6 Cryptography (2)

CS 6810 – Theory of Computing, Fall 2012
Instructor: David Steurer
Scribe: Jiaqi Zhai (jz392)
Date: 09/11/2012

6.1 Pseudorandomness

Last time we discussed the ideas of a) using one-time pads for encryption, and b) simulating one-time pad using pseudorandomness. Pseudorandomness is also useful for other things, such as determining if BPP is different from P.

6.1.1 Pseudorandom Distributions and Pseudorandom Generators

Suppose there is a distribution $\mathcal{D}$ over n bits. $\mathcal{D}$ is pseudorandom if and only if it is indistinguishable from true random bits from the perspective of certain classes of algorithms, i.e., it fools certain classes of algorithms.

Definition 6.1 (Pseudorandom Distribution)**.** A distribution $\mathcal{D}$ over $\{0,1\}^n$ is pseudorandom if no polynomial-time machine can distinguish $\mathcal{D}$ from U_n with a polynomial advantage. More formally, define

$$\triangle_{poly}(\mathcal{D},\mathcal{D}') = \max_{\text{poly-time } M} \left| \mathop{\mathbb{P}}_{x\sim\mathcal{D}}[M(x)=1] - \mathop{\mathbb{P}}_{x\sim\mathcal{D}'}[M(x)=1] \right|$$

then we call $\mathcal{D}$ a pseudorandom distribution if $\triangle_{poly}(\mathcal{D},\mathcal{U}_n) \leqslant n^{-w(1)}$.

In other words, if you want a polynomial advantage ($n^{-o(1)}$), you need a machine that runs in superpolynomial time. Or equivalently, every polynomial-time machine can only get subpolynomial advantage ($n^{-w(1)}$).

Definition 6.2 (Pseudorandom Generator)**.** For a polynomial-time computable function $\ell : \mathbb{N} \to \mathbb{N}$ such that $\ell(n) > n$ for every n and a polynomial-time computable function $G_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}$, we say that G_n is a pseudorandom generator (P.R.G.) if $G_n(U_n)$ is pseudorandom.

Note that we actually need a family of pseudorandom generators, $\{G_n\}$. With such generators, we only need a key of length n to encrypt a message of length $\ell(n)$.

6.1.2 Constructing Pseudorandom Generators From One-Way Functions

To construct these generators, we use a primitive called one-way functions. One-way functions have two properties: (a) easy to compute (polynomial-time computable); (b) hard to invert (on average). For our proof we only need to consider the special case of one-way permutations, to simplify our settings:

Definition 6.3 (One-way Permutations)**.** A polynomial-time computable function $f_n : \{0,1\}^n \to \{0,1\}^n$ is a one-way permutation if f is one-to-one, and for all polynomial-time machine M,

$$\mathop{\mathbb{P}}_{x \in_R \{0,1\}^n}[M(f(x)) = x] \leqslant n^{-w(1)}.$$

Now we need to construct pseudorandom generators from one-way functions (permutations). Note that one-way functions are about inverting whereas pseudorandom generators are about distinguishability. It's useful to first prove the following theorem by Yao:

Theorem 6.4 (A distribution is pseudorandom iff it is unpredicable.)**.** *Given a distribution $\mathcal{D}$ over $\{0,1\}^n$. We call $\mathcal{D}$ unpredictable if for every polynomial-time machine M,*

$$\mathop{\mathbb{P}}_{x \sim \mathcal{D}, 0 \leqslant k \leqslant n-1}\left[(M(x_1, \ldots, x_k) = x_{k+1}) - \frac{1}{2}\right] \leqslant n^{-w(1)}.$$

$\mathcal{D}$ is pseudorandom if and only if it is unpredictable.

Proof. First we prove that $\mathcal{D}$ is pseudorandom $\Rightarrow$ $\mathcal{D}$ is unpredictable. This direction is easy; we prove $\mathcal{D}$ is predictable $\Rightarrow$ $\mathcal{D}$ is not pseudorandom. If $\mathcal{D}$ is predictable, then there exists k such that

$$\mathop{\mathbb{P}}_{x \sim \mathcal{D}}[M(x_1, \ldots, x_k) = x_{k+1}] \geqslant \frac{1}{2} + n^{-o(1)}.$$

Consider a polynomial-time machine M' that returns 0 if $M(x_1, \ldots, x_k) \neq x_{k+1}$ or 1 if $M(x_1, \ldots, x_k) = x_{k+1}$. As $\mathbb{P}_{x \in \mathcal{D}}[M'(x) = 1] - \mathbb{P}_{x \in U_n}[M(x) = 1] \geqslant n^{-o(1)}$, $\mathcal{D}$ is distinguishable.

Then we prove that $\mathcal{D}$ is unpredictable $\Rightarrow$ $\mathcal{D}$ is pseudorandom. Similarly we prove the contrapositive. This involves a technique called the *hybrid argument*. Define $(n+1)$ distributions $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_n$ over $\{0,1\}^n$, such that $\mathcal{D}_k$'s first k bits are taken from $\mathcal{D}$ and its remaining $(n-k)$ bits are taken from U_n. Note that $\mathcal{D}_0 = U_n$ and $\mathcal{D}_n = \mathcal{D}$. As $\mathcal{D}$ is not pseudorandom,

$$\triangle_{poly}(\mathcal{D}, U_n) \geqslant n^{-o(1)}.$$

But if $\mathcal{D}$ and U_n are far apart, then one of the pairs $(\mathcal{D}_k, \mathcal{D}_{k+1})$ must also be far apart:

$$\triangle_{poly}(\mathcal{D}, U_n) = \triangle_{poly}(\mathcal{D}_n, \mathcal{D}_0) \leqslant \sum_{i=1}^{n} \triangle_{poly}(\mathcal{D}_i, \mathcal{D}_{i-1}) \Rightarrow \exists k, \triangle_{poly}(\mathcal{D}_k, \mathcal{D}_{k+1}) \geqslant \frac{n^{-o(1)}}{n}.$$

We can then construct a predictor for the $(k+1)$-th bit of $\mathcal{D}$ as follows. First run the polynomial time machine M (the one that satisfies $\triangle_{poly}(\mathcal{D}_k, \mathcal{D}_{k+1}) \geqslant \frac{n^{-o(1)}}{n}$) on $\mathcal{D}^*$, where $\mathcal{D}^*$'s first k bits are from the input to the predictor and $\mathcal{D}^*$'s last $(n-k)$ bits are from the corresponding bits in U_n. If M's output y is 1, output the $(k+1)$-th bit in $\mathcal{D}^*$; otherwise, output (1 - the $(k+1)$-th bit in $\mathcal{D}^*$). We have

$$\begin{aligned}
\mathop{\mathbb{P}}_{x\sim\mathcal{D}}[M(x_1,\ldots,x_k) = x_{k+1}] &= \frac{1}{2}\mathbb{P}[y=1|U_{k+1}=x_{k+1}] + \frac{1}{2}\mathbb{P}[y=0|U_{k+1}\neq x_{k+1}] \\
&= \frac{1}{2} + \frac{1}{2}\left(\mathbb{P}[y=1|U_{k+1}=x_{k+1}] - \mathbb{P}[y=1|U_{k+1}\neq x_{k+1}]\right) \\
&= \frac{1}{2} + \mathbb{P}[y=1|U_{k+1}=x_{k+1}] - Pr[y=1] \\
&= \frac{1}{2} + \mathbb{P}[M(x_1,\ldots,x_k,x_{k+1},U_{k+2},\ldots,U_n)=1] \\
&\quad - \mathbb{P}[M(x_1,\ldots,x_k,U_{k+1},\ldots,U_n)=1] \\
&\geqslant \frac{1}{2} + \frac{n^{-o(1)}}{n}.
\end{aligned}$$

Thus $\mathcal{D}$ is predictable. □

Theorem 6.5. *Given a one-way function (permutation) $f : \{0,1\}^n \to \{0,1\}^n$, there exists a pseudorandom generator that maps $2n$ bits into $2n+1$ bits, namely the function $G(x,r) = f(x), r, \langle x,r\rangle$, where $\langle x,r\rangle$ is defined as x and r's inner product $\sum_{i=1}^{n} x_i r_i$ (mod 2).*

We are interested in this because once we have a pseudorandom generator that extends the length of input by one bit, we will be able to obtain pseudorandom generators that extend the length of input by any polynomial number of bits.

Proof. Assume to the contrary that $G(x,r)$ is not pseudorandom. By Theorem 6.4, $G(x,r)$ is predictable. Note that the first $2n$ bits of $G(x,r)$ are completely random, so if $G(x,r)$ is not pseudorandom, then the only predictable bit in $G(x,r)$ must be its last bit. If this bit is predictable, then for some polynomial time machine M,

$$\mathop{\mathbb{P}}_{x,r\in_R\{0,1\}^n}[M(f(x),r) = \langle x,r\rangle] \geqslant \frac{1}{2} + n^{-o(1)}.$$

Let $\varepsilon = n^{-o(1)}$. Rewrite this as the expected probability over xs,

$$\mathbb{E}_x\left[\mathop{\mathbb{P}}_{r\in_R\{0,1\}^n}[M(f(x),r) = \langle x,r\rangle]\right] \geqslant \frac{1}{2} + \varepsilon.$$

This means that there's some nontrival fraction of xs that such that $\mathbb{P}_{r\in_R\{0,1\}^n}[M(f(x),r) = \langle x,r\rangle] \geqslant \frac{1}{2} + \varepsilon$. We call these xs *good* and the other xs *bad*. Ignore the bad xs. For a particular x, define $g(r) = M(f(x),r)$.

Claim 6.6. There is a polynomial-time (in n and $\frac{1}{\varepsilon}$) algorithm A, such that A computes the short list L of all xs that satisfy $\mathbb{P}_{r\in_R\{0,1\}^n}[g(r) = \langle x,r\rangle] \geqslant \frac{1}{2} + \varepsilon$.

If Claim 6.6 is true, then to invert $f(\cdot)$ we can simply check if $f(x)$ is the preimage of some element a in L. If $|L|$ is small (polynomial in n and $\frac{1}{\varepsilon}$), we will have succeeded in showing a way of inverting $f(\cdot)$ and thus reaching a contradiction.

Remark: there are two techniques of proving Claim 6.6: one is self-correction, the other is Fourier analysis. The fourier analysis approach is simpler and the proof outline is as follows.

Outline. For αs, define function $\chi_\alpha(r) = (-1)^{\langle \alpha, r \rangle}$, mapping 0 to 1 and 1 to -1. We have (for good αs)

$$\langle g, \chi_\alpha \rangle = \mathbb{E}_r[g(r) \cdot \chi_\alpha(r)] \geqslant \varepsilon$$

The crucial thing is that χ_αs form an orthogonal basis. The inner product of g and χ_α is like coefficient of $g(\cdot)$ in that basis. The norm of $g(\cdot)$ is 1 in this context. If the vector has norm 1, then it has very few coordinates larger than ε in any orthogonal basis. The special thing about this particular basis is that the basis function are indexed by these αs. Note that the dimensionality here is exponentially large, so it is impractical to check the coefficients one by one. It turns out that this space allows you to search through it very efficiently - basically binary search. So we are able to find these coefficients in polynomial time. (*Details after class*) □

□