CSE 599b: Cryptography **(Winter 2006)**

Lecture 8: Creating PRNGs and Pseudorandom Function Families

27 January 2006

Lecturer: Paul Beame *Scribe: Paul Beame*

1 Pseudorandom Number Generators from One-Way Permutations

Recall the definition of hard-core bits from last class:

Definition 1.1. *A function $B : \{0,1\}^* \to \{0,1\}$ is a* hard-core bit *for a function f if and only if for every PPT A', the function $\epsilon' : \mathbb{N} \to \mathbb{N}$ is negligible where*

$$\epsilon'(n) = \Pr[A'(f(x), B(x)) = 1 \mid x \leftarrow \mathcal{U}_n] - \Pr[A'(f(x), b)) = 1 \mid x \leftarrow \mathcal{U}_n;\ b \leftarrow \mathcal{U}_1].$$

We saw that a number of our candidate collections of one-way functions (which were permutations) had associated hard-core bits. Also, based on the Goldreich-Levin theorem from last class, given a length-preserving one-way function f one can produce a related function g that has a hard-core bit; moreover, if f is a permutation then so is g. Therefore, given *any* one-way permutation there is a one-way permutation f with an associated polynomial-time computable hard-core bit B. We will now prove that these can be used to derive pseudo-random generator that increase the seed length by a polynomial amount.

Recall that a pseudo-random generator (PRNG) is a deterministic polynomial-time computable function $G : \{0,1\}^* \to \{0,1\}^*$ that is length-increasing (mapping n bits to $\ell(n)$ bits) and such that for all PPT A,

$$\epsilon(n) = \Pr[A(G(\mathcal{U}_n)) = 1] - \Pr[A(\mathcal{U}_{\ell(n)}) = 1]$$

is negligible.

We will now prove part (b) of the following theorem.

Theorem 1.2 (Blum-Micali, Yao)**.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ such that for all n, $f : \{0,1\}^n \to \{0,1\}^n$ is a permutation and B is a hard-core bit for f that is polynomial-time computable then*

(a) $G : \{0,1\}^* \to \{0,1\}^*$ given by $G(x) = f(x)B(x)$ is a PRNG with $\ell(n) = n + 1$.

(b) For every polynomial $\ell(n) > n$ the function $G^\ell : \{0,1\}^* \to \{0,1\}^*$ given by $G^\ell(x) = B(x)B(f(x))B(f(f(x)))\cdots B(f^{\ell(|x|)-1}(x))$ is a PRNG where f^j is the composition of f with itself j times.

Proof. As is typical in these proofs the argument follows by a reduction which allows to turn a statistical test A which distinguishes $G^\ell(\mathcal{U}_n)$ from $\mathcal{U}_{\ell(n)}$ into an algorithm A' that accepts $(f(x), B(x))$

for random x more frequently than it does $(f(x), b)$ for random (x, b). We also will use a hybrid argument.

Let A be such a statistical test and let $\epsilon(n) = \Pr[A(G^\ell(\mathcal{U}_n)) = 1] - \Pr[A(\mathcal{U}_{\ell(n)}) = 1]$

For each $0 \leq k \leq \ell(n)$ define a hybrid distribution $\mathcal{H}_{k,n}$ on $\{0,1\}^{\ell(n)}$ by That is an element of $\mathcal{H}_{k,n}$ is given by $b_1 \cdots b_k B(f^k(x)) \cdots B(f^{\ell(n)-1}(x))$ for x chosen uniformly from $\mathcal{U}_n$ and bits $b_1 \ldots\ b_k$ chosen uniformly from $\mathcal{U}_k$. A key observation is that if f is a permutation then for x chosen from $\mathcal{U}_n$, for any $j \geq 0$, $f^j(x)$ is distributed as a y chosen from $\mathcal{U}_n$. Thus an alternative way to choose an element of $\mathcal{H}_{k,n}$ is as $b_1 \cdots b_k B(f(y) \cdots B(f^{\ell(n)-k}(y))$ or as $b_1 \cdots b_k B(y) \cdots B(f^{\ell(n)-k-1}(y))$ for y chosen uniformly from $\mathcal{U}_n$ and bits $b_1 \ldots\ b_k$ chosen uniformly from $\mathcal{U}_k$; i.e., $\mathcal{H}_{k,n} = \mathcal{U}_k \times G^{\ell-k}(\mathcal{U}_n)$.

Thus $\mathcal{H}_{0,n} = G^\ell(\mathcal{U}_n)$ and $\mathcal{H}_{\ell(n),n} = \mathcal{U}_{\ell(n)}$. Define $p_{k,n} = \Pr[A(\mathcal{H}_{k,n}) = 1]$. By definition,

$$\begin{aligned} \epsilon(n) &= p_{0,n} - p_{\ell(n),n} \\ &= \sum_{k=1}^{\ell(n)} (p_{k-1,n} - p_{k,n}) \end{aligned}$$

is non-negligible.

Now define a PPT A' that uses A to check the supposedly hard-core bit using the hybrid argument.

On input $w = f(y) \in \{0,1\}^n$ and $b \in \{0,1\}$:

1. Choose k uniformly in $\{1, \ldots, \ell(n)\}$.
2. Choose $b_1 \ldots b_{k-1} \rightarrow \mathcal{U}_{k-1}$.
3. Compute $z = B(f(y)) \ldots B(f^{\ell(n)-k}(y))$.
4. Run A on input $b_1 \ldots b_{k-1} b B(f(y)) \ldots B(f^{\ell(n)-k}(y))$ and output 1 if and only if A does.

We now apply the definition of the distinguishing probability of A':

$$\begin{aligned} \epsilon'(n) &= \Pr[A'(f(y), B(y)) = 1 \mid y \leftarrow \mathcal{U}_n] - \Pr[A'(f(y), b) = 1 \mid y \leftarrow \mathcal{U}_n; b \leftarrow \mathcal{U}_1] \\ &= \frac{1}{\ell(n)} \sum_{k=1}^{\ell(n)} (\Pr[A(b_1 \ldots b_{k-1} B(y) B(f(y)) \ldots B(f^{\ell(n)-k}(y))) = 1 \mid y \leftarrow \mathcal{U}_n;\ b_1 \ldots b_{k-1} \leftarrow \mathcal{U}_{k-1}] \\ &\quad - \Pr[A(b_1 \ldots b_{k-1} b B(f(y)) \ldots B(f^{\ell(n)-k}(y))) = 1 \mid y \leftarrow \mathcal{U}_n;\ b_1 \ldots b_{k-1} \leftarrow \mathcal{U}_{k-1};\ b \leftarrow \mathcal{U}_1]) \\ &= \frac{1}{\ell(n)} \sum_{k=1}^{\ell(n)} (\Pr[A(\mathcal{H}_{k-1,n}) = 1] - \Pr[A(\mathcal{H}_{k,n}) = 1]) \\ &= \frac{1}{\ell(n)} \sum_{k=1}^{\ell(n)} (p_{k-1,n} - p_{k,n}) \\ &= \epsilon(n)/\ell(n) \end{aligned}$$

which is non-negligible if and only if $\epsilon(n)$ is since $\ell(n)$ is polynomial in n. □

This construction is particularly convenient as a replacement for the random bits in a one-time pad for symmetric encryption as a stream cipher as follows:

On shared key $K \in \{0,1\}^k$, to send the bit string $M = m_1 \dots m_n$ Alice computes $C \leftarrow M \oplus B(K)B(f(K)) \dots B(f^{n-1}(K))$ and sends C to Bob who computes $M \leftarrow C \oplus B(K)B(f(K)) \dots B(f^{n-1}(K))$. To be able to send additional messages, Bob and Alice can now each replace their shared key K by $K' = f^{n-1}(K)$ and then continue. We will see that this can be shown to be secure provided the total number of bits ever sent is polynomial in the key length, k.

Of the different candidate one-way functions we considered, if it is a one-way function, the Blum squaring function provides a fairly efficient form of pseudorandom generator given:

Once $N = pq$ where p and q are n-bit primes with $p, q \equiv 3 \pmod 4$ chosen, we can generate a random element of QR_N by choosing a random element $x \in \mathcal{Z}_N^*$ and squaring it modulo N. Therefore since $Blum_N$ is a permutation and LSB is a hard-core bit if $Blum_N$ is one-way, the following loop produces the bits of the generator:

```
for i = 1 to ℓ(n)
    x ← x^2 mod N
    output LSB(x)
end for
```

This construction produces one bit of output per modular multiplication (much faster than using the discrete exponentiation example which costs n multiplications) and is pseudorandom if factoring Blum integers is computationally hard. In fact, one can extend the proof of the Blum-Micali-Yao theorem to show that one can produce pseudorandom bits at a higher rate using the hard-core function for $Blum_N$ which outputs the $\log n$ least significant bits of x instead of just $LSB(x)$. One cannot get hard-core functions that produce signficantly more than $\log n$ bits as output so that is the limit of this construction: $\log n$ bits of output per evaluation of the one-way function. Gennaro and Trevisan have shown that for *any* generic construction of PRNGs from one-way functions this is the highest rate achievable per evaluation (unless one could do so without using the one-way function).

2 PRNGs from any One-way Function

The most general theorem relating one-way functions and pseudorandom generators is the following HILL theorem:

Theorem 2.1 (Håstad, Impagliazzo, Luby, Levin)**.** *One-way functions exist if and only if pseudorandom number generators exist.*

This proof of this theorem is quite complicated and is beyond what we will cover in this course. Moreover, Luby and other have shown that these properties are equivalent to the existence of secure symmetric encryption.

3 Pseudorandomness of Ideal Block Ciphers

As we discussed earlier, pseudorandom generators are enough to produce stream ciphers. What about block ciphers? Recall that ideally a block cipher E has the following properties:

- $E_K : \{0,1\}^n \to \{0,1\}$ is easy to compute,
- $E_K^{-1} : \{0,1\}^n \to \{0,1\}$ is easy to compute,
- For any polynomial number of distinct messages $M_1, M_2, \ldots, M_q$ and random $K \in \{0,1\}^k$, the values of $E_K(M_1), \ldots, E_K(M_q)$ should look random (except that they are distinct).

An arbitrary function from $\{0,1\}^n$ to $\{0,1\}^n$ takes $n2^n$ bits to specify so the conditions on E_K suggest that E_K looks like a block of $n2^n$ bits such that *any* choice $n^{O(1)}$ bits out of this exponential length string look random. This is a much more stringent condition than pseudorandom generators which only produce one sequence of $n^{O(1)}$ bits that look random. We will see that we can indeed produce functions with these properties.

4 Pseudorandom Function Families

Let $Func(D, R)$ be the set of all functions from D to R and let $Func(r, s)$ be the set of all functions from $\{0,1\}^r$ to $\{0,1\}^s$.

Definition 4.1. *A* finite keyed function family *F from D to R is a map $F : \mathcal{K} \times D \to R$ where $\mathcal{K}$ is the called key set of F. For $K \in \mathcal{K}$ we write F_K for the function $F(K, \cdot)$.*

From now on for convenience we will assume that the set $\mathcal{K} = \{0,1\}^k$ and $D = R = \{0,1\}^n$ where n is polynomial in k.

Definition 4.2. *An* infinite keyed function family *$\mathcal{F}$ is an infinite sequence $\{F^k\}_{k \geq 1}$ where F^k is a finite keyed function family with $\mathcal{K} = \{0,1\}^k$ and $D = R = \{0,1\}^n$ for $n = \ell(k)$ for some polynomial function ℓ. (The function ℓ is called the length of the function family.)*

We say that $\mathcal{F}$ is polynomial time computable if and only if the map $F(K, x) = F^{|K|}(K, x)$ is computable in deterministic polynomial time. (Since the length of the key K determines k, we also write F_K for $F^k(K, \cdot)$.)

We would like to say that a function family is pseudorandom so we need the notion of a randomly chosen function to compare it to. A random function will be an element from the distribution $\mathcal{F}unc(D, R)$, the uniform distribution on $Func(D, R)$; $\mathcal{F}unc(r, s)$ is defined similarly.

4.1 Function Oracles

For pseudorandomness we also need to have a notion of how an algorithm can test a function. The most natural notion would be to allow the algorithm have access to the function as a black box that it can query repeatedly. That is we design A so that it can call a subroutine that computes

a function from $\{0,1\}^n$ to $\{0,1\}^n$. This black-box or subroutine access to the function is known as *oracle access*. Each oracle call that the algorithm makes will cost it at least one time step so the total number of calls that an algorithm A can make is its running time. If A is an algorithm and f is a function that can be used in A's subroutine calls then we write A^f for the machine A with the subroutine replaced by function f. If $\mathcal{F}$ is a distribution on functions we write $A^{\mathcal{F}}$ for the distribution of the action of A when given an oracle randomly chosen from $\mathcal{F}$.

Observe that for any PPT algorithm A, $A^{\mathcal{F}unc(r,s)}$ can be simulated by a PPT algorithm A' without oracle:

Whenever A makes a call $x \in \{0,1\}^r$ to its oracle f:

1. If x has not been input previously
   (a) Choose s random bits y from $\mathcal{U}_s$.
   (b) Store the fact that $f(x) = y$ in a table.
   (c) Return y
2. Else (x has been previously input)
   (a) Return the value of $f(x)$ from the table.

4.2 Pseudorandom Function Families

Definition 4.3. *An infinite keyed function family $\mathcal{F} = \{F^k\}_{k\geq 1}$ with length ℓ is a* pseudorandom function family *if and only if*

1. *$\mathcal{F}$ is polynomial-time computable; i.e. the function that computes $F(K,x) = F_K(x) = F^{|K|}(K,x)$ for $x \in \{0,1\}^{\ell(|K|)}$ is computable in deterministic polynomial time.*
2. *For all PPT A,*

$$Adv_A^{\mathcal{F}}(k) = \Pr[A^{F_K}(1^k) = 1 \mid K \leftarrow \mathcal{U}_k] - \Pr[A^F(1^k) = 1 \mid F \leftarrow \mathcal{F}unc(\ell(k), \ell(k))]$$

is negligible.

Theorem 4.4 (Goldreich, Goldwasser, Micali)**.** *If PRNGs with factor 2 stretch exist then pseudorandom function families exist with length $\ell(k) = k$.*

Since we have alread seen that existence of one-way functions implies the hypothesis we have the immediate corollary.

Corollary 4.5. *If one-way functions exist then pseudorandom function families exist.*

The GGM construction is a follows:
Given $G : \{0,1\}^* \to \{0,1\}^*$ such that $G : \{0,1\}^k \to \{0,1\}^{2k}$ is a PRNG.
Define $G_0 : \{0,1\}^k \to \{0,1\}$ and $G_1 : \{0,1\}^k \to \{0,1\}$ by

$$G(y) = G_0(y)G_1(y).$$

That is G_0 gives the left half of the output of G and G_1 gives the right half. We can extend this definition for all subscripts $x \in \{0,1\}^*$ by

$$\begin{aligned} G_\lambda(y) &= y \qquad (\lambda \text{ is the empty string}) \\ G_{x0}(y) &= G_x(G_0(y)) \\ G_{x1}(y) &= G_x(G_1(y)). \end{aligned}$$

Each $G_x : \{0,1\}^k \to \{0,1\}^k$.

Now for $x \in \{0,1\}^k$ and $K \in \{0,1\}^k$ define $F(K,x) = F_K(x) = G_x(K)$.

This is easy to compute and we will show next time that it is secure. One might wonder about why the roles of x and K are not reversed. It can be shown that doing things the other way would not be secure.