University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 11

1 Improving the Stretch of a PRG

Last time we saw that a PRG was enough to construct an encryption scheme which is secure yet beats the one-time pad. In particular, if we have a PRG that "stretches" its input by one bit (i.e., on k-bit inputs, G returns a $k+1$-bit output) then we can construct a secure encryption scheme in which the parties share $k-1$ bits but can encrypt messages of length k.

While this is good, it would be even better if we could improve the efficiency and construct encryption schemes where we save more than just a single bit. To do so, it seems that what we need is a PRG that stretches its input by more than just a single bit (the encryption scheme will then use this PRG exactly as it was used in the last lecture). So the question becomes: can we take a PRG that stretches its input by a single bit, and use it to build a PRG that stretches its input even more?

In class, a number of suggestions were proposed for constructing a PRG H that stretches its input by two bits (here, $G : \{0,1\}^* \rightarrow \{0,1\}^*$ is a PRG that stretches its input by one bit):

1. $H(x) = G(G(x))$
2. $H(x) = G(x \circ 1)$ (where $\circ$ denotes concatenation).
3. $H(x_1 \circ x_2) = G(x_1) \circ G(x_2)$ (where $|x_1| = |x_2| = 1/2|x_1 \circ x_2|$)

(Note that in each case H stretches its input by two bits.) In fact, the second proposal above does *not* work: a proof of security does not go through because the input to G is no longer random (the last bit is fixed) and the security of a PRG depends on its seed being random. Even more so, it is possible to give an *explicit* PRG G such that H is *demonstrably* insecure as a PRG (i.e., even though G is a PRG, we can give an explicit algorithm A which "breaks" H). It is a nice challenge (and might pop up as a test question) to show this.

The other two proposals do yield an H which is a PRG. You are asked to analyze the third proposal on the homework (grad students only). We analyze the first proposal now.

Theorem 1 *Let $G : \{0,1\}^* \rightarrow \{0,1\}^*$ be a PRG that stretches its input by one bit and define $H(x) = G(G(x))$. Then H is a PRG that stretches its input by two bits.*

Proof The proof follows a standard form (that you should get used to):

1. Assume (toward a contradiction) that H is not a PRG.

2. This means that there exists a PPT algorithm A that "breaks" H; i.e.,

$$\left|\Pr[x \leftarrow \{0,1\}^k; y = H(x) : A(y) = 1] - \Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1]\right| = \delta(k), \qquad (1)$$

and $\delta(\cdot)$ is not negligible.

3. We use A to construct a PPT algorithm A' that "breaks" G. This will be a contradiction, since G is a PRG. Thus, our original assumption in step 1 must be wrong, and in fact H is a PRG.

We now give the details.

Construct algorithm A' (which gets input z and must decide whether z is pseudorandom [i.e., an output of G] or random) as follows: $A'(z)$ computes $y = G(z)$ and runs $A(y)$. If A outputs 1 (which may be viewed as a guess by A that y is pseudorandom [i.e., an output of H]), then A' guesses that z is pseudorandom. If A outputs 0, then A' guesses that z is random. We let a guess of "pseudorandom" correspond to an output of 1 and a guess of "random" correspond to an output of 0.

Let's analyze the success of A' in "breaking" G. We are interested in the following:

$$\left|\Pr[x \leftarrow \{0,1\}^k; z = G(x) : A'(z) = 1] - \Pr[z \leftarrow \{0,1\}^{k+1} : A'(z) = 1]\right|.$$

Let $P_1 \stackrel{\text{def}}{=} \Pr[x \leftarrow \{0,1\}^k; z = G(x) : A'(z) = 1]$ and let $P_2 \stackrel{\text{def}}{=} \Pr[z \leftarrow \{0,1\}^{k+1} : A'(z) = 1]$. We can re-write P_1, using the definition of algorithm A', as follows:

$$P_1 = \Pr[x \leftarrow \{0,1\}^k; z = G(x); y = G(z) : A(y) = 1]$$

(this is true because A' outputs 1 exactly when A does). Now, the experiment "$x \leftarrow \{0,1\}^k; z = G(x); y = G(z)$" is exactly the experiment "$x \leftarrow \{0,1\}^k; y = H(x)$"; giving:

$$P_1 = \Pr[x \leftarrow \{0,1\}^k; y = H(x) : A(y) = 1].$$

This is one of the terms in (1) so we must be making progress!

Let's look at P_2. Using the definition of algorithm A gives:

$$P_2 = \Pr[z \leftarrow \{0,1\}^{k+1}; y = G(z) : A(y) = 1].$$

Hmm... this is not quite what we need because this expression does *not* correspond directly to one of the terms in (1). Let's see how to get around this. The term to which we need to relate P_2 is $\Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1]$. Consider the difference between these terms:

$$\begin{aligned}
&\left|P_2 - \Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1]\right| \\
&\quad = \left|\Pr[z \leftarrow \{0,1\}^{k+1}; y = G(z) : A(y) = 1] - \Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1]\right|.
\end{aligned}$$

But we know what this is! This is the success probability of A in "breaking" G. And since G is a PRG, this difference is negligible; call it $\epsilon(k)$.

Recall we are interested in the difference $|P_1 - P_2|$. We have:

$$
\begin{aligned}
&|P_1 - P_2| \\
&= \Big|\Pr[x \leftarrow \{0,1\}^k; y = H(x) : A(y) = 1] - \Pr[z \leftarrow \{0,1\}^{k+1}; y = G(z) : A(y) = 1]\Big| \\
&= \Big|\Pr[x \leftarrow \{0,1\}^k; y = H(x) : A(y) = 1] - \Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1] \\
&\quad + \Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1] - \Pr[z \leftarrow \{0,1\}^{k+1}; y = G(z) : A(y) = 1]\Big| \\
&\geq \Big|\Pr[x \leftarrow \{0,1\}^k; y = H(x) : A(y) = 1] - \Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1]\Big| \\
&\quad - \Big|\Pr[y \leftarrow \{0,1\}^{k+2} : A(y) = 1] - \Pr[z \leftarrow \{0,1\}^{k+1}; y = G(z) : A(y) = 1]\Big| \\
&= \delta(k) - \epsilon(k),
\end{aligned}
$$

Thus, A' "breaks" G with probability $\delta(k) - \epsilon(k)$. If $\delta(\cdot)$ is not negligible, then (since $\epsilon(\cdot)$ *is* negligible) neither is $\delta(\cdot) - \epsilon(\cdot)$. In other words, if A "breaks" H with non-negligible probability, then A' "breaks" G with non-negligible probability, a contradiction. ■