CS195-7: Introduction to Cryptography	DUE: October 10, 2002
Midterm Exam Solutions	
Instructor: Anna Lysyanskaya	

Problem 1: One-way functions and permutations

Let $f : \{0,1\}^* \mapsto \{0,1\}^*$ be a one-way function. Let $p : \{0,1\}^* \mapsto \{0,1\}^*$ be a one-way permutation.

For each of the suggested implications below, prove or disprove that they are valid. That is to say, if an implication is valid, give a reduction. If it is not valid, give an example of a one-way function f and a one-way permutation p for which the implication is false. You may assume existence of one-way functions permutations.

Problem 2 from Problem Set 2 may serve as a helpful hint for a couple of these problems.

Example. Does it follow that $f(x)$ is a permutation?

Solution. It does not. Let $f'(x)$ be a one-way function. Let $f(x) = g(x)$ where $g(x)$ is as defined in Problem 2a of problem set 2. Then $f(x)$ is a one-way function (that's what is shown in that problem) but it cannot be a permutation because it ignores half of its input bits.

(a) Does it follow that $g(x) = f(f(x))$ is a one-way function?

Solution:

It does not follow. Let $p(x)$ be a one-way permutation. Consider the function $f(x)$:

$$f(x) = \begin{array}{ll} 0^{2|x|} & \text{if } x = y \circ 0^{|x|/2} \\ p(x) \circ 0^{|x|} & \text{otherwise} \end{array}$$

The function $f(x)$ always puts a bunch of zeroes at the end of its output. If its input happens to end with a bunch of zeroes, then it just ignores the input and outputs a string of zeroes. Note that $f(f(x)) = 0^{2|x|}$ for all strings x. Therefore, inverting $g(x) = f(f(x))$ is trivial.

However, note that $f(x)$ is a one-way function. Suppose it is not, and there is an adversary $\{A_k\}$ that inverts it non-negligibly often. Then let us invert the one-way permutation p. On input a random y, we must compute x such that $y = p(x)$. First, note that with high probability, x does not end in $|x|/2$ zeroes. Therefore, running $A_|y|(y \circ 0^{|y|})$ will non-negligibly often produce the value x such that $f(x) = p(x) \circ 0^{|y|} = y \circ 0^{|y|}$, and therefore $p(x) = y$.

(b) Does it follow that $g(x) = p(p(x))$ is a one-way permutation?

Solution:

Yes, it follows. Suppose that we have an algorithm $\{A_k\}$ that computes $g^{-1}(y)$ with non-negligible probability $\epsilon(k)$. Our goal is to build an algorithm $\{B_k\}$ that computes $p^{-1}(y)$ with probability $\epsilon(k)$.

Let $\{B_k\}$ be as follows: on input y, $|y| = k$, run $A_k(p(y))$. With probability $\epsilon(k)$, obtain x such that $z = p(p(x)) = p(y)$. Since p is a permutation, p^{-1} is well-defined, and so $y = p^{-1}(p(y)) = p^{-1}(p(p(x))) = p(x)$, and so x is what we are looking for.

(c) Does it follow that $g(x) = f(x) \circ p(x)$ is a one-way function? (Recall that $\circ$ denotes concatenation.)

Solution:

No, it does not follow. Let $p'(x)$ be a one-way permutation. Let $p(x_1 \circ x_2) = p'(x_1) \circ x_2$, and $f(x_1 \circ x_2) = x_1 \circ p'(x_2)$, where by $x = x_1 \circ x_2$ we denote that x_1 is the most significant $\lceil |x|/2 \rceil$ bits of x, and x_2 is the remaining $\lfloor |x|/2 \rfloor$ bits. It is easy to see that both p and f are one-way permutation. Yet it is easy to compute x from $f(x) \circ p(x)$.

(d) Does it follow that, on input $p(x)$, one can efficiently compute $f(x)$?

Solution:

No, it does not follow. Consider p and f as in part (c). Then computing $f(x)$ from $p(x)$ is equivalent to inverting p'. Suppose an adversary $\{A_k\}$ who computes $f(x)$ from $p(x)$ with probability $\epsilon(k)$ is given. Suppose our input is y and our goal is to find the unique value x such that $p'(x) = y$. The reduction $\{B_{|y|}\}$ proceeds as follows: choose a random x_2 of the appropriate length and run $\{A_k(y \circ x_2)$, where k is chosen either $k = 2|y| - 1$ or $k = 2|y|$, so as to maximize $\epsilon(k)$. With probability $\epsilon(k)$, A_k outputs $f(x \circ x_2) = x \circ p'(x_2)$. $B_{|y|}$ will then output x.

Problem 2: The Blum-Rabin trapdoor permutation

Recall the definition of a family of trapdoor permutations. A trapdoor permutation family consists of algorithms $(G, M_{PK}, f_{PK}, f_{PK}^{-1})$. G generates a member of the family, that is to say, a public key PK that allows to efficiently evaluate the permutation f_{PK}, and the secret key SK that allows to efficiently invert f_{PK}. M_{PK} is the algorithm that efficiently samples the domain of the permutation f_{PK}.

For example, in RSA, the procedure G generates the modulus $n = pq$ and the exponent e, and sets $PK = (n, e)$ and $SK = d$, where $de \equiv 1 \bmod \phi(n)$. Furthermore, $M_{(n,e)} = \mathbb{Z}_n^*$, $f_{(n,e)}(x) = x^e \bmod n$, and $f_{(n,e)}^{-1}(y) = y^d \bmod n$.

$(G, M_{PK}, f_{PK}, f_{PK}^{-1})$ constitute a trapdoor permutation if f_{PK} is hard to invert. More formally, for all probabilistic polynomial-time adversaries $\{A_k\}$, there exists a negligible function $\nu(k)$ such that

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

Consider the following collection of algorithms:

Key generation The procedure $G(1^k)$ generates two k-bit primes, p and q, such that $p \equiv q \equiv 3 \bmod 4$. It outputs $PK = n = pq$, and $SK = (p, q)$. (Such a modulus n is called a *Blum integer*.)

Domain The domain M_n of the permutation f_n consists of all the quadratic residues modulo n. More formally,

$$M_n = \{x \mid x \in \mathbb{Z}_n^* \wedge \exists u \text{ such that } x \equiv u^2 \bmod n\}$$

To sample from the domain, pick $u \leftarrow \mathbb{Z}_n^*$, and output $x = u^2 \bmod n$.

Computing the function The permutation f_n is squaring: $f_n(x) = x^2 \bmod n$.

Inverting the function To compute $f_n^{-1}(y)$, one must compute the value $x \in M_n$ such that $x^2 = y \bmod n$.

In this problem, you will prove that the algorithms given above constitute a family of trapdoor permutations.

(a) Show that f_n is a permutation. (Hint: work modulo p and q first, and then combine using the Chinese remainder theorem.)

Solution:

It is clear that f_n maps quadratic residues to quadratic residues. To show that it is a permutation, we must show that it is invertible.

We must show that each quadratic residue y has a (unique) square root that is itself a square. First, let us show this modulo p and q.

By Fact 6 of lecture notes 5-6, y has exactly two square roots modulo p. If one of them is a, then the other is $-a$, because (1) $a \neq -a$ (since one of them is odd, and the other must be even) and (2) $(-a)^2 = (p-a)^2 = p^2 - 2pa + a^2 = a^2 \bmod p$. Moreover, by Fact 8:

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

i.e., exactly one of them is a square. Similarly, exactly one of y's roots modulo q is a square. Without loss of generality, let these be a and b, respectively.

Let $x \in \mathbb{Z}_n^*$ be such that $x = a \bmod p$ and $x = b \bmod q$. By the Chinese remainder theorem, x exists (and x is unique). Moreover:

1. x is a square root of y: $x^2 = y$. This is because $x^2 = y \bmod p, q$, and therefore by the Chinese remainder theorem, $x^2 = y \bmod n$.

2. x is a quadratic residue: let $\alpha^2 = a \bmod p$ and $\beta^2 = b \bmod q$, then $\xi \in \mathbb{Z}_n^*$ such that $\xi = \alpha \bmod p$ and $\xi = \beta \bmod q$ (by the CRT, it exists) has the property that $\xi^2 = x \bmod n$.

Thus, $f^{-1}(y)$ exists for all quadratic residues y.

(b) Suppose that $p = 4m + 3$ is a prime and that a is a quadratic residue modulo p. Prove that a^{m+1} is a square root of a modulo p.

Solution:

Recall that by Fact 4, $a = g^{2v}$, where g is a generator modulo p. Therefore, $a^{(p-1)/2} = g^{v(p-1)} = (g^v)^{p-1} = 1$ by Fermat's little theorem.

Let us verify that $b = a^{m+1}$ is a square root of a, i.e., that $b^2 = a \bmod p$. Note that $2(m+1) = (2m+1)+1 = (p-1)/2+1$.

$$b^2 = a^{2(m+1)} = a^{(p-1)/2+1} = a^{(p-1)/2}a = a$$

(c) Devise an efficient algorithm that, on input (p, q, y), computes $x = f_{pq}^{-1}(y)$, i.e., x such that $x^2 = y \bmod n$, where $n = pq$ is a Blum integer.

Solution:

The algorithm will first use part (c) to compute square roots of y modulo p and q, denote them a and b respectively. Note that a and b are themselves quadratic residues, since they were obtained by exponentiating a quadratic residue y.

We will then compute $x \in \mathbb{Z}_n^*$ such that $x = a \bmod p$ and $x = b \bmod q$ using the Chinese remainder theorem. By Fact 10, x is a quadratic residue. By the CRT $x^2 = y$, since $x^2 = a^2 = y \bmod p$ and $x^2 = b^2 = y \bmod q$.

(d) Devise an efficient algorithm that, on input (n, a, b), where $a, b \in \mathbb{Z}_n^*$, $a \neq \pm b \bmod n$, and $a^2 \equiv b^2 \bmod n$, outputs a non-trivial divisor of n.

Solution:

Note that $a^2 = b^2 \bmod n$ implies that $(a-b)(a+b) = \bmod n$. Therefore $pq \mid (a-b)(a+b)$. But we are given that $pq \nmid (a-b)$ and $pq \bmod (a+b)$. Therefore, WLOG, $p \mid (a-b)$ and $q \mid (a+b)$. So we can output $p = \gcd(a-b, n)$.

(e) Let us assume that factoring Blum integers is infeasible. More precisely, assume that for all probabilistic polynomial-time adversaries $\{A_k\}$, there exists a negligible function $\nu(k)$ such that

$$\Pr[(n,(p,q)) \leftarrow G(1^k); p \leftarrow A_k(n) \; : \; p \mid n \wedge 1 < p < n] = \nu(k)$$

Show that under this assumption, it is infeasible to invert f_n. More precisely, show that for all probabilistic polynomial-time adversaries $\{A_k\}$, there exists a negligible funtion $\nu'(k)$ such that

$$\Pr[(n,(p,q)) \leftarrow G(1^k); y \leftarrow M_n; x \leftarrow A_k(n) \; : \; x^2 = y \bmod n] = \nu'(k)$$

(This fact is due to Michael Rabin.)

Solution:

Suppose we are given an adversary A who runs in polynomial time and inverts f_n with probability ϵ. We will construct an algorithm that also runs in polynomial time and factors n with probability $\epsilon/2$. If ϵ is a non-negligible function of the length of the modulus n (i.e., if f_n is not a family of OWP), this will contradict the hardness of factoring.

Note that, since each quadratic residue in $\mathbb{Z}_n^*$ has exactly four square roots, the following experiments are equivalent':

- Experiment 1: choose $u \in \mathbb{Z}_n^*$ uniformly at random.

- Experiment 2: a random quadratic residue y and let u be chosen at random from y's four square roots.

The algorithm is as follows:

1. Choose $u \leftarrow \mathbb{Z}_n^*$.
2. Let $v \leftarrow A(n, u^2 \bmod n)$.
3. If (1) $v^2 = u^2 \bmod n$ and (2) $v = \pm u \bmod n$, output $\gcd(u - v, n)$.

It is clear by part (d) that if the algorithm terminates, it outputs the factorization of n. We must now analyze the expected time this algorithm must run for before terminating.

The probability that (1) is satisfied is ϵ, because it is just the probability that A inverts f_n on a random input. The probability that (2) is satisfied given that (1) is satisfied, is $1/2$: since we can imagine that first $y = u^2$ is chosen, then v is computed, and then u is chosen as a random one of the square roots of y, and so the probability that $v = \pm u$ is $1/2$. Therefore, the probability that we factor n is $\epsilon/2$.