Lecture 5 - Length Extension, CPA Security

Boaz Barak

September 29, 2005

Increasing the output length Our the PRG axiom only guaranteed us a pseudorandom generator with output m larger than n. As far as we know, it may be that $m = n + 1$. It seems to be a lot of trouble to get into for reducing the key size by only one bit!

Fortunately, it turns out we can use a PRG with $m = n + 1$ to construct a PRG with an arbitrary polynomial stretch.

Theorem 1. *Assume that there exists a PRG. Then for every polynomial $p(\cdot)$, there exists a PRG $\mathsf{G} = \{\mathsf{G}_n\}$ with $\mathsf{G}_n : \{0,1\}^n \to \{0,1\}^{p(n)}$.*

(Note that it is highly recommended that you look at Goldreich's (see website) much cleaner and more rigorous exposition of this proof)

Proof. Assume that we have a PRG $PRG' = \{\mathsf{G}'_n\}$ with $\mathsf{G}'_n : \{0,1\}^n \to \{0,1\}^{n+1}$. Let $p(\cdot)$ be a polynomial. We'll construct a PRG $\mathsf{G} = \{\mathsf{G}_n\}$ with $\mathsf{G}_n : \{0,1\}^n \to \{0,1\}^{p(n)}$ with running-time(G) equal to roughly $p(n)$ times the running time of G'.

The algorithm for G_n will be as follows: (notation: for a string $x \in \{0,1\}^m$, and $i < j \leq m$, $x_{[i...j]}$ is $x_i x_{i+1} \cdots x_j$)

Input: $x \in \{0,1\}^n$.

$i \leftarrow 0$
$x^{(0)} \leftarrow x$
`while` $i \leq p(n)$`:`
    $i \leftarrow i + 1$
    $x^{(i)} \leftarrow \mathsf{G}_n(x^{(i-1)}_{[1...n]})$
    `output` $x^{(i)}_{n+1}$

Useful property. We're going to make use of the following property. The *statistical distance* satisfies the following property: If $\Delta(X, Y) \leq \epsilon$ then for every function $f(\cdot)$, $\Delta(f(X), f(Y)) \leq \epsilon$. It turns out that computational indistinguishability satisfies a similar property, as long as $f(\cdot)$ is efficiently computable.

Claim 1.1 (Functions of indistinguishable distributions.)**.** *Let X, Y over $\{0,1\}^m$ such that $X \approx_{T,\epsilon} Y$ and let $f : \{0,1\}^m \to \{0,1\}^{m'}$ be a function computable by a t circuit (for $t < T$). Then, $f(X) \approx_{T-t-100,\epsilon} f(Y)$.*

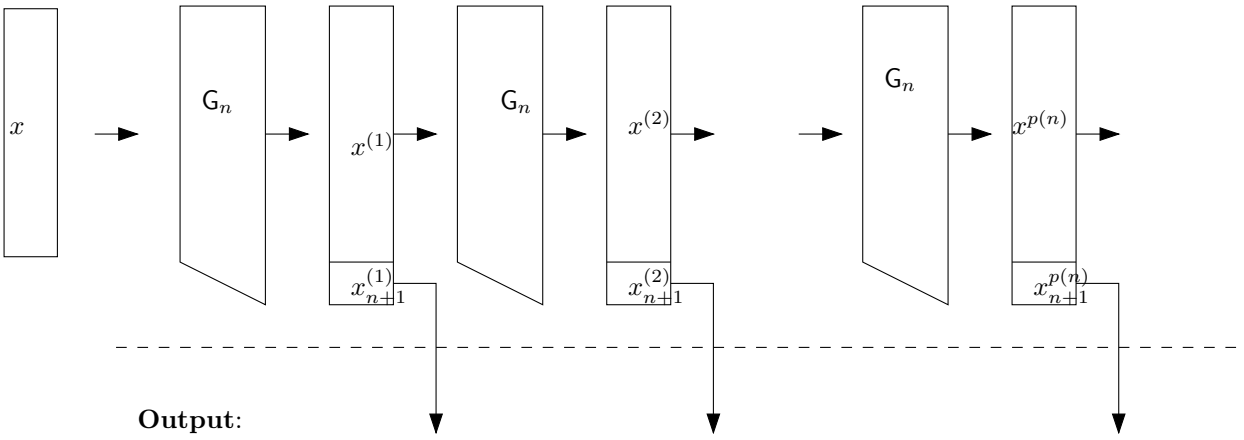

Figure 1: Extending output of pseudorandom genertator

(The proof of the claim is left as an exercise.)

Let $m = p(n)$. We define now the following random variables $Y^{(0)}, \ldots, Y^{(m)}$. The variable $Y^{(i)}$ will range over $\{0,1\}^{n+i}$ and will reflect the state of our pseudorandom generator at the i^{th} step. That is, $Y^{(0)} \triangleq U_n$, $Y^{(1)} \triangleq \mathsf{G}_n(U_n)$, $Y^{(i+1)} = \mathsf{G}_n(Y^{(i)}_{[1\ldots n]})Y^{(i)}_{[n+1\ldots n+1]}$.

We will prove the following claim:

Claim 1.2. *Let t denote the running time of G_n on length-n inputs (note that t is polynomial). Then,*

$$Y^{(m)} \approx_{T-3m^2t,3m\epsilon} U_{n+m}$$

If we prove Claim 1.2 we'll be done since the output of our PRG is simply the last m bits of $Y^{(m)}$ (and its not hard to show that part of a pseudorandom distribution is always pseudorandom).

Claim 1.2 will be proven by simply plugging in $i = m$ in the following claim:

Claim 1.3. *Let t denote the running time of G_n on length-n inputs (note that t is polynomial). For every $1 \leq i \leq m$,*

$$Y^{(i)} \approx_{T-3mti,2i\epsilon} U_{n+i}$$

Proof of Claim 1.3. We prove this by induction. $Y^{(0)}$ is simply equal to U_n so there's nothing to prove in that case. For $Y^{(1)} = \mathsf{G}_n(Y^{(0)})$ the claim follows from the security of G_n. Thus, let $i \geq 1$ and assume that $Y^{(i)} \approx_{T-2ti,2i\epsilon} U_{n+i}$ and we'll prove this for $Y^{(i+1)}$.

Consider the function $f : \{0,1\}^{n+i} \to \{0,1\}^{n+i+1}$ defined as follows: $f(y) = \mathsf{G}_n(y_{[1\ldots n]})y_{[n+1\ldots n+i]}$. That is, $f(Y^{(i)}) = Y^{(i+1)}$. Note that $f(\cdot)$ is computable in $2t$ time (assume $t \geq m$ for convenience). We claim that $f(U_{n+i}) \approx_{T-m,\epsilon} U_{n+i+1}$. Indeed, any $T-m$ sized distinguisher between these two distribution can be turned (by hardwiring the last m bits) into a T sized distinguisher for G_n.

Now by Claim 1.1, this implies that if $Y^{(i)} \approx_{T-2mti,2i\epsilon} U_{n+i}$ then $f(Y^{(i)} \approx_{T-2mti-2t-100,2i\epsilon}$ $f(U_{n+1})$. By transitivity (Claim **??**), we get that

$$f(Y^{(i)}) \approx_{T-2mti-2t-100,2i\epsilon+\epsilon} U_{n+i+1}$$

which implies

$$f(Y^{(i)}) \approx_{T-3mt(i+1),(2i+1)\epsilon} U_{n+i+1}$$

□

□

Note: This proof technique — proving that two distributions X and Y are indistinguishable by presenting *intermediate* distributions $X^{(0)}, \ldots, X^{(m)}$ with $X^{(0)} = X$ and $X^{(m)} = Y$ and the showing that $X^{(i)}$ is indistinguishable from $X^{(i+1)}$ — is called the *hybrid* technique, and is a very important technique in cryptographic proofs. I recommend that you also review the description of the same theorem and proof in Goldreich's book (see course web site for link).

Stronger encryption schemes. By plugging this in we get a *single message* encryption scheme with key arbitrarily smaller than the message. However, in real life we want *multiple* messages. When thinking about multiple messages security, we need to consider the question of where do these messages come from. We can't be sure that they are not affected in some way by our adversary (remember the Brits & Enigma in WWII). Therefore, we want security against *chosen plaintext attack*.

CPA Secure Ecnryption scheme. This is the following game:

- Adversary chooses x_1, x_2.
- Sender chooses $k \leftarrow_R \{0,1\}^n$, $i \leftarrow_R \{1,2\}$ and sends $y = \mathsf{E}_k(x_i)$ to the adversary.
- For as long as adversary desires (but less than T – its running time), adversary chooses x and sees $\mathsf{E}_k(x)$. Note that it is legitimate for the adversary to choose $x = x_1$ or $x = x_2$ but it can also choose other messages.
- Adversary comes up with a guess j. It is *successful* if $i = j$.

(E, D) is (T, ϵ)-CPA secure if for every T-sized adversary, $\Pr[j = i] \leq 1/2 + \epsilon$. We think of a scheme as simply CPA secure if with a key size n it is $(T(n), \epsilon(n))$-CPA secure for superpolynomial $T(\cdot)$ and $\epsilon(\cdot)$.

Note: a deterministic scheme can't be CPA secure (see also exercise).

Constructing a CPA secure scheme. It is not immediate how to construct such a scheme from a pseudorandom generator. To do that, we'll use a new creature called *pseudorandom functions* (PRF). PRFs have many other applications in cryptography and seem quite amazing, but they can be constructed based on any pseudorandom generator.