.2

CS 276 Cryptography **Spring 2004**

Lecture 12.2: March 3

Lecturer: David Wagner *Scribe: Diana Michalek*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

12.1 Security of Rabin's Function

Part 1 of this lecture introduced Rabin's function. We now show that this function is secure if factoring is hard. First consider the following lemma.

Lemma 12.1 *If $x^2 \equiv y^2 \bmod n$ and $x \neq \pm y \bmod n$, then $gcd(x+y,n)$ and $gcd(x-y,n)$ disclose the factors p and q of n.*

Proof: By a lemma in the first part of the lecture, x^2 has 0 or 4 square roots in $(\mathbb{Z}/n\mathbb{Z})^*$. So since x is a square root, x^2 has three more roots: $-x$, wx, and $-wx$.

Assume that $y \equiv wx \ (mod\, n)$. (The case when $y \equiv -wx \ (mod\, n)$ is similar.) Then

$$y \equiv x \ (mod\, p) \tag{12.1}$$
$$x + y \equiv 2x \ (mod\, p) \tag{12.2}$$

In particular, $2x \neq 0 \ (mod\, p)$, since equivalence only occurs if $x \equiv 0 \ (mod\, p)$, which is highly unlikely in $(\mathbb{Z}/n\mathbb{Z})^*$. Also,

$$y \equiv -x \ (mod\, q)$$
$$x + y \equiv 0 \ (mod\, q)$$

The above equations show that $x + y | q$ but $x + y \nmid p$, which means $gcd(x + y, n) = q$. Subtracting y from both sides in Eq. (12.1) and subtracting $2y$ to both sides in Eq. (12.4) gives:

$$0 \equiv x - y \ (mod\, p) \qquad\qquad x - y \equiv -2y \neq 0 \ (mod\, q)$$

Hence $gcd(x - y, n) = p$. Thus $gcd(x + y, n)$ and $gcd(x - y, n)$ disclose factors of n. ∎

Let A be a deterministic algorithm that finds a square root of x^2. So given input x^2, A outputs y such that $y^2 = x^2$. In other words, y equals one of the four square roots x, $-x$, wx or $-wx$. The lemma indicates that if we can obtain one of x and $-x$ and one of wx and $-wx$ from A, then we can easily factor n by computing $gcd(x \pm wx, n)$.

One problem that can be raised is whether algorithm A ever discloses the square roots wx, $-wx$. We show that when the input to A is taken over random $x \in (\mathbb{Z}/n\mathbb{Z})^*$, we can get an algorithm that factors n with probability 1/2.

Claim 12.2 *Given the deterministic square-root-finding algorithm A, we can get an algorithm that factors n with probability 1/2.*

Proof: Let A be a deterministic square-root finding algorithm. Define

$$S_1 = \{x : A(x^2) = x\} \qquad S_2 = \{x : A(x^2) = -x\}$$
$$S_3 = \{x : A(x^2) = wx\} \qquad S_4 = \{x : A(x^2) = -wx\}$$

Note that S_1, S_2, S_3, and S_4 are disjoint and span $(\mathbb{Z}/n\mathbb{Z})^*$. Consider the map defined by $f(x) = -x$. The signature of f is $f : S_1 \to S_2$ since if $A(y^2) = y$, then $A((-y)^2) = y$ and so $f(y) \in S_2$. Also, f is bijective. Therefore, $|S_1| = |S_2|$. Similar arguments prove that $|S_1| = |S_2| = |S_3| = |S4|$. Thus if x is chosen uniformly at random from $(\mathbb{Z}/n\mathbb{Z})^*$, $Pr[x \in S_1] = Pr[x \in S_2] = Pr[x \in S_3] = Pr[x \in S_4] = 1/4$. So just run A on a random input x^2 from $(\mathbb{Z}/n\mathbb{Z})^*$. A will return a desired square root wx or $-wx$ with probability 1/2, which can then be used to factor n. ■

The following theorem proves that if A is a deterministic square-root-finding algorithm that succeeds with probability p and factoring is hard, p must be very small.

Theorem 12.3 *If factoring is hard, $Pr[A(n, x^2 mod\, n) = x]$ is small.*

Proof: Let $p = Pr[A(n, x^2 mod\, n)^2 = x^2]$ be the probability that algorithm A outputs a square root.

Just like in the claim, $(\mathbb{Z}/n\mathbb{Z})^*$ can be partitioned into four equally sized sets S_1, S_2, S_3, S_4 corresponding to the four square root classes that $A(x^2)$ maps to. Hence when x is chosen uniformly at random from $(\mathbb{Z}/n\mathbb{Z})^*$, x belongs to each set S_i with equal probability, so

$$\begin{aligned} Pr[A(n, x^2 mod\, n) = x] &= \frac{p}{4} \\ Pr[A(n, x^2 mod\, n) = -x] &= \frac{p}{4} \\ Pr[A(n, x^2 mod\, n) = wx] &= \frac{p}{4} \\ Pr[A(n, x^2 mod\, n) = -wx] &= \frac{p}{4} \end{aligned}$$

Since the last two probabilities sum to $\frac{p}{2}$, A outputs useful information (i.e. the square roots $y = wx$ or $y = -wx$) with probability $\frac{p}{2}$. So by the lemma, $gcd(x + y, n)$ and $gcd(x - y, n)$ give the factors p and q of n with probability $\frac{p}{2}$. Repeating this process by querying A $O(1/p)$ times gives an algorithm that factors n with probability $O(p)$. Assuming that factoring is hard, $O(1/p)$ must be large (or not polynomial in n). Thus the probability p that A succeeds in finding a square root of x is very small. ■

Note that RSA is used more often than Rabin's function in practise, but for no clear reason. Rabin's function is more efficient to run (squaring x is faster than computing a higher power of x) and only depends on hardness of factoring rather than the RSA assumption.

12.2 IND-CPA for Public Key

First we consider a faulty definition of security for a public key encryption scheme.

Recall the definition of security in the left-or-right sense, where the advantage of an attacker A against an encryption scheme E_k is defined to be

$$Adv^{lr}A = Pr_{k \leftarrow K}[A^{E_k \circ \S_0} = 1] - Pr_{k \leftarrow K}[A^{E_k \circ \S_1} = 1]$$

For a public key encryption scheme E_{pk}, the above definition becomes:

$$AdvA = Pr[A^{E_{pk} \circ \S_0} = 1] - Pr[A^{E_{pk} \circ \S_1} = 1]$$

This definition is pretty useless because any attacker should know the public key pk. Instead we use the following:

Definition 12.4 *The advantage of an adversary A in indistinguishability under chosen-plaintext attack, or IND-CPA, in the left-or-right sense, is given by:*

$$AdvA = Pr_{(pk,tk) \leftarrow G()}[A^{E_{pk} \circ \S_0}(pk) = 1] - Pr_{(pk,tk) \leftarrow G()}[A^{E_{pk} \circ \S_1}(pk) = 1]$$

An encryption scheme E is (t, q, ϵ)-IND-CPA-secure if $AdvA \leq \epsilon$ for all algorithms A running in time $\leq t$ and using $\leq q$ queries to an oracle.

Note that the above definition is equivalent to $Adv^{lr}A = 2\left|Pr[A^{E_{pk} \circ \S_b} = b] - \frac{1}{2}\right|$

12.3 Public Key Example

We consider a sample public key encryption scheme using a trapdoor one-way permutation and see that it is insecure.

Let (G, F, F^{-1}) be a trapdoor one-way permutation. Define

$$E_{pk}(m) = F_{pk}(m)$$
$$D_{tk}(c) = F_{tk}^{-1}(c)$$

Is this a secure public key encryption scheme? I.e. Is it IND-CPA in the left-or-right sense? The answer is no because it is possible for F to leak partial information about the message and still have F^{-1} be hard to compute. For instance, some bits may be unchanged or easy to compute. If any bits leak into the ciphertext, the encryption is insecure. Also, if F is deterministic, the encryption scheme can be broken immediately. For instance, an attacker can encrypt the messages "yes" and "no" and match these against the future ciphertext sent.

So can a trapdoor one-way function help build a public key encryption scheme that is secure? The following section shows that these functions can help with encryption.

12.4 Hardcore Bits

The intuition behind a hardcore bit is that if any information about a bit leaks into the ciphertext, it is detrimental for encryption. So an attacker shouldn't be able to guess partial information about the plaintext, even probabilistically. This motivates the definition of a hardcore bit.

Definition 12.5 *If $f : X \to X$ is bijective and $B : X \to \{0, 1\}$, then B is (t, ϵ)-hardcore for f if for all algorithms A running in time $\leq t$,*

$$AdvA = 2\left|Pr_{x \leftarrow X}[A(f(x)) = B(x)] - \tfrac{1}{2}\right| \leq \epsilon$$

It follows from the above definition that a boolean predicate is hardcore if an adversary cannot predict the value of the predicate with probability greater than 1/2. This definition is adapted to trapdoor one-way permutations which will be discussed in the next lecture:

12.5 Hardcore for a Trapdoor one-way Permutation

Definition 12.6 *Let (G, F, F^{-1}) be a trapdoor one-way permutation and let $B_{pk} : X \to \{0, 1\}$. Then B is (t, ϵ)-hardcore for (G, F, F^{-1}) if for all algorithms A running in time $\leq t$,*

$$AdvA = 2\left|Pr_{(pk,tk) \leftarrow G(), x \leftarrow X}[A(pk, F_{pk}(x)) = B_{pk}(x)] - \tfrac{1}{2}\right| \leq \epsilon$$