T-79.5502 Advanced Course in Cryptology

Lecture 4, Part 1, November 13, 2007
Bit Security (Section 9)

- RSA Bit
- ElGamal Bit
- DL Bit

Reductions

Recall:

Definition 4.10. We say that a language L is polynomially reducible to another language L_0 if there exists a deterministic polynomially-bounded Turing machine M which will convert each instance $I \in L$ into instance $I_0 \in L_0$, such that $I \in L$ if and only if $I_0 \in L_0$.

Assume L_0 is a problem and L is a problem, which is known to be hard. A polynomial reduction of L to L_0 proves that L_0 is at least as hard as L. This gives us a way of proving security.

The goal of this lecture is to show that all-or-nothing security is equivalent to bit-security assuming that encrypted plaintexts are random.

Two security notions are called equivalent if the problem of breaking one is polynomially reducible to the breaking of the other, and vice versa.

RSA Bit Security

For an arbitrary RSA system, consider two problems:

(I) Given the RSA encryption of a message, retrieve the message.

(II) Given the RSA encryption of a message, retrieve the least significant bit of the message.

Theorem 9.1. Problems (I) and (II) are polynomially reducible to each other. That is, all-or-nothing security and bit security for RSA are equivalent security notions.

Halving the interval

Lemma 9.1. Let N be an odd integer and $x \in (0,N)$. Then $2x \pmod N$ is even if and only if $x \in (0,N/2)$.

Proof.

Assume $x \in (0,N/2)$. Then $2x \in (0,N)$ is even, and $2x = 2x \pmod N$.

Assume $x \in (N/2, N)$. Then $2x \in (N,2N)$ is even, and $2x \pmod N = 2x - N$ is odd. □ □

Proof of Thm 9.1(1)

Clearly Bit Security implies All-or-nothing Security. It remains to prove the converse.

Assume we have an RSA ciphertext, and the problem is to decrypt it. We make use of an algorithm, Parity Oracle $\mathcal{PO}_N$, which solves problem (II) for RSA with modulus N and for any given ciphertext. Then we can construct a polynomial time algorithm, which selects ciphertexts, makes calls to $\mathcal{PO}_N$, and solves problem (I), that is, tells the parity of the decryption of the ciphertext. Such an algorithm can be constructed as an iterative algorithm, which iterates a notion called current interval denoted by CI, such that, at each iteration the length of CI is halved, and the plaintext message remains in CI. With the help of the Lemma 9.1, Parity Oracle $\mathcal{PO}_N$ is used to select which of the two halves of the current CI is taken to the next iteration round.

Proof of Thm 9.1 (2)

It is given $c = m^e \bmod N$, and we know that $m \in (0,N)$.

Iteration 1: Set $CI = (0,N)$. Send ciphertext $2^e c = (2m)^e \bmod N$ to $\mathcal{PO}_N$. We get the parity of $2m \bmod N$. Then using Lemma 9.1, we know if $m \in (0,N/2)$ or $m \in (N/2, N)$. Set CI to be the interval where m lies.

Iteration 2: The case $CI = (0,N/2)$ is similar to the case $CI = (N/2, N)$. Let us consider the latter. Send ciphertext $4^e c = (4m)^e \bmod N$ to $\mathcal{PO}_N$. Then we get the parity of $x = 4m \bmod N$, and we can tell if

(1) $2m \bmod N \in (0,N/2)$, that is, $m \in (0,N/4)$ or $m \in (N/2,N/4+N/2)$; or

(2) $2m \bmod N \in (N/2,N)$, that is, $m \in (N/4, N/2)$ or $m \in (N/4+N/2,N)$.

Recall that $m \in (N/2, N)$. It follows that, in both (1) and (2), only the latter interval is possible. Hence, from the parity information given by the oracle, we get if $m \in (N/2,N/4+N/2)$ or $m \in (N/4+N/2,N)$, thus halving the CI again.

Proof of Thm 9.1 (3)

We get the following general rule to update CI :

If $\mathcal{PO}_N$ replies 0, then the new CI is the lower half of the current CI.

If $\mathcal{PO}_N$ replies 1, then the new CI is the upper half of the current CI.

Clearly after $\lfloor \log 2\ N \rfloor + 1$ steps the length of CI is less than 1. The last CI containing an integer gives this integer as m.

□

Conclusion: The owner of the RSA private key should not act as ”halving” or ”parity” oracle, since such partial information can be used to decrypt any given ciphertext.

The Rabin Bit

A similar $\mathcal{PO}_N$ works. For example, if N is such that (2/N) =1 (e.g., N = pq, p = q (mod 8)), if it outputs the parity of the smaller square root of those with Jacobi symbol = 1.

Application of Rabin bits:

Blum-Blum-Shub Pseudo-random Bit Generator

x_0 seed, outputs the least significant bit of the following integers:

$x_1 = x_0^2 \pmod N$, …, $x_i = x_{i-1}^2 \pmod N$, …

The ElGamal Bit

Given a ciphertext (c_1,c_2) of an unknown message m, a binary search can be used, exactly the same way as for RSA, by quering ciphertexts of the form

$$(c_1,2^i c_2) \pmod p,\ \ \mathrm{i} = 1,2,\ldots,\lfloor \log_2 p \rfloor + 1$$

This are the encryptions of messages

$$2^i m \pmod p,\ \ \mathrm{i} = 1,2,\ldots,\lfloor \log_2 p \rfloor + 1.$$

The DL Bit

Assume that the order q of the generator $g \in G$ is known and odd (as q typically is a prime). Then we can compute the square root of $h \in G$ by raising it to the power $(q+1)/2$. Given a parity oracle $\mathcal{PO}$, which for a given $h \in G$, replies with the parity of x such that $h = g^x$, we can reverse the square-and-multiply algorithm. Denote

$$x = x_0 + x_1 2 + x_2 2^2 + \ldots + x_{k-1} 2^{k-1}, \text{ where } k = \lceil \log_2 q \rceil .$$

Given h the $\mathcal{PO}$ replies with x_0. At the next round element

$h_1 = (h\, g^{-x_0})^{(q+1)/2} = g^y$, where $y = x_1 + x_2 2 + \ldots + x_{k-1} 2^{k-2}$

is given to $\mathcal{PO}$ and it replies with x_1.

In this manner, by quering the oracle $\mathcal{PO}$ k times all bits of x can be found.