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

Lecture 6: Collections of One-way Functions; Candidates

20 January 2006

Lecturer: Paul Beame *Scribe: Paul Beame*

1 Collections of one-way functions

Definition 1.1. *A* collection of functions *is a set* $\{f_i : D_i \to R_i\}_{\{i \in I\}}$ *for* $I \subseteq \{0,1\}^*$ *such that* D_i *and* R_i *are finite sets for each* $i \in I$.

Definition 1.2. *A collection of functions is* one-way *if and only if*

(0) Sampling:

- *There is a PPT* C_I *that on input* 1^n *produces an element of* $I \cap \{0,1\}^n$.
- *There is a PPT* C_D *that on input* $i \in I$ *produces an element of* D_i.

Note that neither C_I *nor* C_D *is required to be uniform (or even have support that is all of* $I \cap \{0,1\}^n$ *or* D_i *respectively) so all we need is that* C_I *and* C_D *choose elements from the appropriate sets.*

(1) Easy to Compute: There is a deterministic polynomial-time algorithm F *that on input* $i \in I$ *and* $x \in D_i$ *computes* $f_i(x)$.

(2) Hard to Invert: For all PPT A *the function* ϵ *is negligible for*

$$\epsilon(n) = \Pr[A(f_i(x), i) \in f_i^{-1}(f_i(x)) \mid i \leftarrow C_I(1^n); x \leftarrow C_D(i)].$$

Some additional properties of I and D_i that are not essential but are useful in certain circumstances are efficient algorithms that recognize whether $i \in I$ and given (i, x) for $i \in I$ recognize whether $x \in D_i$.

Theorem 1.3. *Collections of one-way functions exist* $\Leftrightarrow$ *one-way functions exist.*

Proof. Problem Set 1. □

2 Candidates for Collections of One-way Functions

2.1 RSA

RSA stands for Rivest-Shamir-Adleman who designed this function which was one of the first candidates suggested. As we will see it is also a trapdoor function which will be useful for public-key cryptography.

The components associated with RSA are:

- $I = \{(N, e) \mid N = p \cdot q,\ p, q \text{ prime}, |p| = |q|,\ 2 < e < N-1,\ \gcd(e, (p-1)(q-1)) = 1\}$.
- $D_{(N,e)} = \{0, \ldots, N-1\} = \mathbb{Z}_N$.
- $RSA_{(N,e)}(x) = x^e \mod N$.

Sampling from I: Choose random n-bit strings with leading bit 1, representing n-bit integers, and test these for primality until two primes p and q are found. The Prime Number Theorem implies that a $\Theta(1/n)$ fraction of n-bit integers are prime. Thus only $O(n)$ trials are needed. To test primality a variant of the randomized Rabin-Miller test is typically used (see Sipser's text for example), although there is now a deterministic polynomial time test due to Agrawal, Kayal, and Saxena. Once p and q have been found compute $N = p \cdot q$ and e is repeatedly chosen uniformly at random from $\{1, \ldots, N-1\}$ and Euclid's algorithm is run until e is found such that $\gcd(e, (p-1)(q-1)) = 1$.

Sampling from D: This is trivial.

Computing F: e is up to $m = 2n$ bits long which is an exponentially large exponent. This is done using repeated squaring and taking results modulo N after each squaring. The total number of modular multiplications required by repeated squaring is m plus the number of 1's in the binary expansion of e.

$RSA_{(N,e)}$ has a number of additional properties that we will use. To discuss these properties we review a little number theory.

Definition 2.1. *Let $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid gcd(x, N) = 1\}$ and define the Euler φ function $\varphi(N) = |\mathbb{Z}_N^*|$.*

Observe that for RSA, $\varphi(N) = (p-1)(q-1)$ and thus the condition on e is that $\gcd(e, \varphi(N)) = 1$. The key property of $\varphi(N)$ we need is:

Theorem 2.2 (Euler's Theorem)**.** *For $x \in \mathbb{Z}_N^*$, $x^{\varphi(N)} \equiv 1 \pmod N$.*

Now since $\gcd(e, \varphi(N)) = 1$ using Euclid's algorithm one can solve $ez \equiv 1 \pmod{\varphi(N)}$ for a $z = d$.

For $x \in \mathbb{Z}_N^*$ and $y = RSA_{(N,e)}(x) = x^e \mod N$ we have

$$\begin{aligned} y^d \mod N &= (x^e)^d \mod N \\ &= x^{e \cdot d} \mod N \\ &= x^{1 + k\varphi(N)} \mod N \qquad \text{for some integer } k \\ &= x^1 \mod N \\ &= x \end{aligned}$$

Thus we see that $RSA_{(N,e)}$ is a *permutation* on $\mathbb{Z}_N^*$ and can view d as a decryption key. It also is reasonable to choose $D_{(N,e)} = R_{(N,e)} = \mathbb{Z}_N^*$. Because of the choice of N, $|\mathbb{Z}_N^*|$ is roughly $N - 2\sqrt{N}$ so only an exponentially small fraction of elements of $\mathbb{Z}_N$ are not in $\mathbb{Z}_N^*$.

Note that the difficulty of inverting RSA depends on the difficulty of factoring N since knowing p and q yields $\varphi(N)$ and thus allows one to find d given e. (In general computing $\varphi(N)$ is as hard as factoring N.)

2.2 Rabin squaring

The associated components are:

- $I = \{N \mid N = p \cdot q,\ p, q \text{ prime}, |p| = |q|\}$.
- $D_N = \{0, \ldots, N-1\} = \mathbb{Z}_N$ (or $\mathbb{Z}_N^*$).
- $Rabin_N(x) = x^2 \mod N$.

Sampling from I and D_N is even simpler than for RSA. Given a $Rabin_N(x)$ there are precisely 4 inverses (square roots) modulo N as follows: By the Chinese Remainder Theorem, $\mathbb{Z}_N$ is equivalent to $\mathbb{Z}_p \times \mathbb{Z}_q$, that is given any $a \in \mathbb{Z}_p$ and $b \in \mathbb{Z}_q$, there is a unique $c \in \mathbb{Z}_N$ such that $c \mod p = a$ and $c \mod q = b$. Also, for $a = z^2 \mod p$, there are precisely two solutions modulo p of $x^2 \equiv a \pmod p$, call them v and $-v$ and for $b = z^2 \mod q$ two solutions modulo q of $x^2 \equiv b \pmod q$, call them w and $-w$. These two pairs of solutions can be combined in 4 ways to get solutions in $\mathbb{Z}_N$. Note that we can pair the solutions to get two solutions x and $-x$ in $\mathbb{Z}_N$ based on (v, w) and $(-v, -w)$ and two solutions y and $-y$ in $\mathbb{Z}_N$ based on $(v, -w)$ and $(-v, w)$.

An algorithm that works well at inverting the Rabin squaring function for N can factor N. Suppose one could get two square roots x and y of the same number as above, such that $x \neq \pm y$ and $x^2 \mod N = y^2 \mod N$. Then $(x^2 - y^2) \equiv 0 \pmod N$ and thus $(x - y)(x + y) \equiv 0 \pmod N$. Since $x \neq \pm y$, $(x - y) \not\equiv 0 \pmod N$ and $(x + y) \not\equiv 0 \pmod N$. Thus we can factor N by computing $\gcd(x - y, N)$. (Note that this idea is also the key to one of the best practical factoring algorithm, the Quadratic Sieve.) Because the inverting algorithm as no idea which square root x of $Rabin_N(X)$ one started with, it can be used to find such a pair x and y.

2.3 Blum Squaring

The associated components are:

- $I = \{N \mid N = p \cdot q,\ p, q \text{ prime}, |p| = |q|,\ p, q \equiv 3 \pmod 4\}$.
- $D_N = QR_N = \{x^2 \mid x \in \mathbb{Z}_N^*\}$. ($QR$ stands for *quadratic residues*.)
- $Blum_N(x) = x^2 \mod N$.

Elements of I are called *Blum integers*. They have the property that $-1 = N - 1 \notin QR_N$ which implies that precisely one of the 4 square roots of an element of QR_N is itself in QR_N. Thus, $Blum_N$ is a permutation of QR_N. It has the same relationship to factoring as Rabin squaring.

2.4 Discrete Log/Exponentiation

The associated components are:

- $I = \{(p, g) \mid p \text{ is prime, } g \text{ is a generator of } \mathbb{Z}_p^*\}$ where we say that g is a generator if and only if $\mathbb{Z}_p^* = \{g^x \mod p \mid x \in \mathbb{Z}_p\}$. Since $g^{p-1} \mod p = 1$ this is the same as saying that $\mathbb{Z}_p^* = \{1 = g^0 = g^{p-1}, g, g^2, \ldots, g^{p-2}\}$ where we drop the $\mod p$ and often assume that this is implicit in the following.
- $D_{(p,g)} = \{0, \ldots, p-2\} = \mathbb{Z}_{p-1}$ or $D_{(p,g)} = \{1, \ldots, p-1\} = \mathbb{Z}_p^*$.
- $EXP_{(p,g)}(x) = g^x \mod p$ (often this is expressed as $DLP_{(p,g)}(x) = g^x \mod p$ even though the term "discrete log" is more appropriate for the inverse problem, namely finding logarithms base g; that is, given $y \in \mathbb{Z}_p^*$ find an x such that $g^x \mod p = y$.

Observe that $EXP_{(p,g)}$ is 1-1 and can be viewed as a permutation of $\mathbb{Z}_p^*$.

Sampling from I: We can find p as in the previous examples but it is not so obvious how to find g. For primes of a certain form specific values of g are know to work however in general there is work to be done. The general idea is to choose random elements h from $\mathbb{Z}_p^*$ and test to see if they are generators. In order for this to work we will need lots of elements of $\mathbb{Z}_p^*$ to be generators.

Since $h \in \mathbb{Z}_p^*$, $h = g^k$ for some unique k, if k and $p-1$ have some common prime factor q then $h^{(p-1)/q} = (g^k)^{(p-1)/q} = (g^{k/q})^{(p-1)} = 1$ since $g^{k/q} \in \mathbb{Z}_p^*$. This will mean that h will only produce at most $(p-1)/q$ different values and so can't be a generator. Furthermore if k and $p-1$ have no common prime factor then $\gcd(k, p-1) = 1$ and k has an inverse j modulo $p-1$. Thus $h^j = (g^k)^j = g^{kj \mod (p-1)} = g$ so h generates all of $\mathbb{Z}_p^*$. Therefore

- h is a generator if and only if $h^{(p-1)/q} \not\equiv 1 \pmod p$ for all prime factors q of $p-1$, and
- there are $\varphi(p-1)$ generators of $\mathbb{Z}_p^*$.

Since every prime in $\{1, \ldots, p-2\}$ is relatively prime to $p-1$, by the Prime Number Theorem, $\varphi(p-1)$ is at least an $\Omega(1/n)$ fraction of $p-1$ where n is the number of bits in p and thus randomly choosing h from $\{1, \ldots, p-1\}$ will produce a generator at least after $O(n)$ trials.

However, in order to run this test for h we need to know the prime factorization of $p-1$. Of course factoring $p-1$ is probably hard so we don't want to do that. One solution would to be use an algorithm (originally due to Eric Bach or a vastly simpler very slick one due to Adam Kalai) that produce a random integer m together with its factorization and then check that $m+1$ is prime. An alternative solution that would be choose p such that $p-1 = 2q$ for some prime q. To do this one would choose random numbers q and check for primality and then check to see if $2q+1$ is prime. Enough of such primes exist for this to work. Note that it is generally believed that the complexity of taking discrete logarithms (inverting this function) is as hard as largest prime factors of $p-1$ so the latter method seems to generate particularly hard problems.

To compute $g^x \mod p$ efficiently one uses repeated squaring as before.

3 Homomorphic Properties

All of the above candidate functions have special properties, namely they are *homomorphisms* as we define below. Functions of this form will have a number of useful properties. To define this we need the notion of a group.

Definition 3.1. *Set G together with an associate binary operation $\bullet$ defined on G together form a* group $(G, \bullet)$ *if and only if*

1. *There is an element $\mathbf{1} \in G$, called an* identity *element, such that for any $x \in G$, $x \bullet \mathbf{1} = \mathbf{1} \bullet x = x$.*

2. *For any $x \in G$ there is a unique $y \in G$, called an* inverse *of x and denoted x^{-1}, such that $x \bullet y = y \bullet x = \mathbf{1}$.*

Definition 3.2. *Given groups $(G, \bullet)$ and $(H, \circ)$, a function $f : G \to H$ is a* homomorphism *from $(G, \bullet)$ to $(H, \circ)$ if and only if for all $x, y \in G$, $f(x \bullet y) = f(x) \circ f(y)$.*

Let's see how each of $RSA_{(N,e)}$, $Rabin_N$, $Blum_N$, and $EXP_{(p,g)}$ is a homomorphism.

$RSA_{(N,e)}$ is a homomorphism from the group $(\mathbb{Z}_N^*, \cdot_N)$ to itself where $\cdot_N$ is integer multiplication modulo N. (In usual notation $\mathbb{Z}_N^*$ actually refers the group itself not just the underlying set as we have used it here so formally the definition of $\bullet$ is implicit in the notation.) Clearly $(x \cdot_N y)^e \mod N = (x^e \mod N) \cdot_N (y^e \mod n)$ as required.

$Rabin_N$ is also a homomorphism from $(\mathbb{Z}_N^*, \cdot_N)$ to itself for a similar reason.

$Blum_N$ is a homomorphism from $(QR_N, \cdot_N)$ to itself for the same reason. (It is not hard to check that QR_N forms a group under multiplication modulo N. It is *subgroup* of $\mathbb{Z}_N^*$.)

$EXP_{(p,g)}$ is a homomorphism from $(\mathbb{Z}_{p-1}, +_{p-1})$ to $(\mathbb{Z}_p^*, \cdot_p)$ where $+_{p-1}$ is integer addition modulo $p-1$ and $\cdot_p$ is integer multiplication modulo p. This follows since $g^{x+y \mod (p-1)} \mod p = ((g^x \mod p)(g^y \mod p)) \mod p$. Note that there is a close correspondence between $\mathbb{Z}_{p-1}$ and $\mathbb{Z}_p^*$: The sets are the same size and the only difference is that 0 in the former set is replaced by $p-1$ in the latter set.

Thus all four of these candidates are homomorphisms. Being homomorphisms implies the following property

Theorem 3.3. *If a collection of functions that satisfies the sampling and easy-to-compute properties of the definition of one-way functions such that*

- *each f_i is a homomorphism from D_i to R_i,*
- *the group operations and inverses on D_i and R_i are polynomial-time computable,*
- *the sampling algorithm C_D samples (nearly) uniformly from D_i*
- *the f_i are weakly hard to invert, i.e. there is a constant c such that for any PPT A and any n, $\epsilon(n) = \Pr[A(f_i(x)) \in f_i^{-1}(f_i(x)) \mid i \leftarrow C_I; x \leftarrow C_D(i)] \leq 1 - 1/n^c$,*

then it is a collection of (strong) one-way functions.

Sketch. The basic idea is called *random self-reduction*. It uses the ability to uniformly sample from D_i and the homomorphic property to convert an $f_i(x) \in R_i$ into an image of a random element of D_i whose relationship to x is known. Using this an algorithm that inverts only a small fraction of the time can be converted to one that inverts almost all the time. This argument is often called a worst case to average case reduction. □

Thus the homomorphic property on groups whose operations are easy to compute and whose elements are easy to sample (nearly) uniformly is enough to say that weak one-wayness already implies strong one-wayness at no cost at all. This is much more efficient than the expensive conversion we described in the previous class for arbitrary one-way functions. Having such a worst-case to average case reduction is one of the hallmarks of good candidates for use in cryptography theory.