CS 282A/MATH 209A: Foundations of Cryptography Prof. Rafail Ostrovsky

Lecture 2

Lecture date: January 12-19, 2005 *Scribe: Ratzin/ Oliviera/ Draganova/ Sung*

1 Introduction to One-Way Functions

1.1 Overview

In secure cryptosystems the user should have the ability to encrypt its data in probabilistic polynomial time; but the adversary should fail to decrypt the encrypt data in probabilistic polynomial time.

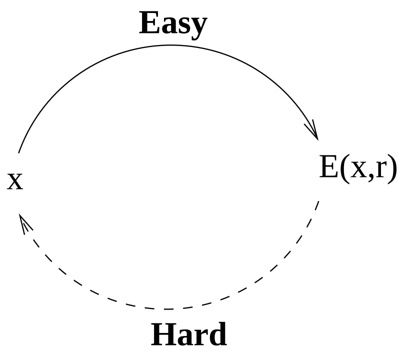

Figure 1: An illustration for a secure cryptosystem

As a motivating example, suppose we wish to construct a cryptosystem consisting of an encryption E and a decryption D, both which are poly-time. The encryption takes a clear-text message x and some random bits r, and gives $y = E(x, r)$. A polynomial-time adversary has access only to the cipher-text y. For the cryptosystem to be secure, it should be hard for the adversary to recover the clear-text, i.e. a poly-time adversary who is given $E(x, r)$ should not be able to figure out any information about x.

This brings up two questions: what assumptions do we need to design that such a cryptosystem, and what is meant by security of the cryptosystem? In this lecture we will answer only the first question.

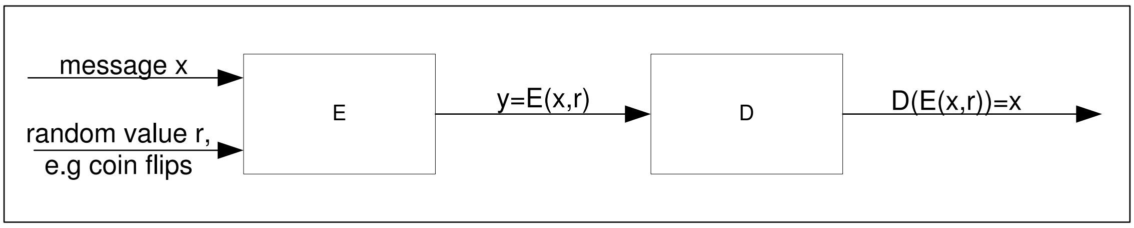

Figure 2: An example for a cryptosystem

1.2 Necessary assumptions

P and NP

For 1-way functions to exist, our first assumption must be that P $\neq$ NP. If we allow the adversary to use non-deterministic polynomial time machines, then A can always break our encryption. If P $\neq$ NP, then we know that there is some function f such that $f^{-1}(y)$ is hard to compute by a poly-time adversary A in some number of instances y. All we can say about the number of those hard instances is that they are infinitely many. If there were only finitely many hard instances, then one can create a look-up table for those hard pairs, and thus get a polynomial time algorithm that easily inverts f in all instances.

BPP and NP

Next, we want to assume that the adversary A is as 'smart' as possible. Since we are allowing our machine to flip coins, we want to assume that A can do that as well, and that A can occasionally make mistakes. Thus, let A be any BPP (bounded probabilistic polynomial time) machine. Our next assumption becomes BPP $\neq$ NP. Since P $\subseteq$ NP[1] this assumption is at least as strong as the assumption P $\neq$ NP. Now we are guaranteed that there is some f such that a BPP adversary fails to invert f on some infinite number of instances. By fails to invert we mean that the probability that A finds some z so that $f(z) = y$ is very small. We make this notion precise in the next section.

Ave-P and Ave-NP

It is important to note that the assumption BPP $\neq$ NP, while necessary, is not sufficient to guarantee the existence of 1-way functions. Even though we know that infinitely many hard instances exist, we may not have a way of actually finding them. Consider the function f defined below, that illustrates the well-known NP-complete problem of determining whether

[1]See Lecture 1 for a detailed discussion.

a given graph G has a Hamiltonian cycle.

$$f(G,H) = \begin{cases} G & \text{if } H \text{ is a hamiltonian cycle of } G, \\ \underbrace{00\ldots0}_{|G|} & \text{otherwise.} \end{cases}$$

If, BPP $\neq$ NP, there may be infinitely many pairs (G,H) for which f is hard to invert, but no poly-time algorithm is known that can generate them.

For a 1-way function to exist, we want hard instances to be easy to find. One way is to ask that 'most' instances are hard. Maybe, for a sufficiently large input size $|x|$, it is hard to invert f for most $f(x)$[2]. Is assuming ave-P $\neq$ ave-NP enough to guarantee the existence of 1-way functions? It is not clear, since it could be the case that whenever we pick y at random, and try to find $f^{-1}(y)$ it is hard, but whenever we pick x at random and ask our enemy to invert $f^{-1}(f(x))$ it is easy. The reason for this is that the distribution of $f(x)$, if we start from the uniform distribution on x, maybe far from uniform. Thus, we need to assume not only that ave-P $\neq$ ave-NP, but also that there are functions which are hard to invert on the uniform distribution of the inputs.

Still, we are only talking about the existence of hard 'unsolved' problems. Given an output y, we want $f^{-1}(y)$ to be hard to compute. For a 1-way function to exist, however, we need hard 'solved' problems. We want an output y for which $f^{-1}(y)$ is hard for the adversary to compute, but we know an answer x such that $f(x) = y$.

In summary, if 1-way functions exist, than it must be that P $\neq$ NP, BPP $\neq$ NP, and ave-P $\neq$ ave-NP; i.e. our assumptions are necessary. It is not know whether they are sufficient.

1.3 Negligible and noticeable functions

When we talk about 1-way functions f, we do not require that f is one-to-one. The same output y can be produced by more than one output x. That is, we can have $f(x) = f(x')$ while $x \neq x'$. We consider the adversary A successful in inverting $y = f(x)$ if A produces some x' such that $f(x') = y$. Formally, A inverts $f(x)$ if

$$A(f(x)) \subseteq f^{-1}(f(x)).$$

$A(f(x))$ is the value x' that A produces given $f(x)$, and $f^{-1}(f(x))$ are all those inputs z for which $f(z) = f(x)$. A is successful if it is able to produce one inverse of $f(x)$. Certainly, if f is one-to-one, then each $f(x)$ has a unique inverse and if A is successful in this case, then A was able to recover x. For simplicity, we sometimes write $Pr_{x,w}[A \text{ inverts } f(x)]$ instead of $Pr_{x,w}[A(f(x)) \subseteq f^{-1}(f(x))]$.

[2]This idea was formalized by Levin as an average case analog of the P vs NP question

Next, we want to give a formal definition of what it means for A to fail. We introduce negligible functions.

Definition 1 *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is* **negligible** *if for all $c > 0$, there exists an integer N_c so that $\forall n \geq N_c$*

$$\epsilon(n) < \frac{1}{n^c}.$$

A negligible function is a function that vanishes faster than the inverse of any polynomial. We say that A fails to invert $f(x)$ if

$$Pr_{x,w}[A(f(x)) \subseteq f^{-1}(f(x))] < \epsilon(n)$$

for some negligible function $\epsilon(n)$, where $n = |x|$.

Here it is worth mentioning that if A has a negligible probability of success, then even if A attempts to invert f a polynomial number of times, its probability of success will not amplify but will remain negligible.

We defined negligible probability of success as occurring with probability smaller than any polynomial fraction. A polynomial probability of success makes a function noticeable (non-negligible).

Definition 2 *A function $\nu : \mathbb{N} \to \mathbb{R}$ is* **noticeable (non-negligible)** *if there exists $c > 0$, and there exists an integer N_c so that $\forall n \geq N_c$*

$$\nu(n) > \frac{1}{n^c}.$$

Noticeable and negligible functions are not perfect negations of each other. There are functions that are neither noticeable nor negligible. For example, the function $f : \mathbb{N} \to \mathbb{R}$ given by

$$f(n) = \begin{cases} n & \text{if } n \text{ is odd,} \\ \frac{1}{2^n} & \text{if } n \text{ is even} \end{cases}$$

is negligible on the even lengths and noticeable on the odd lengths, so overall, f is neither.

2 One-Way Functions

2.1 Informal Definition of One-way Function

The 1-way function problem can be described as a game between a Challenger C (a P-time machine) and an Adversary A (a BPP machine):

1. The Challenger chooses an input length n for a one-way function, which he hopes is "large enough". He then picks x such that $|x| = n$ and computes $y = f(x)$, giving the result y to A.
2. A tries to compute $f^{-1}(y)$ during a polynomial amount of time in the length of $|f(x)|$, and sends its guess z back to the Challenger.
3. A wins if $f(x) = f(z)$, otherwise the Challenger wins. f is a 1-way function if the probability of all BPP adversaries to win is negligible, for a sufficiently big n.

Taking in account the above perspective, we can define a 1-way function f informally:

1. f can be computed in deterministic polynomial time.
2. f is "hard to invert" for all PPT adversaries A.
3. f has polynomially-related input/output.

2.2 Uniform and Non-Uniform One-way Functions

Definition 3 *A function f is said to be a uniform strong one-way function if the following conditions hold:*

1. *f is polynomial-time computable.*
2. *f is hard to invert for a random input: $\forall c > 0$ $\forall A \in PPT$ $\exists N_c$ such that $\forall n > N_c$:*

   $$Pr_{\{x,w\}}[A \textit{ inverts } f(x)] < \frac{1}{n^c}$$

   where "PPT" stands for probabilistic polynomial time, $|x| = n$ and w are coin-flips of the probabilistic algorithm A.
3. *I/O length of f is polynomially related: $\exists \epsilon, c$ such that $|x|^\epsilon < |f(x)| < |x|^c$.*

Condition 3 is necessary to assure that both the Challenger and the Adversary do polynomially related work as a function of their input. Note that the Adversary still has two trivial ways of attacking $f(x)$:

(1) A can always try to invert $f(x)$ by simply guessing what the inputs are and

(2) A can use a huge table to store pairs $(x, f(x))$, sorted, say by the value of $f(x)$.

Neither (1) nor (2) are good strategies for attack since (1) is successful only with a negligible probability and (2) is avoided by requiring that A is of polynomial size.

A *non-uniform* 1-way function is defined exactly as above, except that the adversary is formulated not as a PPT machine, but as a family of poly-size circuits. Recall that a family of poly-size circuits is a set of circuits, one for each input length n, such that the size of each circuit is polynomially related (in size) to the length of the input.

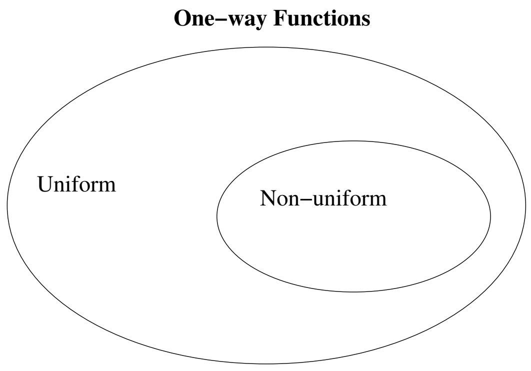

Figure 3: Uniform and non-uniform 1-way functions

Definition 4 *A function f is said to be a non-uniform strong one-way function if the following conditions hold:*

1. *f is polynomial-time computable.*
2. *f is hard to invert: $\forall c > 0$ $\forall$ non-uniform poly-size families A of circuits $\exists N_c$ such that $\forall n > N_c$:*

   $$Pr_x[A \text{ inverts } f(x)] < \frac{1}{n^c}$$

   where $|x| = n$.
3. *I/O length of f is polynomially related: $\exists \epsilon, c$ such that $|x|^\epsilon < |f(x)| < |x|^c$.*

For simplicity, we say that f is a 1-way function when f is a uniform strong 1-way function. We now prove that if a function f is *non-uniform* then it is also *uniform*, hence we have the scenario in Figure 3.

Fact 5 *If f is a non-uniform one-way function, then f is also a uniform one-way function.*

Proof We will prove the contrapositive, i.e., instead of proving $A \Rightarrow B$, we will prove $\neg B \Rightarrow \neg A$. Suppose that f is not a uniform one-way function. Then there exists a constant $c > 0$, and a PPT adversary A such that for an infinite number of integers n, for all strings x of length n,

$$Pr_{\{x,w\}}[A\ inverts\ f(x)] > \frac{1}{n^c}$$

where w are coin-flips of the adversary. Our objective is to find a poly-size collection of circuits A' to substitute our PPT adversary A. Let $\epsilon(n) = \frac{1}{n^c}$ and define the set GOOD to be

$$GOOD = \{x | Pr_w[A\ inverts\ f(x)] > \frac{\epsilon(n)}{2}\}.$$

By conditioning on whether $x \in GOOD$ we get

$$\begin{aligned} Pr_{x,w}[A \text{ inverts } f(x)] &= Pr_{x,w}[A \text{ inverts } f(x) | x \in GOOD] \cdot Pr_x[x \in GOOD] \\ &\quad + Pr_{x,w}[A \text{ inverts } f(x) | x \notin GOOD] \cdot Pr_x[x \notin GOOD] \qquad (1) \end{aligned}$$

Hence,

$$\begin{aligned} Pr_x[x \in GOOD] &= \frac{Pr_{x,w}[A \text{ inverts } f(x)] - Pr_{x,w}[A \text{ inverts } f(x) | x \notin GOOD] \cdot Pr_x[x \notin GOOD]}{Pr_{x,w}[A \text{ inverts } f(x) | x \in GOOD]} \\ &> Pr_{x,w}[A \text{ inverts } f(x)] - Pr_{x,w}[A \text{ inverts } f(x) | x \notin GOOD] \cdot Pr_x[x \notin GOOD] \\ &> Pr_{x,w}[A \text{ inverts } f(x)] - Pr_{x,w}[A \text{ inverts } f(x) | x \notin GOOD] \\ &\geq \epsilon(n) - \frac{\epsilon(n)}{2} \\ &= \frac{\epsilon(n)}{2} \qquad (2) \end{aligned}$$

First, using many attempts of the adversary to invert using fresh coin-flips each time, we can amplify $Pr_w[A \text{ inverts } f(x) | x \in GOOD]$. Then, using the same technique as in the proof of $BPP \subseteq P/poly$, it is possible to show that there are sequences of coin flips r such that A correctly inverts all elements of $GOOD$ on r. Therefore we can hardwire r and build a circuit family A' which inverts at least $\frac{\epsilon(n)}{2}$ of all strings x. Therefore f is not a non-uniform one-way function. ■

3 Number Theory

3.1 Modular Arithmetic

Informally, we can think of modular arithmetic as arithmetic as usual over the integers, except that if we are working modulo n, then every result x is replaced by the element of $0, 1, ..., n-1$ that is equivalent to x, modulo n (that is, x is replaced by x mod n). This informal model is sufficient if we stick to the operations of addition, subtraction, and multiplication. A more formal model for modular arithmetic, which we now give, is best described within the framework of group theory.

Definition 6 *A* group $(S, \bigoplus)$ *is a set S together with a binary operation $\bigoplus$ defined on S for which the following properties hold:*

1. Closure: *For all $a, b \in S$, we have $a \bigoplus b \in S$.*
2. Identity: *There is an element $e \in S$, called the* identity *of the group, such that $e \bigoplus a = a \bigoplus e = a$ for all $a \in S$.*
3. Associativity: *For all $a, b, c \in S$, we have $(a \bigoplus b) \bigoplus c = a \bigoplus (b \bigoplus c)$.*
4. Inverses: *For each $a \in S$, there exists a unique element $b \in S$, called the* inverse *of a, such that $a \bigoplus b = b \bigoplus a = e$.*

As an example, consider the familiar group $(\mathbb{Z}, +)$ of the integers $\mathbb{Z}$ under the operation of addition: 0 is the identity, and the inverse of a is $-a$. If a group $(S, \bigoplus)$ satisfies the *commutative law* $a \bigoplus b = b \bigoplus a$ for all $a, b \in S$, then it is an *abelian group.* If a group $(S, \bigoplus)$ satisfies $|S| < \infty$, then it is a *finite group.*

We give two small facts about finite groups.

Lemma 7 *For any finite group $(G, \cdot)$, $g^m = 1$ for any nonzero $g \in G$ and $m = |G|$.*

Lemma 8 *For any finite group $(G, \cdot)$, $g^x = g^{x \bmod m}$ for any nonzero $g \in G$, $m = |G|$, and $x \in \mathbb{Z}$.*

Proof Let $x = x' \bmod m \Rightarrow x = km + x' \Rightarrow g^x = g^{km+x'} \Rightarrow g^x = g^{km} g^{x'} = 1^k g^{x'} = g^{x'}$. ■

3.2 The multiplicative group Z_N^*

For any positive integer n, let Z_n stand for the set $\{0, 1, 2, \ldots, n-1\}$ of n elements. Define

$$Z_N^* = \{x | 1 \leq x \leq N \text{ and } \gcd(x, N) = 1\}$$

i.e., Z_N^* contains all positive integers less than and relatively prime to N. Z_N^* is a group under multiplication modulo N. The function $\varphi(N) : \mathbf{Z} \to \mathbf{Z}$ defined by

$$\varphi(N) = |Z_N^*|$$

is the so called Euler phi function.

When N is a prime number, say p,

$$Z_p^* = \{1, 2, \ldots, p-1\}$$

and $\varphi(p) = |Z_p^*| = p - 1$.

We are mostly interested in those integers N that are the product of two distinct primes p and q. For $N = pq$,

$$\varphi(N) = \varphi(pq) = |Z_N^*| = (p-1)(q-1).$$

3.3 The Chinese Remainder Theorem for the case $N = pq$

Let $N = pq$ where p and q are distinct prime numbers. The Chinese Remainder Theorem allows us to understand the structure of Z_N^* by considering the 'easier' to work with Z_p^* and Z_q^*.

Theorem 9 *Let $N = pq$ where p and q are distinct prime numbers. Then, the map from Z_N^* to $Z_p^* \times Z_q^*$ given by*

$$x \longmapsto (x \pmod p, x \pmod q)$$

is one-to-one and onto.

By the above theorem, every x in Z_N^* can be written as (x_p, x_q) where $x_p \in Z_p^*$ and $x_q \in Z_q^*$. Conversely, for every element (a, b) in $Z_p^* \times Z_q^*$, there is a unique x in Z_N^* so that $x \equiv a \pmod p$ and $x \equiv b \pmod q$.

Consider, for example, $N = 10$, $p = 2$, and $q = 5$. Then, $|Z_N^*| = (5-1)(2-1) = 4$ and, in particular, $Z_N^* = \{1, 3, 7, 9\}$ while $Z_2^* = \{1\}$ and $Z_5^* = \{1, 2, 3, 4\}$. The bijection is

$$\begin{array}{ccc} 1 & \longmapsto & (1,1) \\ 3 & \longmapsto & (1,3) \\ 7 & \longmapsto & (1,2) \\ 9 & \longmapsto & (1,4). \end{array}$$

In fact, if we know the factorization of N we can simplify computations that have to be performed modulo N into computations modulo p and modulo q. Say that we are trying to multiply two elements x and y of Z_N^* and that p and q are two k-bit primes. Rather than first computing xy and reducing it modulo the $2k$-bit number N, we can instead multiply the corresponding (x_p, x_q) and (y_p, y_q) and thus perform two multiplications modulo k-bit numbers.

Clearly, it is easy to find (x_p, x_q) if we are given x by computing x modulo p and x modulo q. For the above simplification to work, we should also be able to convert back using a polynomial time algorithm. To do so, we first find integers $s, t < N$ so that

1) $s \equiv 1 \pmod{p}$ and $s \equiv 0 \pmod{q}$ and
2) $t \equiv 0 \pmod{p}$ and $t \equiv 1 \pmod{q}$

i.e., we can informally think of s as $(1, 0)$ and of t as $(0, 1)$[3]. Then, given any (a, b) in $Z_p^* \times Z_q^*$ compute $x = as + bt \pmod{N}$. The value x is the unique element of Z_N^* that gets mapped to (a, b).

For example, consider Z_{10}^* again. We get $s = 5$ since $5 \equiv 1 \pmod{2}$ and $5 \equiv 0 \pmod{5}$, and $t = 6$ since $6 \equiv 0 \pmod{2}$ and $6 \equiv 1 \pmod{5}$. To convert, say, $(1, 4)$ from an element of $Z_2^* \times Z_5^*$ into an element of Z_{10}^*, we compute

$$1 \cdot s + 4 \cdot t = 1 \cdot 5 + 4 \cdot 6 \pmod{10} = 9.$$

3.4 Quadratic Residues and quadratic non-residues

We call an element $a \in Z_N^*$ a *quadratic residue (QR) modulo N* if there exists an $x \in Z_N^*$ such that $x^2 \equiv a \pmod{N}$. Informally, we refer to a as a square in Z_N^* and we call x its square root. If a in not a square in Z_N^*, we call a a *quadratic non-residue modulo N*. We let QR_N denote the set of all quadratic residues in Z_N^*. For example, in Z_{10}^*, $QR_N = \{1, 9\}$.

In Z_p^*, where p is an odd prime, exactly half of the elements of Z_p^* are quadratic residues. This fact follows from the following lemma.

Lemma 10 *If p is an odd prime and $a \in Z_p^*$, then a has either 0 square roots or 2 distinct square roots in Z_p^*.*

Proof Take any $a \in Z_p^*$. If a is a quadratic non-residue modulo p, then a has no square roots in Z_p^* and we are done. Otherwise, there is some $x \in Z_p^*$ so that $x^2 \equiv a \pmod{p}$.

[3]Computing s and t is done in polynomial time by using the generalized Euclidean algorithm.

Then, $p-x$ is also in Z_p^* and $(p-x)^2 = p^2 - 2px + x^2$ so that $(p-x)^2 \equiv a \pmod{p}$. Thus, both x and $p-x$ are square roots of a and they are distinct since $x = p-x$ contradicts the fact that p is odd.

Now, if y is yet another square root of a, then $x^2 \equiv y^2 \pmod{p}$ and $p|x^2 - y^2 = (x-y)(x+y)$. Since p is a prime, this implies that either $p|x-y$ or $p|x+y$[4]. That is, $y \equiv x \pmod{p}$ or $y \equiv -x \pmod{p}$ leading to $y = x$ or $y = p-x$. ■

We turn again to the case $N = pq$ where p and q are distinct primes. We require, in addition, that both p and q are odd. It turns out that exactly $\frac{1}{4}$ of the elements of Z_N^* are quadratic residues. Note that this is not true for Z_{10}^* where $|QR_{10}| = 2$ while $|Z_{10}^*| = 4$.

Lemma 11 *If $N = pq$ where p and q are distinct odd primes, and if $a \in Z_N^*$, then a has either 0 square roots or 4 distinct square roots in Z_N^*.*

Proof Again, if a has no square roots, there is nothing to prove. Assume that there exists some $x \in Z_N^*$ so that

$$x^2 \equiv a \pmod{N}. \tag{3}$$

By the Chinese Remainder Theorem, we can write x as (x_p, x_q) and a as (a_p, a_q) in $Z_p^* \times Z_q^*$. But then (3) implies that

$$x_p^2 \equiv a_p \pmod{p} \text{ and } x_q^2 \equiv a_q \pmod{q}$$

i.e., x_p is a square root of a_p in Z_p^* and x_q is a square root of a_q in Z_q^*. By Lemma 10, the only possibilities for x are (x_p, x_q), $(p-x_p, x_q)$, $(x_p, q-x_q)$, and $(p-x_p, q-x_q)$. All four of those are distinct since both p and q are odd. Similar argument using Lemma 10 shows that those are the only square roots of a. Thus, a has exactly 4 distinct square roots in Z_N^*. ■

It is known that computing square roots in Z_p^* can be done in polynomial-time. If we are given the factorization of N as pq, then using the bijection given by the Chinese Remainder Theorem we will see that we can also compute square roots in Z_N^* in polynomial-time. We will show, however, that when the factorization of N is not known, then it is as 'hard' to compute square roots modulo N as it is to factor N.

[4] We are using the fact that if p is a prime number that divides the product ab, then either p divides a or p divides b.

3.5 The Legendre symbol and the Jacobi symbol

Quadratic residues are important enough to prompt the definition of further notation that allows dealing with them . For any prime p, the *Legendre symbol*, $L_p(y)$ is defined to be

$$L_p(y) = \begin{cases} 1 & \text{if } y \text{ is a quadratic residue modulo } p, \\ -1 & \text{otherwise.} \end{cases}$$

For any $N = pq$, where p and q are distinct primes, the *Jacobi symbol*, $J_N(y)$ is defined to be

$$J_N(y) = L_p(y)L_q(y).$$

The Jacobi symbol provides a generalization of the Legendre symbol and can further be defined for any integer. Note that it is not true that $J_N(y) = 1$ implies that y is a quadratic residue modulo N. It could be that $L_p(y) = L_q(y) = -1$ and therefore y is not a quadratic residue modulo N.

We can compute $L_p(y)$ in polynomial-time. If $N = pq$ we can also compute $J_N(y)$ in polynomial-time even if we are given N but not p and q. Even if we have computed that $J_N(y) = 1$, however, no polynomial-time algorithm is known that can determine whether y is a quadratic residue modulo N.

4 The Rabin candidate for a 1-way function

Based on the Number theory background we introduced so far, we can consider the function $f_N : Z_N^* \to QR_N$ given by

$$f_N(x) \equiv x^2 \pmod N$$

where $N = pq$ as before. Note that f_N is not one-to-one but, in fact, is 4-to-1 as shown by Lemma 11.

In 1979, Michael Rabin was the first to show that f_N would be a 1-way function if factorization is 'hard'. In order to prove this, we first show a small fact.

Lemma 12 *Let $N = pq$ where p and q are distinct odd primes. If $x, y \in Z_N^*$ are such that $x \neq \pm y$ and*

$$x^2 \equiv y^2 \pmod N$$

then given x, y, and N, we can efficiently determine p and q;i.e. factor N.

Proof Since $x^2 \equiv y^2 \pmod N$, $N | x^2 - y^2 = (x - y)(x + y)$. On the other hand, $x \neq \pm y$ implies that $x - y \neq 0 \pmod N$ and $x + y \neq 0 \pmod N$. The prime $p | (x - y)(x + y)$

so it must be that $p|x-y$ or $p|x+y$ while N does not divide $x-y$ nor $x+y$. Thus, $\gcd(x-y,N)=p$ or $\gcd(x+y,N)=p$ (It is known that the gcd of two numbers can be computed in poly time). We are done since we were able to find a factor of N. ■

Theorem 13 *(Rabin, 1979) Let $N = pq$ where p and q are distinct odd primes. The function $f_N(x) \equiv x^2 \pmod N$ is a 1-way function if and only if factoring N cannot be done in polynomial time.*

Proof (can factor ⇒ can invert) If N can be factored efficiently, given an output $y \in QR_N$, compute p and q, and then, find the representation (y_p, y_q) in $Z_p^* \times Z_q^*$. Also using a poly-time algorithm we can then find a square root z_p of y_p in Z_p^* and z_q of y_q in Z_q^*. We have produced a square root (z_p, z_q) of (y_p, y_q) in $Z_p^* \times Z_q^*$. Converting (z_p, z_q) back to some $z \in Z_N^*$, we get $f_N(z) = y$. We were able to describe a polynomial time algorithm that inverts f_N which contradicts our assumption that f_N is a 1-way function.

(can invert ⇒ can factor) The converse statement holds the essence of the theorem and requires more work. Assume that F_N is not a 1-way function. Formally, this means that there exists some constant $c > 0$ and some PPT adversary A such that for any integer M, there is some input length $n \geq M$ such that

$$Pr_{x,w}[A \text{ inverts } f(x)] > \frac{1}{n^c} \tag{4}$$

for inputs x of length n. Let $\epsilon(n) = \frac{1}{n^c}$. We will use A to find another PPT algorithm A' for which

$$Pr_{N,w'}[A' \text{ factors } N] > \frac{\epsilon(n)}{2}.$$

The algorithm A' can be described as follows:

(1) Given N, choose some $y \in Z_N^*$, compute $z = y^2 \pmod N$, and give z and N to A.

(2) Take the output $x = A(z, N)$ produced by A and check whether $x^2 = z$ and whether $x \neq \pm y$.

(3) If both of the above are true, use Lemma 12 to factor N. Otherwise, give up.

Certainly, A' runs in polynomial time but we need to consider the probability of success for A'.

$$\begin{array}{rcl} Pr_{N,w'}[A' \text{ factors } N] & = & Pr_{x,w}[A \text{ inverts } f_N] \cdot Pr_x[x \neq \pm y | A \text{ inverts } f_N] \\ & > & \epsilon(n) \cdot \frac{1}{2} = \frac{\epsilon(n)}{2} \end{array}$$

This follows from (4) and from the fact that z has two square roots $\pm y$ and two other square roots $\pm y'$. So, if A gives a square root x of z, then with probability $\frac{1}{2}$ $x \neq \pm y$. Thus, assuming that f_N is not a 1-way function we were able to show that N can then be factored in polynomial time which is a contradiction and we are done. ■

5 Weak One-Way Functions

A one-way function, also called a strong one-way function, is a function that one cannot invert successfully in polynomial time except with negligible probability. A weak one-way function is a function that one cannot invert successfully in polynomial time with noticeable probability.

A motivating example for weak one way function: The problem of factoring $N = pq$ when p, and q are very big prime numbers (about the same number of bits) is widely believed to be a hard problem. What if we define a function $f(x, y) = x \cdot y$ where x and y are big (k bits) random integers. Is f a 1-way function?

Let A be the following poly-time algorithm:

(1) A receives z and checks if $\frac{z}{2}$ is an integer.

(2) If it is, A outputs $(2, \frac{z}{2})$. Otherwise, A gives up.

Since each of x and y are even with probability a half, then the probability that z is even is $\frac{3}{4}$. With certainty, f is not a 1-way function.

What we really refer to when saying that factoring is hard is that the function f above is hard to invert, by density of primes, on a particular part of its domain. The probability for a k-bit integer to be a prime number is $\frac{1}{k}$, making the probability that $f(x, y)$ is a product of two primes $\frac{1}{k^2}$. In this case, it is believed hard to invert f with probability greater than $\frac{1}{n^2}$.

Definition 14 Weak One-Way Functions

f is a **weak one-way function** *if:*

1. *f is polynomial-time computable.*

2. *$\exists\ c > 0\ \forall$ probabilistic polynomial-time A, $\exists N_c$ such that $\forall n > N_c$*

$$P_{r_{w,x}}[A_w(f(x)) \notin\ f^{-1}(f(x))] > \frac{1}{n^c} = \epsilon(n)$$

where $|x| = n$, w are coin-flips of A, and $A_w(f(x)) \notin f^{-1}(f(x))$ means "A does not invert f(x)" .

3. *I/O size is polynomially related.*

Let's now prove the main result about weak and strong 1-way functions [Yao]:

Theorem 15 *There exists a weak one-way function if and only if there exists a strong one-way function.*

Proof First, let us show a trivial direction, i.e., the existence of a strong 1-way function implies that of a weak 1-way function: condition 2 of a strong 1-way function can be re-written as follows: $\forall c > 0\ \forall A \in PPT\ \exists N_c\ s.t.\ \forall n > N_c$:

$$Pr_{\{x,w\}}[A \text{ does not invert } f(x)] > 1 - \frac{1}{n^c} > \frac{1}{n^c}$$

which implies condition 2. of a weak 1-way function.

We now prove the converse: given a weak 1-way function f_0, we will construct a strong 1-way function f_1. We will demonstrate that f_1 is a strong 1-way function by contradiction: we assume an adversary A_1 for f_1 and then demonstrate an effective adversary A_0 for f_0. We can assume that f_0 is length-preserving and maps m bits to m bits. Condition 2. of a weak one-way function f_0 can be restated as:

- $\exists c_{f_0} > 0\ \forall A_0 \in PPT\ \exists M_{c_{f_0}}\ s.t.\ \forall m > M_{c_{f_0}}$:

$$Pr_{\{x,w\}}[A_0 \text{ inverts } f_0(x)] \leq 1 - \frac{1}{m^{c_{f_0}}} = 1 - \epsilon_0(m)$$

  where $|x| = m; w$ are coin-flips of A; and $\epsilon_0(m) \triangleq \frac{1}{m^{c_{f_0}}}$.

To construct f_1, we amplify the "hardness" of weak 1-way function f_0 by applying f_0 in parallel $q \triangleq \frac{2m}{\epsilon_0(m)}$ times:

$$f_1(x_1, ..., x_q) \triangleq f_0(x1), ..., f_0(x_q).$$

where each x_i, $1 \leq i \leq q$ is uniformly and independently chosen m-bit input to f_0. Notice that our f_1 maps $n = \frac{2m^2}{\epsilon_0(m)}$ bits to n bits. We claim that f_1 is a strong 1-way function. The proof is by contradiction. Suppose f_1 is not a strong 1-way function. Then $\exists A_1, \exists c\ s.t.$ for infinitely many input length n,

$$Pr_{\{\vec{x},w\}}[A_1 \text{ inverts } f_1(\vec{x})] > \frac{1}{n^c} \triangleq \epsilon_1(n) \triangleq \epsilon_2(m)$$

Notice that we can redefine $\epsilon_1(n)$ in terms of $\epsilon_2(m)$ since m and n are polynomially related. If we can show how to construct A_0 (using above A_1 as a subroutine) such that A_0 will invert f_0 with probability (over x and w) greater than $1-\epsilon_0(m)$ we will achieve a contradiction with weak one-wayness of f_0. Our algorithm $A_0(f_0(x))$ is as follows:

```
Algorithm $A_0(y)$:
repeat procedure $Q(y)$ at most $\frac{4m^2}{\epsilon_2(m)\epsilon_0(m)}$ times;
stop whenever $Q(y)$ succeeds and output $f_0^{-1}(y)$,
otherwise output “fail to invert”.
```

```
Procedure $Q(y)$:
for $i$ from 1 to $q = \frac{2m}{\epsilon_0(m)}$ do:
STEP1: pick $x_0, ..., x_{i-1}, x_{i+1}, ..., x_q$
(where each $x_j$ is independently chosen $m$-bit number)
STEP2: call $A_1(f_0(x_0), ..., f_0(x_{i-1}), y, f_0(x_{i+1}), ..., f_0(x_q))$
(procedure Q(y) succeeds if $A_1$ above inverts)
```

We must estimate the success probability of $A_0(f_0(x))$, where the probability is over x and coin-flips w of A_0. Define x (of length m) to be BAD if

$$Pr_w[Q(f(x)) \ succeeds] < \frac{\epsilon_2(m)\epsilon_0(m)}{4m}$$

We claim that:

$$Pr_x[x \ is \ BAD] < \frac{\epsilon_0(m)}{2}$$

To show this we assume (towards the contradiction) that $Pr_x[x \ is \ BAD] \geq \frac{\epsilon_0(m)}{2}$. Then

$$\begin{array}{rcl}
Pr_{\{\vec{x},w\}}[A_1 \ inverts \ f_1(\vec{x})] & = & Pr_{\{\vec{x},w\}}[A_1 \ inverts \ f_1(\vec{x}) | some \ x_i \in BAD] \cdot Pr_{\vec{x}}[some \ x_i \in BAD] \\
& + & Pr_{\{\vec{x},w\}}[A_1 \ inverts \ f_1(\vec{x}) | \forall i, x_i \notin BAD] \cdot Pr_{\vec{x}}[\forall i, x_i \notin BAD] \\
& \leq & \sum_{i=1}^{\frac{2m}{\epsilon_0(m)}} [Pr_{\{\vec{x},w\}}[A_1 \ inverts \ f_1(\vec{x}) | x_i \in BAD]] \cdot Pr_{\vec{x}}[\forall i, x_i \notin BAD] \\
& + & Pr_{\{\vec{x},w\}}[A_1 \ inverts \ f_1(\vec{x}) | \forall i, x_i \notin BAD] \cdot Pr_{\vec{x}}[\forall i, x_i \notin BAD] \\
& \leq & \frac{2m}{\epsilon_0(m)}(\frac{\epsilon_2(m)\epsilon_0(m)}{4m}) \cdot 1 + 1 \cdot (1 - \frac{\epsilon_0(m)}{2})^{\frac{2m}{\epsilon_0(m)}} \\
& \leq & \frac{\epsilon_2(m)}{2} + e^{-m} \\
& < & \epsilon_2(m)
\end{array}$$

But we assumed that $Pr_{\{\vec{x},w\}}[A_1 \ inverts \ f_1(\vec{x})] \geq \epsilon_2(m)$ a contradiction. Hence we have shown that $Pr_x[x \ is \ BAD] < \frac{\epsilon_0(m)}{2}$. We are now ready to estimate the failure probability of A_0, using the fact that we try procedure Q in case of failure a total of $\frac{4m^2}{\epsilon_2(m)\epsilon_0(m)}$ times:

$$\begin{array}{rcl} Pr_{\{x,w\}}[A_0 \; does \; not \; invert \; f_0(x)] & = & Pr_{\{x,x\}}[A_0 \; does \; not \; invert \; f_0(x) | x \in BAD] \cdot Pr_x[x \in BAD] \\ & + & Pr_{\{x,w\}}[A_0 \; does \; not \; invert \; f_0(x) | x \notin BAD] \cdot Pr_x[x \notin BAD] \\ & \leq & 1 \cdot \frac{\epsilon_0(m)}{2} + (1 - \frac{\epsilon_2(m)\epsilon_0(m)}{4m})^{\frac{4m^2}{\epsilon_2(m)\epsilon_0(m)}} \cdot 1 \\ & \leq & \frac{\epsilon_0(m)}{2} + e^{-m} \\ & < & \epsilon_0(m) \end{array}$$

Thus $Pr_{x,w}[A_0 \; inverts \; f_0(x)] > 1 - \epsilon_0(m)$ contradicting the assumption that f_0 is a weak one-way function.

■