6.875J/18.425J Cryptography and Cryptanalysis	October 29, 2001
Lecture 14	
Instructor: Anna Lysyanskaya	*Scribed by: Abhinav Kumar*

1 Recap

1.1 PRF definition

A random function is one whose output is fixed at random. A pseudorandom function, intuitively, is one that is not really random, but you cannot tell. This intuition can only make sense if the access to the function is oracle-based, for if we were given the code for a polynomial-time TM which computed the function, we would immediately know that it wasn't a function chosen at random. This notion was introduced by Goldreich, Goldwasser, and Micali, who also gave a construction.

Let $\mathcal{F} : D \mapsto R$ be a function; let $A(\cdot)$ be an algorithm. By $A^{\mathcal{F}}(\cdot)$ we denote an algorithm that makes oracle calls to the function $\mathcal{F}$. More precisely, we may imagine that A has an additional tape Q (for "query"), on which it can write queries to $\mathcal{F}$, and as soon as it is done writing such a query q, an answer to it, namely $\mathcal{F}(q)$ magically appears on this tape.

Definition 1 *A family of functions $F_{\cdot}(\cdot)$ is pseudorandom if F is efficiently computable and for a randomly chosen S, F_S cannot be distinguished from a random function. More precisely, let $\rho(k, l(k), m(k))$ denote a family of random functions from $\{0,1\}^{l(k)}$ to $\{0,1\}^{m(k)}$. $F_{\cdot}(\cdot)$ determines a family of pseudorandom functions from $l(k)$ bits to $m(k)$ bits if $\forall$ PPTF $\{A_k\}$, there exists a negligible function $\nu(k)$ such that*

$$\begin{aligned} \Pr[S \leftarrow \{0,1\}^k; O_0 = F_S &\quad ; \\ R \leftarrow \rho(k, l(k), m(k)); O_1 = R &\quad ; \\ b \leftarrow \{0,1\}; b' \leftarrow A_k^{O_b}(1^k) &\quad : \quad b = b'] \leq 1/2 + \nu(k) \end{aligned}$$

Intuitively, the Turing machine for F and the random k bit seed implicitly defines a string of length $2^{l(k)} \cdot m(k)$.

1.2 The GGM construction of PRFs from PRGs

Suppose G is a PRG that expands k random bits into $2k$ pseudorandom bits. Then here is a tree-like construction of a function, F_S, from $\{0,1\}^k$ to $\{0,1\}^k$ with a k-bit seed:

Since $G(x)$ is length $2k$, we can divide it in half and create G_0 and G_1. $G_0(x)$ is the first k bits of $G(x)$ and $G_1(x)$ is the last k bits of $G(x)$.

Let $S \leftarrow \{0,1\}^k$ be the random seed and $x = x_1 \circ x_2 \circ \ldots \circ x_k$ be a k-bit long number.

Then $F_s(x) \triangleq l_x$ where

$$\begin{aligned}
l_x &= l_{x_1 \circ x_2 \circ \ldots \circ x_k} \\
&= G_{x_k}(l_{x_1 \circ x_2 \circ \ldots \circ x_{k-1}}) \\
&= G_{x_k} G_{x_{k-1}}(l_{x_1 \circ x_2 \circ \ldots \circ x_{k-2}}) \\
&\ldots \\
&= G_{x_k} G_{x_{k-1}} G_{x_{k-2}} \ldots G_{x_1}(S)
\end{aligned}$$

$$\begin{aligned}
l_0 &= G_0(S) \\
l_1 &= G_1(S)
\end{aligned}$$

We proved that this gives us a PRF using a hybrid argument which involved vertical hybrids. In fact, the same proof shows that if x had a polynomial length $p(k)$, then we would still get a PRF $F : \{0,1\}^{p(k)} \rightarrow \{0,1\}^k$.

2 The Decisional Diffie-Hellman assumption and applications

2.1 The Decisional Diffie-Hellman assumption

Let $P(1^k)$ be an algorithm that generates a prime p of length k and an element $g \in \mathbb{Z}_p^*$ such that g has large multiplicative order q. (Let's say that $q = \Omega(\sqrt{p})$.) Since the order divides $p-1$ by Fermat's Little theorem, we can write $p - 1 = aq$ or $p = aq + 1$. Then saying q is large is the same as saying a is small (i.e. $O(\sqrt{p})$. Let $G =< g >$. Let's write $(p, q, g) \leftarrow P(1^k)$.

The **decisional Diffie-Hellman** (DDH) assumption is that, for all adversaries $\{A_k\}$, there exists a negligible function $\nu(k)$ such that

$$\begin{aligned}
&\Pr[(p,q,g) \leftarrow P(1^k); x \leftarrow \mathbb{Z}_q; \\
&\quad y \leftarrow Z_q; z_0 = xy \bmod q; z_1 \leftarrow \mathbb{Z}_q; b \leftarrow \{0,1\}; \\
&\quad b' \leftarrow A_k(p,q,g,g^x,g^y,g^{z_b}) : b = b'] \leq 1/2 + \nu(k)
\end{aligned}$$

In contrast, the **Diffie-Hellman assumption** (DH) is that, if $P'(1^k)$ is an algorithm that generates a prime p and an element $g \in \mathbb{Z}_p^*$, then for all adversaries $\{A_k\}$, there exists a negligible function $\nu(k)$ such that,

$$\begin{aligned}
&\Pr[(p,g) \leftarrow P'(1^k); x \leftarrow \mathbb{Z}_p; y \leftarrow Z_p; \\
&\quad Z \leftarrow A_k(p,g,g^x,g^y) : Z = g^{xy}] \leq \nu(k)
\end{aligned}$$

That is, it's hard to compute g^{xy} given p, g, g^x, g^y.

Finally, the **discrete log** (DL) assumption is that given $p, g \in \mathbb{Z}_p^*$ a generator, and g^x it's hard to compute $x \in \mathbb{Z}_p$. That is,

$\forall$ PPTF $\{A_k\}$, $\exists\, \nu(k)$ negligible,
$\Pr[(p,g) \leftarrow P'(1^k); x \leftarrow \mathbb{Z}_p;$
$\quad z \leftarrow A_k(p,g,g^x) : z = x] \leq \nu(k)$

If an adversary can break the Diffie-Hellman assumption, then it can compute g^{xy} with non-negligible probability, and so it can be used to make an adversary which can break the decisional Diffie-Hellman assumption. In fact, using similar reasoning, we get the following string of implications : DDH $\Rightarrow$ DH $\Rightarrow$ DL $\Rightarrow$ (P $\neq$ NP).

How would we use P = NP to break DL? We want to answer the question "What is x?" if we're given $y = g^x$. So we try to guess x bitwise. We ask "Is there an $x = x_1 \circ \ldots \circ x_k$ such that it's first bit $x_1 = 1$ and $g^x = y$. This problem is in NP, and if P = NP, it's polynomial time decidable. So we can find out whether or not the first bit is 1. And so on to guess the other bits of x. Since x has polynomially many bits, we can find the discrete logarithm in polynomial time. (In fact, since we only need the bits with high probability, all we really need in fact is BPP = NP).

2.2 The PRG construction

We want to first construct a PRG using the DDH assumption. Suppose p, q, g are fixed as in the DDH construction. As a first try, let's take

$$G(x,y) = (g^x \circ g^y \circ g^{xy})$$

The intuition is that if one can't tell g^{xy} from a random g^z (under DDH), then this could be pseudorandom. However, this is not a PRG in the usual sense. Recall that a PRG takes $\{0,1\}^{l(k)}$ into $\{0,1\}^{m(k)}$ and more or less uniformly distributes it. However, here $g^x < p$ and so the first part of the output only covers a fraction $p/2^k$ of the whole space $\{0,1\}^k$, in fact a particular portion of the space, namely those numbers y which are less than p. This fact might enable an adversary to distinguish between the output of G from a truly random distribution.

We could argue that if p is very close to 2^k then the above contradiction doesn't hold. However, there's another problem with this generator. Since g is of order q, g^x is of order dividing q. However, it's easy to show that a random element of $\mathbb{Z}_p^*$ has order $\phi(p) = p-1$ (the Euler phi function) with good probability. In fact, the number of elements of $\mathbb{Z}_p$ of order $\phi(p)$ is $\phi(\phi(p)) = \phi(p-1) = \Omega(\sqrt{p-1})$. So the adversary could just check if its input had order dividing q or not (by raising its input to the power q and checking if it's 1 mod p), and thus distinguish between the output of G and a random distribution.

So we need to modify our PRG somewhat. The solution to our problems just lies in throwing away the first few bits. The construction proceeds as follows:

First odd primes p, q are generated with $p = 2q + 1$ (therefore $p = 3 \bmod 4$).
Suppose x, y are randomly chosen in $\mathbb{Z}_q$. G will output $u \circ v \circ w$ computed as follows.
Let $G_0(x,y) = g^x, g^y, g^{xy}$.
Note that $p = 2q+1$ means that elements of order q are quadratic residues modulo p (since

$x^q = x^{(p-1)/2} = 1 \bmod p$ is the condition for QR). Therefore g^x is a square modulo p and we can extract square roots $\pm a$ (since $p = 3 \bmod 4$). Let u' be the square root which is $\leq (p-1)/2$. Similarly compute square roots $v', w' \leq (p-1)/2$ for g^y and g^z. u', v', w' are k-bit numbers, so truncate the first $k/4$ bits to get u, v, w and return $u \circ v \circ w$. This returns a $9k/4$ bit string (the initial string $x \circ y$ was $2k$ bits long).

2.3 Proof of pseudorandomness over G

The proof is just by reduction. Suppose we have an adversary A who breaks the pseudorandomness assumption. The our new adversary B will break DDH as follows. Its input is a DDH instance p, q, g, g^x, g^y, g^z and it has to tell whether $z = xy$ or z is chosen randomly. B just computes the latter part of the construction given in the previous paragraph (the part relating to square roots), and passes it to A. If z was indeed random along with x and y, the input to A will be "very close to random" (we will quantify this in a minute). However, if $z = xy$ then A is actually passed a valid G instance. So A will be able to distinguish between the two distributions and if B just outputs the answer A gives, B will break DDH.

2.4 Near-randomness of output of G on random inputs

Now let's quantify why, if x, y, z are chosen randomly, then each of u, v, w are close to uniform over $\{0,1\}^{3k/4}$. It suffices to do this for x by symmetry. The output of g^x is a quadratic residue, and then we take a square root $\leq (p-1)/2 = q$. As x ranges uniformly over $\mathbb{Z}_q$ this gives *distinct* values in $\mathbb{Z}_p^*$.
Proof: Let g' be a square root of g in $\mathbb{Z}_p$. Then g' is a generator of $\mathbb{Z}_p^*$, since the order of g' will be $2q = p-1$. Also, a square root of g^x is necessarily $\pm(g')^x$. So $\sqrt{g^x} = \sqrt{g^{x'}}$ implies $(g')^x = \pm(g')^{x'} \bmod p$. So $(g')^{x-x'} = \pm 1 \bmod p$. But $q = (p-1)/2$ is the smallest positive integer t so that $g^t = -1 \bmod p$ and $2q$ is the smallest positive integer such that $g^t = 1 \bmod p$. So in any case, we find $x' = x \bmod q$, i.e. $x' = x \in \mathbb{Z}_q$, which is what we wanted to show.

Now, we've proved the function $x \to \sqrt{g^x}$ is injective from $\mathbb{Z}_q$ to $\{1, ..., q\}$. Since the domain and range have the same number of elements, it's bijective too. So what $\sqrt{g^x}$ produces is a uniform distribution on $\{1, ..., q\}$. Now we start throwing away bits. In the sequel assume s is uniformly distributed on $\{1, ..., q\}$ (i.e. s represents the new distribution after the square root). We want to show that the distribution obtained by taking the last $3k/4$ bits of s, is random. Let's compute the probability that a certain $x \in \{0,1\}^{3k/4}$ is obtained. This is the number of preimages s such that $s = t \cdot 2^{3k/4} + x$. Let's write

$$q = u_0 \cdot 2^{3k/4} + x_0$$

where $u_0 = \lfloor \frac{q}{2^{3k/4}} \rfloor$. If $x < x_0$, then the preimages are exactly $x, x + 2^{3k/4}, ..., x + u_0 2^{3k/4}$, so there are $u_0 + 1$ preimages. If $x > x_0$ the preimages are $x, x + 2^{3k/4}, ..., x + (u_0 - 1)2^{3k/4}$, so there are u_0 preimages.

Therefore, for all $x \in \{0,1\}^{3k/4}$,

$$\frac{u_0}{q} \leq \Pr[x] \leq \frac{u_0 + 1}{q}$$

$$\frac{\lfloor \frac{q}{2^{3k/4}} \rfloor}{q} \leq \Pr[x] \leq \frac{\lfloor \frac{q}{2^{3k/4}} \rfloor + 1}{q}$$

which implies

$$2^{-3k/4} \leq Pr[x] \leq 2^{-3k/4} + \frac{1}{q}$$

so that the probability is almost uniform.

In fact, if the probability distribution above is D, then

$$\sum_x |\Pr_D[x] - 2^{-3k/4}| \quad = \quad x_0\left(2^{-3k/4} - \frac{\lfloor \frac{q}{2^{3k/4}} \rfloor}{q}\right) + (2^{3k/4} - x_0)\left(\frac{\lfloor \frac{q}{2^{3k/4}} \rfloor + 1}{q} - 2^{-3k/4}\right)$$

Now,

$$\lfloor \frac{q}{2^{3k/4}} \rfloor \leq \frac{q}{2^{3k/4}} \leq \lfloor \frac{q}{2^{3k/4}} \rfloor + 1$$

So

$$\frac{\lfloor \frac{q}{2^{3k/4}} \rfloor}{q} \rfloor \leq 2^{-3k/4} \leq \frac{\lfloor \frac{q}{2^{3k/4}} \rfloor}{q} + \frac{1}{q}$$

and therefore $\left(2^{-3k/4} - \frac{\lfloor \frac{q}{2^{3k/4}} \rfloor}{q}\right)$ and $\left(\frac{\lfloor \frac{q}{2^{3k/4}} \rfloor + 1}{q} - 2^{-3k/4}\right)$ are both $\leq \frac{1}{q} \leq \frac{1}{2^{k-2}}$ (remember q is a $k-1$-bit number since p is a k-bit number). So

$$\begin{aligned} \sum_x |\Pr_D[x] - 2^{-3k/4}| \quad &\leq \quad x_0(2^{-k+2}) + (2^{3k/4} - x_0)(2^{-k+2}) \\ &= \quad 2^{3k/4-k+2} = 2^{-k/4+2} \end{aligned}$$

which is a negligible function. Thus, the distribution D is statistically close to a uniform distribution.

2.5 Making the Naor-Reingold PRF

We made a pseudorandom generator which generates a $9k/4 = 2k+k/4$ bit integer from a $2k$ bit integer. To get a PRF, one way is to proceed as follows: the seed is some $S = (a_0, ..., a_k)$. Then set

$$f_S(x_1 \circ x_2 \circ ... \circ x_k) = g^{a_0 \prod_{i=1}^{n} a_i^{x_i}}$$

We will see next time why this is a PRF. This construction is due to Naor and Reingold.

3 Closeness of distributions

We will discuss when two distributions D_1 and D_2 on a domain U can be considered close. The distributions are said to be **identical** if

$$\forall x,\ \Pr_{D_1}[x] = \Pr_{D_2}[x]$$

They are said to be **statistically close** if

$$\exists\, \nu(k) \in negl$$
$$\sum_{x \in U} |\Pr_{D_1}[x] - \Pr_{D_2}[x]| \leq \nu(k)$$

Here we assume that there are distributions $D_{1,k}$ and $D_{2,k}$ defined on domains U_k for all k. The distributions are said to be **computationally indistinguishable** if no PPTF's can distinguish between them with better than non-negligible advantage. You can think about how to formalize this. We will write down the formal definition next time and show that statistical closeness implies computational indistinguishability. That, along with the result of statistical closeness that we proved for the NR construction, will show that we indeed have a PRG. In fact, the following more general result holds:

Lemma 2 *If D_1 and D_2 are statistically close then given a polynomial number of sample from D_b, no (even unbounded) adversary has better than $\nu'(k)$ advantage of guessing b (for some $\nu'(k)$ negligible depending on the polynomial number of samples).*