1 Last lecture, and plan for today's lecture

Last lecture, we saw how to improve the 'PRG stretch' (that is, ratio of output size/input size) by composing the base PRG appropriately. Today, we will prove security of the composed PRG, see a 'tree-based' construction (the proof of which is left as an exercise). We will then discussed how PRGs are constructed from more *basic* building blocks.

1.1 Questions from last class

The following are a few questions from last class:

Qn: What is the average-case subset-sum problem?
Ans: The average case subset-sum problem is same as the worst-case subset-sum problem, except that the instance (which is defined by $a_1, a_2, \ldots, a_n$ and the subset are chosen from some appropriate probability distribution.

Qn: We showed in last class that if G is a secure PRG, then $\mathcal{E}_G$ satisfies No-Query-Semantic-Security security. Is the converse also true (that is, if $\mathcal{E}_G$ satisfies No-Query-Semantic-Security, then G is a secure PRG?)
Ans: Yes, the converse is also true (and it is a good exercise to get comfortable with reductions). However, the converse is not very interesting for the following sense: we are using G as a building block for $\mathcal{E}_G$. Therefore, it makes sense to ask that if G satisfies certain properties (PRG security in this case), then can we say something about the resulting encryption scheme (No-Query-Semantic-Security in this case). However, the converse is not very interesting, because we are using G as a building block, but want to prove something about G, assuming something about $\mathcal{E}_G$.

2 Expanding output space of PRG

Recall the chaining based construction from last class: $G : \{0,1\}^n \to \{0,1\}^{n+1}$ is a secure PRG, and we want to build a PRG with greater 'stretch' — that is, a function that maps n bits to $n+2$ bits. Consider the following function G' that maps n bits to $n+2$ bits:

$$G' : \{0,1\}^n \to \{0,1\}^{n+2}$$
$$G'(x) = z_1 z_2 \ldots z_{n+1} \mid\mid y_{n+1} \text{ where}$$
$$G(x) = y_1 y_2 \ldots y_{n+1}, G(y_1 y_2 \ldots y_n) = z_1 z_2 \ldots z_{n+1}$$

We wish to prove that G' is secure, assuming G is. As before, we will assume the contrapositive. Suppose there exists a p.p.t. adversary $\mathcal{A}$ such that $p_0 = \Pr\left[\mathcal{A} \text{ outputs 0, given } G'(s) \text{ for random } s\right]$ and $p_1 = \Pr\left[\mathcal{A} \text{ outputs 0, given uniformly random } u\right]$ are non-negligibly far apart. We will use $\mathcal{A}$ to break PRG security of G, via a reduction algorithm $\mathcal{B}$.

An attempt that 'almost' works: The following was one of the suggestions in class: consider a reduction algorithm $\mathcal{B}$ that interacts with the PRG challenger for G. It receives a string $y = (y_1, y_2, \ldots, y_{n+1}) \in \{0,1\}^{n+1}$ which is either the PRG output on a uniformly random input, or a uniformly random $n+1$ bit string. The reduction algorithm first computes $(z_1, z_2, \ldots, z_{n+1}) = G(y_1, y_2, \ldots, y_n)$ and sends $(z_1, z_2, \ldots, z_{n+1}, y_{n+1})$ to $\mathcal{A}$. The adversary sends its guess b', which the reduction algorithm forwards to the challenger.

Let us analyse this reduction algorithm's probability of winning against G.

$$
\begin{aligned}
&\Pr\left[\mathcal{B} \text{ wins PRG game w.r.t. } G\right] \\
=&\Pr\left[\mathcal{A} \text{ sends } 0 \mid \text{Chall. sends } G(s) \text{ for random s}\right] \cdot \frac{1}{2} \\
&+ \left(1 - \Pr\left[\mathcal{A} \text{ sends } 0 \mid \text{Chall. sends uniformly random } y\right]\right) \cdot \frac{1}{2}
\end{aligned}
$$

Now, note that if the G-challenger sends $G(s)$, then this corresponds to world-0 for $\mathcal{A}$, and therefore $\Pr\left[\mathcal{A} \text{ sends } 0 \mid \text{Chall. sends } G(s) \text{ for random s}\right] = p_0$.

But can we conclude that $\Pr\left[\mathcal{A} \text{ sends } 0 \mid \text{Chall. sends uniformly random } y\right] = p_1$? No, since p_1 is the probability of $\mathcal{A}$ outputting 0 when it receives a uniformly random $n+2$ bit string, and $(z_1, z_2, \ldots, z_{n+1}, y_{n+1})$ (as defined in the reduction above) is not a uniformly random $n+2$ bit string.

As was pointed out in class, using the PRG security, we expect (intuitively) that the above probability (that is, $\Pr\left[\mathcal{A} \text{ sends } 0 \mid \text{Chall. sends uniformly random } y\right]$) should be close to p_1. This intuition is formally captured via the hybrid technique.

A formal proof via hybrid technique: In the hybrid world, the challenger chooses a uniformly random $n+1$ bit string $(y_1, \ldots, y_{n+1})$, computes $(z_1, \ldots, z_{n+1}) = G(y_1, \ldots, y_n)$ and outputs $(z_1, \ldots, z_{n+1}, y_{y+1})$ to $\mathcal{A}$. Let p_{hyb} denote the probability that $\mathcal{A}$ outputs 0 in the hybrid world.

Since $\mathcal{A}$ breaks the security of G', we know that $p_0 - p_1$ is non-negligible.

Claim 08.01. There exists a p.p.t. reduction algorithm that uses $\mathcal{A}$ and wins the PRG security game against G with probability $1/2 + (p_0 - p_{\text{hyb}})/2$.

Proof. The reduction algorithm receives $(y_1 \ldots y_{n+1})$. It computes $(z_1, \ldots, z_{n+1}) = G(y_1, \ldots, y_n)$ and sends $(z_1, \ldots, z_{n+1}, y_{n+1})$ to $\mathcal{A}$. The algorithm $\mathcal{A}$ sends its guess b', which the reduction algorithm forwards to the G-challenger.

Note: if the challenger sent a pseudorandom string $y = (y_1, \ldots, y_{n+1}) = G(s)$ (for a random s), then the adversary $\mathcal{A}$'s view is identical to its view in world-0. If $(y_1, \ldots, y_{n+1})$ is a uniformly random string, then the adversary $\mathcal{A}$'s view is identical to its view in the hybrid world. Therefore, the reduction algorithm's success probability is $1/2 + (p_0 - p_{\text{hyb}})/2$ (complete the calculations).

□

Claim 08.02. There exists a p.p.t. reduction algorithm that uses $\mathcal{A}$ and wins the PRG security game against G with probability $1/2 + (p_{\text{hyb}} - p_1)/2$.

Proof. The reduction algorithm receives $(z_1 \ldots z_{n+1})$ from the PRG challenger. It picks a uniformly random bit y_{n+1} and sends $(z_1, \ldots, z_{n+1}, y_{n+1})$ to $\mathcal{A}$. The adversary sends a guess, which the reduction forwards to the challenger.

Note: if the G-challenger sent a pseudorandom string, then the adversary's view is identical to its view in the hybrid world. If $(z_1, \ldots, z_{n+1})$ are uniformly random bits, then the adversary's view is identical to its view in world-1. Therefore, the reduction algorithm's success probability in the PRG security game w.r.t. G is $1/2 + (p_{\text{hyb}} - p_1)/2$. □

Note that the reduction algorithms in the above claims are different. A unified reduction algorithm, therefore guesses which of these two differences (that is, $(p_0 - p_{\text{hyb}})$ and $(p_{\text{hyb}} - p_1)$) is non-negligible, and chooses an appropriate strategy according to its guess. For the rest of this course, it suffices to just describe the hybrid experiments, and then show that the consecutive hybrid worlds are indistinguishable.

3 PRGs and one-wayness

Suppose $G : \{0,1\}^n \to \{0,1\}^\ell$ is a secure PRG, where $\ell > n$. Is it possible for an efficient adversary to recover the input of the PRG, given only the output? Suppose an adversary $\mathcal{A}$ could 'invert' the function G, then this is **not a secure PRG**. In particular, we can use this adversary $\mathcal{A}$ to break the PRG security of G. The reduction algorithm receives a string u from the PRG challenger, it sends u to $\mathcal{A}$. The adversary sends an input s. The reduction algorithm checks if $G(s) = u$. If so, it guesses that u is pseudorandom, else it guesses that u is truly random.

A secure PRG is an example of a *one-way function* — a function that can be efficiently computed, but it is hard to invert, given the output on a random input.

Definition 08.01. A function $f : \{0,1\}^n \to \{0,1\}^m$ is a secure one way function if, for any p.p.t. adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all n,

$$\Pr_{x \leftarrow \{0,1\}^n} \left[\mathcal{A}(f(x)) = x' \wedge f(x') = f(x)\right] \leq \mu(n),$$

where the probability is over the choice of $x \leftarrow \{0,1\}^n$, as well as the adversary's randomness.

Note that the adversary does not need to output the same x that was chosen to compute $f(x)$; it suffices for the adversary to output any preimage of $f(x)$.

One way functions are the most fundamental object in cryptography. Most cryptographic primitives will imply the existence of one way functions. For instance, the existence of no-query semantically secure encryption schemes (with message space $\mathcal{M} = \{0,1\}^\ell$ much larger than the key space $\mathcal{K} = \{0,1\}^n$) implies the existence of OWFs. The function $\mathsf{Enc}(0^\ell, \cdot) : \mathcal{K} \to \mathcal{C}$ is a one way function (proof left as an exercise).

The other direction is more interesting.

What cryptographic primitives can we build from one way functions?

It turns out that one way functions imply the existence of PRGs. This should look very surprising! A one way function can be any arbitrary function, the only guarantee is that computing in one direction is easy, and computing in the other direction is hard. The function might not even be length-expanding. Still, even without assuming any structure in the one way function, one can show that if one way functions exist, then so do pseudorandom generators. We will not prove this in class, but will see a 'baby-version' of this theorem in the next assignment.

Note that this also implies that encryption schemes with $\mathsf{No\text{-}Query\text{-}Semantic\text{-}Security}$ can be built from any one way function. And similarly, one-round bit commitment schemes can be built from any one way function.

3.1 What practitioners care about

One-way functions implying PRGs is a significant result for theoreticians, but these constructions are prohibitively expensive to be used in practice. Instead, quite often, PRGs are implemented in practice using the DES or AES circuits. This is the next topic of our course. We will not go into the details of the $\mathsf{AES}, \mathsf{DES}$ circuits, but instead, we will view them as an abstract cryptographic primitive called *block ciphers* (practitioners' term) or *pseudorandom functions/permutations* (theoreticians' preferred term).

4 AES: Pseudorandom Functions/Permutations

The Advanced Encryption Standard (AES) consists of two efficiently implementable, **deterministic**, keyed functions. The first function AES takes as input a message block, a key, and outputs a cipher block. Abstractly, we can think of it as $\mathsf{AES} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. Here, for simplicity, we are taking block size = key size. This function must have the following properties:

- for every key k, the function $\mathsf{AES}(\cdot, k)$ is a permutation.
- for a random key k, the function $\mathsf{AES}(\cdot, k)$ 'looks like' a uniformly random permutation. We will formally define what it means to look like a uniformly random permutation.

The second function is the inverse: $\mathsf{AES}^{-1} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. Again, for every key k, $\mathsf{AES}^{-1}(\cdot, k)$ is a permutation. Moreover, for correctness, we require that for any input x and key k,

$$\mathsf{AES}^{-1}\Big(\ \mathsf{AES}(x,k), k\Big) = x.$$

5 Using AES to encrypt long messages

The AES was designed primarily for encryption of long messages. Here is a bad way to use AES, and one that should NEVER be used in practice. It is called the Electronic Codebook (ECB) mode of encrytion. The encryption key is an AES key $k \in \{0,1\}^n$. Given a message $m \in \{0,1\}^{n \cdot \ell}$, the ECB method breaks the message into chunks, each chunk being n bits long. Let $m = (m_1 \mid\mid m_2 \mid\mid \ldots \mid\mid m_\ell)$ be the message. The encryption algorithm computes $\mathsf{ct}_i = \mathsf{AES}(m_i, k)$ and outputs $\mathsf{ct} = (\mathsf{ct}_1 \mid\mid \mathsf{ct}_2 \mid\mid \ldots \mid\mid \mathsf{ct}_\ell)$ as the final ciphertext.

The ECB mode does not satisfy No-Query-Semantic-Security, since the encryption of $0^{n \cdot \ell}$ can be distinguished from the encryption of a random message. Why?

6 Lecture summary, plan for next lecture, additional resources

Summary This lecture concludes our discussion on pseudorandom generators (also known as *stream ciphers*). Here are the key takeaways from this lecture (and this topic overall).

- PRGs are deterministic, length-expanding functions, whose output on a random input looks random.
- These can be used to construct no-query semantic secure encryption schemes (that is $\mathcal{E}_G$), commitment schemes (described in assignment), and many other cryptographic primitives.
- Given a PRG that maps n bits to $n+1$ bits, we can construct another PRG that maps n bits to $n+\ell$ bits, for any $\ell > 0$.
- We started with the topic of block ciphers/pseudorandom functions, and discussed a broken encryption mode (the ECB mode).

Next Lecture: Next lecture, we will formally define what it means for a keyed function/permutation to look like a random function/permutation. We will also see a few other modes of encryption that are better than ECB mode. Finally, we will discuss how to use the AES circuit to build a PRG.

Relevant sections from textbook [Boneh-Shoup]: Sections 3.4.

7 Questions

Question 08.01. Let $G : \{0,1\}^n \to \{0,1\}^\ell$ be a deterministic function with $\ell > n$. Let $\mathcal{E}_G$ be the encryption scheme derived from G (with message space $\mathcal{M} = \{0,1\}^\ell, \mathcal{K} = \{0,1\}^n$). Show that if $\mathcal{E}_G$ satisfies No-Query-Semantic-Security, then G is a secure pseudorandom generator.

Hint: Reduction chooses two messages m_0, m_1 uniformly at random and sends them to the challenger. The challenger sends a ciphertext ct. The reduction algorithm sends $m_0 \oplus \mathsf{ct}$ to the PRG attacker.

Question 08.02. The construction in Section 2 can be generalized to transform a secure PRG $G : \{0,1\}^n \to \{0,1\}^{n+\ell}$ to another secure PRG $G' : \{0,1\}^n \to \{0,1\}^{n+2\ell}$. Suppose we modify the construction given above. Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a secure PRG. Consider the function G' that maps n bits to $4n$ bits, and is defined below.

$$G'(x) = y_0 \mid\mid y_1 \mid\mid y_{10} \mid\mid y_{11}$$
$$\text{where } G(x) = y_0 \mid\mid y_1 \quad (\text{each } y_b \in \{0,1\}^n)$$
$$G(y_1) = y_{10} \mid\mid y_{11} \quad (\text{each } y_{1b} \in \{0,1\}^n)$$

First, you should check that G' is not a secure PRG. If you try to prove that G' is secure, assuming G is secure, where does the reduction break down?

Hint: Given output $z_1 \mid\mid z_2 \mid\mid z_3 \mid\mid z_4$, check if $G(z_2) = z_3 \mid\mid z_4$. If we try to prove security via hybrids, in the intermediate hybrid, the challenger chooses y_0, y_1 at random, and outputs $y_0 \mid\mid y_1 \mid\mid G(y_1)$. However, we will get stuck when we try to show that this is indistinguishable from World-1. The reduction does not get the PRG input from the challenger.

Question 08.03. Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a secure PRG. Consider the function G' that maps n bits to $4n$ bits, and is defined below.

$$G'(x) = y_{00} \mid\mid y_{01} \mid\mid y_{10} \mid\mid y_{11}$$
$$\text{where } G(x) = y_0 \mid\mid y_1 \quad (\text{each } y_b \in \{0,1\}^n)$$
$$G(y_0) = y_{00} \mid\mid y_{01} \quad G(y_1) = y_{10} \mid\mid y_{11} \quad (\text{each } y_{b_1 b_2} \in \{0,1\}^n)$$

Prove that G' is a secure PRG, assuming G' is.

Hint: We need to construct two intermediate hybrids for this. In the first hybrid, the challenger chooses y_0, y_1 at random, and outputs $G(y_0) \mid\mid G(y_1)$. In the second hybrid, the challenger chooses y_1, y_{00}, y_{01} at random, and outputs $y_{00} \mid\mid y_{01} \mid\mid G(y_1)$. Hybrid-1 is indistinguishable from world-0, hybrid-2 is indistinguishable from hybrid-1, and world-1 is indistinguishable from hybrid-2, all via PRG security.