University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 10

1 Note on the Squaring One-Way Function

I previously claimed in class that the Rabin "squaring" function $f : \mathbb{Z}_N^* \rightarrow \mathcal{QR}_N$ is a permutation when its domain is restricted to $\mathcal{QR}_N$. This is actually not quite true (as you saw on homework 1 in the group $\mathbb{Z}_{35}^*$). Rather, $f : \mathcal{QR}_N \rightarrow \mathcal{QR}_N$ is a permutation *only* when $N = pq$ and $p \equiv q \equiv 3 \bmod 4$. See the updated notes for Lecture 7 for further details.

2 Pseudorandom Generators

Remember that our goal is to design a secure encryption scheme which beats the one-time pad; namely, that allows us to securely encrypt messages longer than the shared key. One proposal we had (reviewed below) used the notion of a pseudorandom generator (PRG): this is a function $G : \{0,1\}^* \rightarrow \{0,1\}^*$ which (for now) has the property that its output is 1 bit longer than its input. (So if $|x| = k - 1$, then $|G(x)| = k$.) We need the following two properties from G (informally, for now): (1) G should be efficiently computable; (2) the output of G should "look random".

The encryption scheme we suggested worked as follows. Alice and Bob share a key sk of length $k-1$. When Alice wants to communicate message $m \in \{0,1\}^k$ to Bob, she computes $C = m \oplus G(sk)$ and sends C to Bob. To decrypt, Bob computes $m = C \oplus G(sk)$. Note the analogy to the one-time pad scheme: $G(sk)$ results in a "shared key" of length k which is then used as a one-time pad. The hope is that if we define G appropriately, then we can prove security for this construction.

Overall, then, we need to do two things: First, we will have to propose a rigorous definition of what it means for the output of G to "look random" and then verify that G satisfying this definition will indeed result in the above encryption scheme being secure. Second, we need to construct a PRG satisfying this definition! We handle the first of these concerns today.

What does it mean for the output of G to "look random"? One could require, say, that the last bit of $G(x)$ have equal probability of being 0 or 1, or that the fraction of 1s in $G(x)$ should be roughly 1/2. But these are just particular conditions, and don't seem to capture everything about what it means to be random. What we would like to say is that *no* possible (efficient) test can distinguish between $G(x)$ and a random value. The formal way we do this is as follows:

Definition 1 *G (as described above) is a* pseudorandom generator *(PRG) if it is efficiently computable and for all* PPT *distinguishing algorithms D the following is negligible:*

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

(In words: in the first experiment we pick a random "seed" of length $k-1$, apply G to this seed to get y of length k, and give y to D. In the second experiment we pick a completely random y of length k and give this to D. We require that D not be able to distinguish between these two scenarios with more than negligible probability. Note that when D outputs "1" we can view this as D's guess that y is pseudorandom and when D outputs "0" we can view this as D's guess than y is random.)

Now let's recall our definition of a secure encryption scheme. An encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is secure if, for all PPT algorithms A and all messages m_0, m_1, the following is negligible:

$$\left| \Pr[sk \leftarrow \mathcal{K}(1^k); C \leftarrow \mathcal{E}_{sk}(m_0) : A(C) = 0] - \Pr[sk \leftarrow \mathcal{K}(1^k); C \leftarrow \mathcal{E}_{sk}(m_1) : A(C) = 0] \right|.$$

(Note: the notation here is a little different from what I used in class — in class I used m_1, m_2 at first — but otherwise is the same.) Restating this for the encryption scheme described above (in which $\mathcal{K}(1^k)$ simply picks a random sk of length $(k-1)$ and encryption/decryption are as specified above), gives the following: For all PPT algorithms A and all messages m_0, m_1 the following is negligible:

$$\Big| \Pr[sk \leftarrow \{0,1\}^{k-1}; y = G(sk); C = m_0 \oplus y : A(C) = 0] \\ - \Pr[sk \leftarrow \{0,1\}^{k-1}; y = G(sk); C = m_1 \oplus y : A(C) = 0] \Big|. \quad (1)$$

With this in mind, we now prove the following theorem:

Theorem 1 *If G is a PRG then the encryption scheme described above is secure.*

Proof We prove this via the following methodology: Assume (toward a contradiction) that the encryption scheme given above is *not* secure. Then there exists some algorithm PPT A that "breaks" it; i.e., for which (1) is not negligible. We show how to use any such algorithm to construct a PPT distinguisher D which can distinguish the output of G from a random string with non-negligible probability. This will contradict the fact that G is a PRG, hence our original assumption is false and the encryption scheme must be secure.

A proof of this sort is known as a *reduction*: we reduce the security of the encryption scheme to that of G. We saw another proof of this form when we reduced the hardness of inverting the Rabin squaring function to the hardness of factoring.

So, assume we have an algorithm A and messages m_0, m_1 for which:

$$\Big| \Pr[sk \leftarrow \{0,1\}^{k-1}; y = G(sk); C = m_0 \oplus y : A(C) = 0] \\ - \Pr[sk \leftarrow \{0,1\}^{k-1}; y = G(sk); C = m_1 \oplus y : A(C) = 0] \Big| = \gamma(k),$$

where $\gamma(k)$ is not negligible. We construct an algorithm D as follows: D — which gets input y and has to guess whether y is random or pseudorandom — first picks a random bit $b \in \{0,1\}$. D then sets $C = m_b \oplus y$ and runs $A(C)$ to get a bit b'. This b' represents A's guess as to what message was encrypted. If $b = b'$ (i.e., A guessed correctly) then D guesses "pseudorandom" (which we will denote by having D output "1"). If $b \neq b'$ (i.e., A did not guess correctly) then D guesses "random" (which we will denote by having D output "0").

We want to know how well D does at distinguishing outputs of G from random strings. The quantity we are interested in is:

$$\left|\Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : D(y) = 1] - \Pr[y \leftarrow \{0,1\}^k : D(y) = 1]\right|. \tag{2}$$

(See Definition 1.)

Let's look at each of these terms individually. Let $P_1 \stackrel{\text{def}}{=} \Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : D(y) = 1]$. Just by looking at what D does, we can write:

$$P_1 = \Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x); b \leftarrow \{0,1\}; b' \leftarrow A(m_b \oplus y) : b' = b]$$

because D only outputs 1 when A guesses the bit b correctly. Conditioning on the value of b gives:

$$\begin{aligned} P_1 &= \Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) :\leftarrow A(m_0 \oplus y) = 0] \cdot \Pr[b = 0] \\ &\quad + \Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : A(m_1 \oplus y) = 1] \cdot \Pr[b = 1]. \end{aligned}$$

Using the fact that $\Pr[b = 0] = \Pr[b = 1] = 1/2$ and that

$$\begin{aligned} &\Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : A(m_1 \oplus y) = 1] \\ &= 1 - \Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : A(m_1 \oplus y) = 0] \end{aligned}$$

gives:

$$\begin{aligned} &P_1 \\ &= 1/2 + 1/2 \cdot \Big(\Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : A(m_0 \oplus y) = 0] \\ &\qquad - \Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : A(m_1 \oplus y) = 0]\Big). \end{aligned}$$

But we have seen the quantity in parentheses before! This is exactly $\pm\gamma(k)$ (recall that the absolute value of the expression in parentheses is $\gamma(k)$), the "success probability" of A when attacking our encryption scheme. The key point here is that when the input y given to D is pseudorandom, then the view of A is exactly the view A has when attacking our encryption scheme.

We now look at the second term in (2) (whew!). Let $P_2 \stackrel{\text{def}}{=} \Pr[y \leftarrow \{0,1\}^k : D(y) = 1]$. Just as before, we can express this in terms of how we constructed D:

$$P_2 = \Pr[y \leftarrow \{0,1\}^k; b \leftarrow \{0,1\}; b' \leftarrow A(m_b \oplus y) : b' = b].$$

Just as before (here we omit the details), we eventually get:

$$P_2 = 1/2 + 1/2 \cdot \Big(\Pr[y \leftarrow \{0,1\}^k : A(m_0 \oplus y) = 0] - \Pr[y \leftarrow \{0,1\}^k : A(m_1 \oplus y) = 0]\Big).$$

And we have seen the expression in parentheses before also! This is just the "success probability" of A when attacking *the one-time pad* (since y is now completely random).

And we know that the one-time pad provides perfect secrecy, so that the expression in parentheses has value exactly 0, and $P_2 = 1/2$!

Putting everything together from (2) gives:

$$\begin{aligned}
&\left|\Pr[x \leftarrow \{0,1\}^{k-1}; y = G(x) : D(y) = 1] - \Pr[y \leftarrow \{0,1\}^k : D(y) = 1]\right| \\
&\qquad = |P_1 - P_2| \\
&\qquad = |1/2 \pm \gamma(k)/2 - 1/2| \\
&\qquad = |\pm\gamma(k)/2| \\
&\qquad = \gamma(k)/2.
\end{aligned}$$

In particular, if $\gamma(k)$ was not negligible (i.e., A had non-negligible advantage in "breaking" the encryption scheme) then $\gamma(k)/2$ is not negligible and therefore D has non-negligible advantage in "breaking" the pseudorandom generator (i.e., distinguishing output of G from random). But this contradicts the fact that G is a PRG (note that if A is a PPT algorithm than so is D). So our original assumption must be wrong and no such A can exist; hence, the encryption scheme is secure. ∎