6.875J/18.425J Cryptography and Cryptanalysis	October 22, 2001
Lecture 12	
Instructor: Anna Lysyanskaya	*Scribed by: Jelena Spasojevic*

1 Recap of definitions

1.1 One-way functions and one-way and trapdoor permutations

Definition 1 *A set of efficient probabilistic algorithms $(G(\cdot), S(\cdot), f.(\cdot))$, specifies a family of one-way functions (OWF family) if for all PPTF $\{A_k\}$ there exists a negligible function $\nu(k)$ such that*

$$\Pr[PK \leftarrow G(1^k); x \leftarrow S(PK); y \leftarrow f_{PK}(x); x' \leftarrow A_k(PK, y) : f(x) = f(x')] \leq \nu(k)$$

and for all $PK \in G$, for all $x \in S(PK)$, f_{PK} is a function.

Definition 2 *A family of one-way functions (G, S, p) is a family of one-way permutations (OWP family) if for all $PK \in G$, p_{PK} is a permutation.*

Definition 3 *A family of one-way permutations (G, S, p) is trapdoor (TDP family) if (1) the output of G consists of two values, denoted (PK, SK); and (2) there is an efficient algorithm D such that for all k,*

$$\Pr[(PK, SK) \leftarrow G(1^k); x \leftarrow S(PK); x' \leftarrow D(SK, p_{PK}(x) : x = x'] = 1$$

Definition 4 *A family of one-way functions (resp., permutations) (G, S, f) is a one-way function (resp., permutation) if G is a deterministic algorithm.*

1.2 Hardcore bit

Definition 5 *Let (G, S, f) be a OWF family. Let $b_{(\circ)}(\circ)$ be a Boolean function. b is a hardcore bit for (G, S, f) if it is hard to compute $b_{PK}(x)$ from the values PK and $f_{PK}(x)$. More precisely, for all PPTF $\{A_k\}$ there exists a negligible function $\nu(k)$ such that*

$$\Pr[PK \leftarrow G(1^k); x \leftarrow S(PK); b' \leftarrow A_k(PK, f_{PK}(x)) : b' = b_{PK}(x)] \leq 1/2 + \nu(k)$$

2 Factoring and RSA-based OWP families and OWFs

(This is continued from previous lecture)

Last time we saw a few candidates based on DL-type assumption. Now let us talk about factoring and RSA.

RSA is a OWP family. RSA is a rather strong assumption. Let me describe the generation algorithm.

On input 1^k generate two k-bit primes p, q and set $n \leftarrow pq$, find l such that $(l, \phi(n)) = 1$. Output $p_{(n,l)}(x) = x^l \bmod n$.

We know how to invert it from pervious lectures.

LSB is hardcore for RSA. This result is due to Alexi, Chor, Goldreich and Schnorr.

But some may say: I can believe factoring is hard, but I am skeptical about RSA. Now we introduce an OWP that is based on weaker assumption, which is that it is hard to factor Blum integers. Let me define Blum integers. Generation algorithm is as same as for RSA with $l = 2$ and with addition we pick up n to be Blum integer, i.e. we pick up p and q in a special way.

Definition 6 *$n = pq$ is a Blum integer iff $p = q = 3 \bmod 4 \wedge p, q$ are primes.*

Now we know that for a Blum integer n, -1 is not a square modulo n but $\left(\frac{-1}{n}\right) = 1$.

Theorem 7 *Squaring is a permutation over $QR(n)$.*

Proof of Theorem 7: It is sufficient to show that it is $1-1$.

Claim 1: Let a be in $QR(n)$. Then at least one square root of a is a quadratic residue.

Why: By fact 10 of lecture 3, $v^2 \equiv a \bmod n$ if and only if $v^2 equiva \bmod p$ and $v^2 equiva \bmod q$. Suppose v is not a square modulo n. Then either it is not a square modulo p, or it is not a square modulo q, or both. Set $v'_p = v \bmod p$ if v is a square modulo p, otherwise $v'_p = -v \bmod p$ and analogously for v'_q. Note that v'_p is such that $(v'_p)^2 \equiv v^2 \bmod p$, and analogously for v'_q. Yet note that v'_p is a square modulo p. Why? Let's compute its Legendre symbol. Suppose v is a square modulo p, then clearly $\left(\frac{v'_p}{p}\right) = \left(\frac{v}{p}\right) = 1$. Suppose, v is not a square modulo p, so $v'_p = -v \bmod p$. Then by Fact 8 of lecture 3, and recalling that $p = 4m + 3$

$$\left(\frac{v'_p}{p}\right) = \left(\frac{-1}{p}\right)\left(\frac{v}{p}\right) = (-1)^{(p-1)/2}(-1) = (-1)^{2m+1}(-1) = (-1)(-1) = 1$$

Analogously, v'_q is also a square modulo q. By the Chinese Remainder Theorem (CRT), there exists a unique $v' \in QR(n)$ such that $v' \equiv v'_p \bmod p$ and $v' \equiv v'_q \bmod q$. And, also by CRT, $(v')^2 \bmod n = v^2 \bmod n = a$.

Claim 2: Let a be in $QR(n)$. At most one square root of a can be a quadratic residue. Proof: this is just by counting: if some quadratic residue had more than one square root that was itself a square, then some quadratic residue would have no such square root, and this would contradict claim 1. ■

Why is this permutation hard to invert? Let me give a reduction that uses assumption that factoring of Blum integers is hard.

Proof of Hard to Invert: Suppose one could invert it, i.e. there is an adversary A that on input (n, x) outputs u such that $u^2 = x \bmod n$ with $\frac{1}{poly}$ probability. Then I claim that the following reduction B produces adversary that factors n.

B: On input (n):

Run A(n, y^2) for random y^2.

If $u = y$ or $u = -y$, repeat
Otherwise output $p = gcd(n, u - y)$ as a factor of n.

Let me prove that this reduction gives a factor of n. So in the last case we get that A produces some u such that:

$$u^2 - y^2 = 0 \bmod n \iff (u - y)(u + y) = 0 \bmod n \iff n|(u - y)(u + y).$$

But now because $u \neq y \wedge u \neq -y$ it follows that n does not divide both $u - y$ and $u + y$. So we must have that one of two prime divisors of n, say p, divides $u - y$. But then $p = gcd(n, u - y)$. ■

Please note that we do not use the fact that n is Blum integer in proof of correctness of this reduction. This fact matters for making permutation over $QR(n)$. The theorem that computing square roots is as hard as factoring is due to Rabin.

Theorem 8 *LSB of this permutation is hardcore. (Due to Alexi, Chor, Goldreich, Schnorr.)*

This is what Blum, Blum, Shub PRG (the best PRG) is based on. The proof based on hardness of factoring is difficult, so here's a simple proof based on QRA. It is due to Blum, Blum and Shub.

Proof of Theorem 8: Reduction follows.

We know that under QRA it is hard to say on input (n, a) if $a \in QR(n)$. Suppose now that we have some adversary $A(n, x^2)$ that can tell us the LSB of x such that

$$x \in QR(n) \wedge x^2 = a^2,$$

with probability $\frac{1}{2} + \epsilon$. Then the following adversary B solves QR problem with probabilty non-negligibly better than $\frac{1}{2}$:
B: On input (n, a), $\left(\frac{a}{n}\right) = 1$:
Run $A(n, a^2) \to b$
If $b = LSB(a)$ output "SQUARE" otherwise guess it is not a square

Observe now that if $x \in Z_n^*$, n is Blum integer then $LSB(x) \neq LSB(-x)$. If n is Blum and Jacobi symbol of x over n is 1 then exactly one of x, -x is in QR(n).

Now see that $p_n^{-1}(a^2) = a$ if $a \in QR(n)$ and is $-a$ otherwise and also that $LSB(p_n^{-1}(a^2) = LSB(a)$ if $a \in QR(n)$ and $LSB(-a)$ otherwise which is equal to $1 - LSB(a)$. ■

Actually, we can show that squaring is a OWF using a weaker assumption, which is that factoring cannot be done efficiently (but not necessarily for Blum integers). More precisely this assumption looks like:

$$\forall A_k \exists \text{ polynomial } p(k) \text{ s.t. } Pr[n \leftarrow \{0,1\}^k; u \neq\leftarrow A_k(n) : u|n] \leq 1 - \frac{1}{p(k)}.$$

This assumption is equivalent that there are always some integers that we cannot factor and it is strictly weaker than any of the above mentioned. But note that if it is *permutations* that we want, this is insufficient.

3 Other interesting facts with no proofs

3.1 Weak one-way functions

Some may still be skeptical, and say: I am willing to believe that you cannot always factor, I am even willing to believe that for any procedure, there is a polynomial fraction of integers that this procedure cannot factor. But I don't believe they happen to have any particular structure like Blum integers.

This is an interesting notion known as a weak one-way function.

Definition 9 *$f(x)$ is a weak OWF iff:*

$$\forall A_k \exists \text{ polynomial } p(k) \text{ s.t. } Pr[x \leftarrow \{0,1\}^k; y \leftarrow A_k(f(x)) : f(y) = f(x)] \leq 1 - \frac{1}{p(k)}.$$

Theorem 10 *One-way functions exist if and only if weak one-way functions exist. [Yao, Goldreich]*

I will omit the proof, but it is a worthwhile exercise. The one-way function is constructed pretty much as one would expect, i.e. by treating the input as a concatenation of $p^(k)$ strings of length k bits each, and then evaluating the weak one-way function on each of these strings. Proving that this results in a one-way function is not straightforward.

Another exercise (this one is easier): Assume one-way functions exist. Can you give an example of a weak one-way function that is not a one-way function?

3.2 OWF $\Rightarrow$ PRG

Here's another interesting theorem for which I won't show you the proof. This is a celebrated result. The proof is widely considered the most complicated proof of Cryptography, that's why I am not giving it. If any of you is willing to actually study it and simplify it, that would be a good PhD thesis. It is due to Håstad and Impagliazzo, Levin, Luby.

Theorem 11 *If a OWF exists, then a PRG exists.*

4 Pseudorandom functions

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. For PRG we have that if you give me a string you cannot tell if it is random or pseudorandom. For PRF if you give me a function (oracle access to the function and query) then I cannot tell if its output is pseudorandom or not.

This interesting notion was introduced by Goldreich, Goldwasser, and Micali, who also gave the first construction based on PRGs.

The intution is like this:

Why is this a useful notion? It gives a more versatile source of pseudorandomness than a PRG. Also we this notion yields message-authentication codes (this will be a HW problem.) Finally, this is a really neat kind of object!

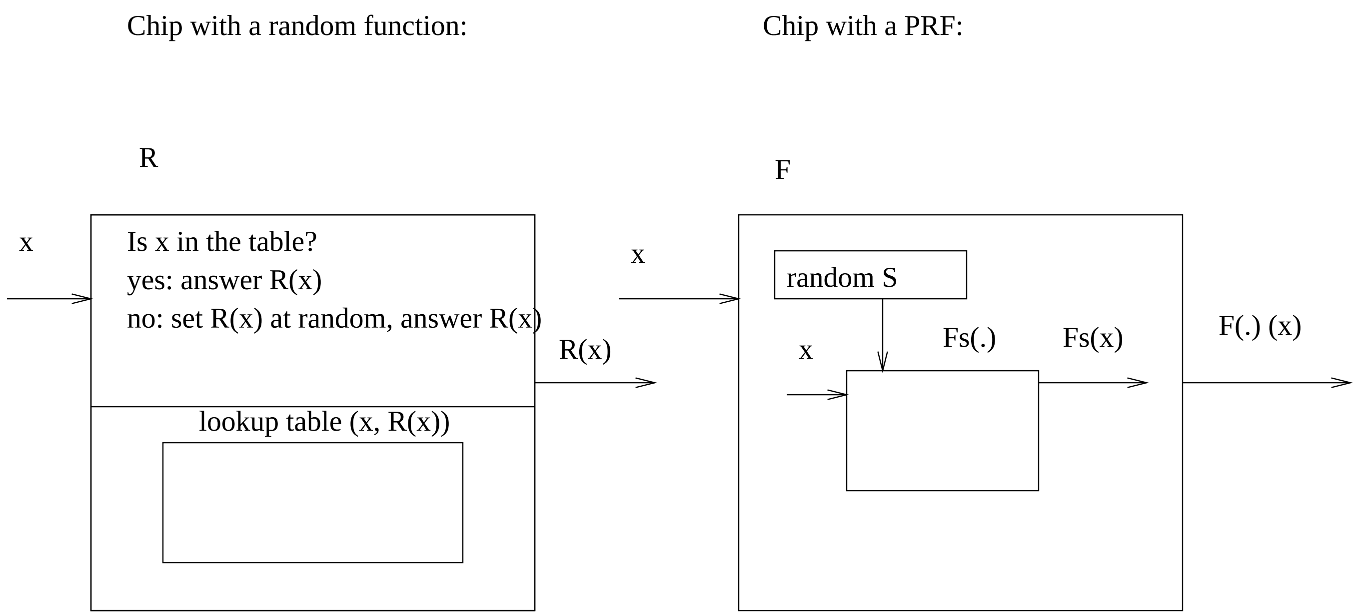

Figure 1: Intuition for PRFs

To define it formally, let us first introduce some notation. Let $\mathcal{F} : D \mapsto R$ be a function; let $A(\cdot)$ be and 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 12 *A family of functions $F_S(\cdot)$ is pseudorandom if*

1. *F is a polynomial-time algorithm.*
2. *For a randomly chosen S, F_S cannot be distinguished from a random function. More precisely, let $\rho(k)$ denote the set of random functions from k bits to $l(k)$ bits. $F.(\cdot)$ is a pseudorandom function if*

$$\forall S \;\; F_S : \{0,1\}^{|S|} \rightarrow \{0,1\}^{l(S)}, 0 < l(|S|) \leq p(|S|)$$

$$\forall A_k \exists \textit{ negligible } \nu(k) \textit{ s.t.}$$

$$Pr[S \leftarrow \{0,1\}^k; O_0 = F_s(\cdot); R \leftarrow \rho(k); O_1 = R(\cdot); b \leftarrow 0,1; b' \leftarrow A_k^{F_b}(1^k); b' = b] \leq \frac{1}{2} + \nu(k).$$