwards $z \oplus m_b$ to the $OneSec^b$ - $OneSecR^b$ distinguisher D. Then it forwards the output of the distinguisher to the Challenger.

Then R is clearly efficient. Also if the sample comes from $G(U_n)$ then the distinguisher D receives samples from $OneSec^b$; if the samples come from $U_{\ell(n)}$ then the distinguisher D receives samples from $OneSecR^b$. Therefore, the reduction R distinguishes $G(U_n)$ and $U_{\ell(n)}$ with the same probability as the distinguisher D does for $OneSec^b$ and $OneSecR^b$.

Also, we have that: $OneSec^b \approx OneSecR^*$ for any b (actually we already showed that $OneSec^b \equiv OneSecR^*$ when studying the One-Time Pad, which is a stronger statement).

Overall, we have $OneSec^0 \approx OneSecR^0 \approx OneSecR^* \approx OneSecR^1 \approx OneSec^1$, and so, a hybrid argument gives:

$$OneSec^0 \approx OneSec^1,$$

which concludes the proof. □

Next, we show how one can increase the stretch of a PRG:

Theorem 3 *Let $G : \left(\{0,1\}^n \to \{0,1\}^{n+1}\right)_{n\in\mathbb{N}}$ be a PRG (where $\ell(n) = n+1$). Then there exists a PRG $G' : \left(\{0,1\}^n \to \{0,1\}^{\ell(n)}\right)_{n\in\mathbb{N}}$ for any polynomial $\ell(n) > n$.*

Remark 2 *We cannot hope to achieve a PRG with arbitrarily large, say, exponential stretch, because the distinguisher is allowed to run in time polynomial to its input. If $\ell(n) = 2^n$ for instance, the distinguisher can run G on all inputs $\{0,1\}^n$; with high probability, a sample in $U_{\ell(n)}$ will not lie in $G(\{0,1\}^n)$.*

Proof: Let G be a PRG with 1-bit stretch (*i.e.* G outputs $n+1$ bits). The idea is to iterate G to get an extra bit of pseudorandomness at every step:

$$\begin{array}{ccccccccc} & & & & b_1 & & & & b_k \\ x_0 & \xrightarrow{G} & & & x_1 & \xrightarrow{G} & \cdots & \xrightarrow{G} & x_k \end{array}$$

where $x_i \in \{0,1\}^n, b_i \in \{0,1\}$ are defined by $x_0 \leftarrow \{0,1\}^n$, and $G(x_i) = (x_{i+1} \| b_{i+1})$ for $i < k$, and set:

$$G'(x_0) = (x_k \| b_1 \| \cdots \| b_k) \in \{0,1\}^{n+k}.$$

This gives a new family of functions with $\ell(n) = n+k$. Let us prove that, for constant k, G' is a PRG. However, the definition of a PRG only gives guarantees with respect to *uniform* inputs. Let us define the following distributions:

$$H_i = (x_k \| b_1 \| \cdots \| b_k),$$

where:

$$
\begin{array}{lccccccc}
 & & & & b_{i+1} & & & b_k \\
b_1, \cdots, b_i \leftarrow \{0,1\}, & & & & & & & \\
 & x_i \leftarrow \{0,1\}^n & \xrightarrow{G} & x_{i+1} & \xrightarrow{G} & \cdots & \xrightarrow{G} & x_k
\end{array}
$$

Note that we have for all $i < k$: $H_i \approx H_{i+1}$. This follows from a similar reduction as in the previous theorem: a distinguisher D between H_i and H_{i+1} for any i implies a distinguisher R for the PRG G with same success probability.

More precisely, R receives some sample $y = (x_{i+1} \| b_{i+1}) \in \{0,1\}^{n+1}$, and computes $k - i - 1$ iterations of G with input x_{i+1}, as above. It picks uniformly $b_1, \cdots b_i$ uniformly itself, and sends $(x_k \| b_1 \| \cdots \| b_k)$ to D, and forwards the output of D. Such a reduction R is efficient (as G itself is efficient to compute).

Then, if y comes from $G(U_n)$ then D receives samples distributed as H_i; if y comes from U_{n+1} then D receives samples distributed as $H_i + 1$.

As H_0 corresponds to the distribution $G(U_n)$, and H_k corresponds to $U_{\ell(n)}$, a hybrid argument concludes the proof for constant k.

One has to be slightly more careful if $k(n)$ is polynomial in n for instance. Recall that all random variables X are indexed with a security parameter n. We now have to consider a family of increasingly many random variables $\{X_n^k\}_{n \in \mathbb{N}, k \leq \ell(n)}$.

Then we can define our intermediate distributions as before: $\{H_n^{i(n)}\}_{n \in \mathbb{N}}$ for any polynomial $i(n)$ (which is efficiently computable OR if we allow non-uniform reductions).

We then prove similarly as above that:

$$\forall \text{ polynomial } i(n), \{H_n^{i(n)}\}_n \approx \{H_n^{i(n)+1}\}_n.$$

The hybrid argument generalizes naturally in this case: if for all polynomials $i \leq k$ (where $n + k(n) = \ell(n)$), we have $\{H_n^{i(n)}\}_n \approx \{H_n^{i(n)+1}\}_n$, then:

$$\{H_n^0\}_n \approx \{H_n^{\ell(n)}\}_n,$$

which concludes the proof for polynomial $k(n)$.

□

Remark 3 *One advantage of the construction of the PRG in the proof above is that you do not have to know in advance how many outputs bits you need; you can always continue iterating the process whenever needed.*