CSE 594 : Modern Cryptography	2/21/2017

Pseudorandomness-II

Instructor: Omkant Pandey *Scribes: Gustavo Poscidonio, Subathra Vijayakumar*

1 Pseudorandom Generators

Before continuing, let us recall some definitions:

Definition 1 (Pseudorandom Ensembles) *An ensemble $\{X_n\}$, where X_n is a distribution over $\{0,1\}^{\ell(n)}$, is said to be pseudorandom if:*

$$\{X_n\} \approx \{U_n\}$$

That is, X_n is computationally indistinguishable from U_n.

Definition 2 (Next-bit Unpredictability) *An ensemble of distributions $\{X_n\}$ over $\{0,1\}^{\ell(n)}$ is next-bit unpredictable if, for all $0 \leq i \leq \ell(n)$ and non-uniform PPT $\mathcal{A}$, $\exists$ negligible function $\nu(\cdot)$ such that:*

$$\Pr[t = t_1 \dots t_{\ell(n)} \sim X_n : \mathcal{A}(t_1 \dots t_i) = t_{i+1}] \leq \frac{1}{2} + \nu(n)$$

That is, next-bit unpredictability implies that given some prefix of a sample t from X_n it is impossible to predict the next bit of t with probability better than $\frac{1}{2}$.

Theorem 1 (Completeness of Next-bit Test) *If $\{X_n\}$ is next-bit unpredictable then $\{X_n\}$ is pseudorandom.*

Understanding the above recollections allows us to proceed and define a **pseudorandom generator**.

Definition 3 (Pseudorandom Generators) *A deterministic algorithm G is called a pseudorandom generator (PRG) if:*

- *G can be computed in polynomial time*
- $|G(x)| > |x|$
- $\{x \leftarrow \{0,1\}^n : G(x)\} \approx_c \{U_n\}$ *where* $\ell(n) = |G(0^n)|$

*The **stretch** of G is defined as $|G(x)| - |x|$.*

Elaborating on the above definition, we have that G should be efficient, that it should produce some 'extra bits' (hence that it is a *generator*) and the output of G should produce an ensemble which is computationally indistinguishable from the uniform ensemble.

Here, we impose a short term goal upon ourselves: construct a PRG with 1-bit stretch. Doing so will allow us to then extrapolate on that construction and generate polynomially many bits. So consider the hardcore predicate h for some function f. We know that $h(s)$ is hard to guess even if given $f(x)$. So let $G(s) = f(x)||h(s)$. Here we encounter some minor issues:

- $|f(s)|$ might be smaller than s which would prevent G from generating more bits.
- $f(s)$ may always start with some non-random prefix.

We solve both of these issues by letting f be a one-way **permutation** over $\{0,1\}^n$. This way we have that:

- Domain and Range are of the same size. That is, $|f(s)| = |s| = n$.
- $f(s)$ is uniformly random over $\{0,1\}^n$ since f establishes a bijection over $\{0,1\}^n \to \{0,1\}^n$. This prevents $f(s)$ from starting with any fixed value.

Theorem 2 (PRG based on OWP) *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a OWP. Let $h : \{0,1\}^* \to \{0,1\}$ be a hardcore predicate for f. Then we define G to be:*

$$G(s) = f(s)||h(s)$$

G is a pseudorandom generator with 1-bit stretch.

If you did the proof from the previous lecture where the 'next bit test' implies pseudorandomness, then the proof for this statement is trivial. By contradiction you would assume that G is not a PRG. Then an attacker D should succeed in guessing the i^{th} bit of $G(s)$ given the first $i-1$ for some i. But of course the frist n bits of $G(s)$ are uniformly random since f is a permutation, and the $(n+1)^{th}$ bit is the hardcore bit, which is hard to guess. So D can't possibly guess any of the bits from any prefix, so D fails the next bit test which is a contradiction. For completeness, we will provide a complete proof based on hardcore bits.

Proof. First, we know G is computable in polynomial time because f and h are both computable in polynomial time. Additionally, we know that the stretch of G is 1 because $|G(s)| = |f(s)| + |h(s)| = |s| + 1$. All we have to show know is that the output of G computationally indistinguishable from randomly sampled values. That is:

$$\{s \leftarrow \{0,1\}^n : G(s)\} \approx_c \{U_{n+1}\}$$

We begin by assuming to the contrary that this is not true. Then $\exists$ an efficient distinguisher D and a polynomial $q(\cdot)$ such that:

$$|\Pr[s \leftarrow \{0,1\}^n; D(G(s)) = 1] - \Pr[u \leftarrow U_{n+1}; D(u) = 1]| \geq \frac{1}{q(n)}$$

for large enough n. Our goal is to use D to break the OWP f. Let us define $u = u_1 \ldots ||u_{n+1} = y||u_{n+1}$ where $y \in \{0,1\}^n$. Observe that since f is a permutation, $\exists$ a unique s such that $y = f(s)$. And of course, by the bijective properties of f, since y is uniform over $\{0,1\}^n$, s is also uniform over $\{0,1\}^n$. *Note that in the subsequent equations, the domains of each variable in the probability will be omitted to simplify the notation.* So we have:

$$\begin{aligned}
\Pr[D(u)=1] &= \Pr[D(y||u_{n+1})=1] \\
&= \Pr[D(f(s)||u_{n+1})=1] \\
\text{splitting this up for } u_{n+1}=0 & \\
\text{and } u_{n+1}=1 \text{ we have} & \\
&= \sum_{r\in\{0,1\}} \Pr[D(f(s)||u_{n+1})=1 | u_{n+1}=r]\cdot \Pr[u_{n+1}=r] \\
&= \sum_{r\in\{0,1\}} \Pr[D(f(s)||u_{n+1})=1 | u_{n+1}=r]\cdot \frac{1}{2} \\
&= \frac{1}{2}\cdot \sum_{r\in\{0,1\}} \Pr[D(f(s)||u_{n+1})=1 | u_{n+1}=r] \\
&= \frac{1}{2}\cdot \sum_{r\in\{0,1\}} \Pr[D(f(s)||r)=1] \\
&= \frac{1}{2}\cdot (\Pr[D(f(s)||0)=1]+\Pr[D(f(s)||1)=1])
\end{aligned}$$

At this point we are going to substitute $f(s)||0$ and $f(s)||1$ with $f(s)||h(s)$ and $f(s)||\overline{h(s)}$ where $\overline{h(s)} = 1 - h(s)$. We don't know which one is which, but we know that if $h(s) = 0$ then $\overline{h(s)} = 1$ and vice versa. So we have:

$$\Pr[D(u)=1] = \frac{1}{2}\cdot (\Pr[D(f(s)||{\color{red}h(s)}=1]+\Pr[D(f(s)||{\color{red}\overline{h(s)}})=1])$$

We also have that by definition of $G(s)$:

$$\Pr[D(G(s))] = \Pr[f(s)||h(s)]$$

Subtracting the two equations above and taking their absolute value, we have that:

$$|\Pr[D(u)=1]-\Pr[D(G(s))=1]| = \frac{1}{2}\cdot |\Pr[D(f(s)||h(s))=1]-\Pr[D(f(s)||\overline{h(s)})=1]|$$

Playing with the notation, we can rewrite the right-hand side as:

$$\left|\Pr[b\leftarrow\{0,1\}; z\leftarrow X^b; D(z)=b]-\frac{1}{2}\right|$$

where:

$$\begin{aligned}
X^0 &:= \{s\leftarrow\{0,1\}^n : f(s)||h(s)\} \\
X^1 &:= \{s\leftarrow\{0,1\}^n : f(s)||\overline{h(s)}\} \\
z &= f(s)||(h(s)\oplus b)
\end{aligned}$$

So we have that:

$$|\Pr[D(u)=1]-\Pr[D(G(s))=1]| = \left|\Pr[b \leftarrow \{0,1\}; s \leftarrow \{0,1\}^n; D(f(s)||(h(s)\oplus b)) = b] - \frac{1}{2}\right|$$

or, with less verbose notation:

$$|\Pr[D(u)=1]-\Pr[D(G(s))=1]| = \left|\Pr_{b,s}[D(f(s)||(h(s)\oplus b)) = b] - \frac{1}{2}\right|$$

Now we know that the left-hand side $\geq \frac{1}{q(n)}$. Therefore, we have:

$$\left|\Pr_{b,s}[D(f(s)||(h(s)\oplus b)) = b] - \frac{1}{2}\right| \geq \frac{1}{q(n)}$$

Here, we write $r = h(s) \oplus b$ so that r is uniform if b is and $h(s) = r \oplus b$. Making this substitution allows use to manipulate the inequality as follows:

$$\left|\Pr_{r,s}[D(f(s)||r) = b \wedge h(s) = r \oplus b] - \frac{1}{2}\right| \geq \frac{1}{q(n)}$$

An observation we should make here is that we can assume the probability in the inequality is $\geq \frac{1}{2}$ without loss of generality. The reason being that if D's advantage is less than $\frac{1}{2}$, we may always construct D' from D such that the advantage of $D' \geq \frac{1}{2}$. Therefore:

$$\Pr_{r,s}[D(f(s)||r) = b \wedge h(s) = r \oplus b] \geq \frac{1}{2} + \frac{1}{q(n)}$$

Finally, we will use D to break the hardcore bit. Consider the following algorithm:

Algorithm $\mathcal{A}(f(s))$:

1. Sample bit r uniformly and compute $b \leftarrow D(f(s)||r)$
2. Output $r \oplus b$.

Analyzing the probability of success of $\mathcal{A}$, we find:

$$\Pr_s[\mathcal{A}(f(s)) = h(s)] = \Pr_{r,s}[D(f(s)||r) = b \wedge h(s) = r \oplus b] \geq \frac{1}{2} + \frac{1}{q(n)}$$

$\Longrightarrow\Longleftarrow$ Contradiction! You shouldn't be able to predict the hardcore bit with probability better than half. Therefore, $G(s)$ must be a pseudorandom generator, as required. ■

2 One-bit stretch PRG $\Rightarrow$ Poly-stretch PRG

We can do $G(G(G(s)))$ recursively or $G(s_1)G(s_2)...G(s_n)$. Here we present a slightly different version which gives out bits one at a time (without having to wait for the entire output to generate).

Construction of $G_{poly} : \{0,1\}^{(n)} \rightarrow \{0,1\}^{l(n)}$ using a 1-bit stretch PRG G proceeds as follows: output $b_1 b_2 \ldots b_l$ where $G(s_i) = s_{i+1}||b_{i+1}$ yields the bit b_{i+1} for $i = 0$ to $l-1$ and we set $s_0 = s$.

Proof: We prove that G_{poly} is a poly-stretch pseudorandom generator.

$$(i.e)s \leftarrow \{0,1\}^{(n)} : G_{poly}(s) \approx_c U_{l(n)}$$

Suppose not. Then, let D be a non-uniform PPT algorithm which can tell the two distributions above apart with noticeable probability. We use hybrid arguments to show that this cannot be the case.

s is a n-bit seed selected uniform randomly from $\{0,1\}^{(n)}$; let us write $X_0 = s$. Then our first hybrid experiment is really just the output of the distinguisher on the actual PRG value:

$$\underline{Experiment H_0}$$

$$s = X_0$$

$$G(X_0) = X_1||b_1$$

$$G(X_1) = X_2||b_2$$

$$.$$

$$.$$

$$.$$

$$G(X_{l-1}) = X_l||b_l$$

$$\text{Output } D(b_1 b_2 b_3 .. b_{l(n)})$$

Our next hybrid changes the first bit b_1 (of the output of the PRG) to a uniformly random bit u_1 (and the corresponding value X_1 to a random value s_1)

$$\underline{Experiment H_1}$$

$$s = X_0$$

$$X_1||b_1 = s_1||\underline{u_1}$$

$$G(X_1) = X_2||b_2$$

$$.$$

$$.$$

$$.$$

$$G(X_{l-1}) = X_l||b_l$$

$$\text{Output } D(\underline{u_1} b_2 b_3 .. b_{l(n)}).$$

We prove using security of PRG G that H_0 and H_1 can be distinguished with advantage no more than $\mu(n)$ for negligible function μ. For any distinguisher who distinguishes H_0 and H_1 consider the following attacker A for G:

<u>Attacker A</u>:

- A gets a challenge $Z||r$ sampled either as $X_1||b_1$ or $s_1||u_1$ (i.e. either pseudorandom output of G or a uniform string)

- A computes the remaining values as in the construction, i.e., $X_2||b_2 = G(Z)$ and so on for bits $b_2, \ldots, b_l$.
- A outputs the output of $D(r_1 b_2 b_3 .. b_l)$

Note that if $Z||r$ is pseudorandom then output of D produced from previous step is directly identical to the output of H_0. On the other hand, if $Z||r$ is truly random then output of D is distributed identically to the output of H_1. Thus, advantage of A in breaking G is the same as that of D in distinguishing H_0 and H_1. Continuing in this way for each of the next l hybrids, we conclude that the advantage between H_0 and H_l (which will have all uniform bits as output) can be at most $l\mu$. Since the advantage is ϵ, we have that $\epsilon \leq l\mu$. This is a contradiction since $l\mu$ is negligible but ϵ is not.

3 Function vs Generators

PRGs convert one short random string s into one long pseudorandom string. s is a seed and can be used only once. Pseudorandom Fuctions(PRF) can be used instead which will be discussed in the next class.