Cryptography and Security

Cunsheng Ding

HKUST, Hong Kong

Version 3

Lecture 08: The RSA & ElGamal Public-Key Cipher

Objectives of this Lecture

1. To introduce the RSA and ElGamal public-key ciphers.
2. To look at their security issues.

- The RSA public-key cipher was invented in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT.
- The ElGamal public-key cipher was described by Taher ElGamal in 1985.

The RSA Public-Key Cipher

Euler's Totient Function $\phi(n)$

$\phi(n)$: The number of positive integers less than n that is relative prime to n.

Example: $\phi(7) = 6$ because

$$\{x : 1 \le x < 7, \gcd(x, 7) = 1\} = \{1, 2, 3, 4, 5, 6\}.$$

Example: $\phi(6) = 2$ because

$$\{x : 1 \le x < 6, \gcd(x, 6) = 1\} = \{1, 5\}.$$

Question: What is $\phi(8)$?

Formula for Euler's Totient Function ϕ

Theorem:

- $\phi(p) = p - 1$ for any prime number p.
- $\phi(pq) = (p-1)(q-1)$ for any two distinct primes p and q.

Exercise: Give a direct proof for the two claims using the definition of $\phi(n)$.

Assignment: Work out a formula for $\phi(n)$ in terms of the canonical factorization of $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$, where these p_i are pairwise distinct and t is a positive integer.

The RSA Public-key Cipher

Plaintext space: $\mathcal{M} = \{0,1\}^*$.

Ciphertext space: $\mathcal{C} = \{0,1\}^*$.

Binary representation and integers:

A binary block $M = m_0 m_1 \cdots m_{k-1}$ is identified with integer

$$m_0 + m_1 2 + m_2 2^2 + \cdots + m_{k-1} 2^{k-1}$$

which is in $\{0, 1, \cdots, 2^k - 1\}$.

The RSA Public-key Cipher

Choose two distinct primes p and q. Define $n = pq$.

Select d: $1 \leq d < \phi(n)$ with $\gcd(d, \phi(n)) = \gcd(d, (p-1)(q-1)) = 1$.

Compute e: e is the multiplicative inverse of d modulo $\phi(n)$.

Public key: (e, n)

Private key: d

Public-key space: $\mathcal{K}_e = \{1 \leq i < \phi(n) : \gcd(i, \phi(n)) = 1\} \times \{n\}$

Private-key space: $\mathcal{K}_d = \{1 \leq i < \phi(n) : \gcd(i, \phi(n)) = 1\}$.

Remark: The relation between the public key and private key is clear.

The RSA Public-key Cipher

Let $2^k < n < 2^{k+1}$, i.e., $k = \lfloor \log_2 n \rfloor$. Plaintext is broken into blocks of length k.

Encryption: For each block M, $C = M^e \bmod n$.

Decryption: $M = C^d \bmod n$.

Remark: Each message block M, when viewed as an integer, is at most $2^k - 1 < n - 1$.

Exercise: Prove the correctness of the decryption process above.

The Parameters of the RSA Public-key Cipher

Parameters:

p	q	n	ϕ	e	d

Public key: (e, n)

Private key: d

Other parameters: p, q, $\phi(n)$ must be kept secret.

Question: Why?

The Security of the RSA Public-key Cipher

Brute force attack: Trying all possible private keys.

The number of decryption keys:

$$|\{1 \leq d < \phi(n) | \gcd(d, \phi(n)) = 1\}| = \phi(\phi(n)) = \phi((p-1)(q-1)).$$

Comment: As long as p and q are large enough, this attack does not work as $\phi((p-1)(q-1)) - 1$ will be large! But the larger the n, the slower the system.

Attacking the RSA Using Mathematical Structures

Attack: Factor n into pq. Thus $\phi(n)$ and d are known.

Attack: Determine $\phi(n)$ directly, without first determining p and q.

Attack: Determine d directly, without first determining $\phi(n)$.

Attacking the RSA Using Mathematical Structures

Comment: It is believed that determining $\phi(n)$ given n is equivalent to factoring n.

Comment: With presently known algorithms, determining d given e and n, **appears** to be at least as time-consuming as the factoring problem.

Claim: We may use factoring as the benchmark for security evaluation.

RSA Security: Factoring

Security of RSA with respect to factoring depends on:

(1) the development of algorithms for factorization; and

(2) the advance in computing power.

Comment: A number of algorithms for factorization. Most of them involve too much number theory and cannot be introduced here. See https://en.wikipedia.org/wiki/Factorization

Comment: The computing power increases dramatically each year due to advances in hardware technology.

Estimation: If the RSA modulus n has about 2024 bits, the factorisation of n is computationally infeasible.

Security of the RSA Public-Key Cipher

Question: Does the RSA public-key cipher satisfy Conditions C1 and C2 specified in the previous lecture?

Answer: People believe that the answer is positive due to the difficulty of the integer factorisation problem. But no one has proved this belief.

How to Choose p and q

- They should be both random primes, not primes of special form, say for example, $2^k - 1$ or $2^k + 1$. It may be easier to factor n if so. Why?
- They should not be too close to each other. Why?
- They should not be too far away, in particular, they should differ in length by only a few digits. Why?
- Both $(p-1)$ and $(q-1)$ should contain a large prime factor. Why?
- $\gcd(p-1, q-1)$ should be small. Why?

Suggestion: If you wish to learn more, try to work out these problems.

How to Choose e and d

In theory, e and d could be any integer between 1 and $\phi(n)$ and relative to $\phi(n)$. However,

- d and e should not be too small. Why?

Suggestion: If you wish to learn more, try to work out this problem.

Further Comments on RSA

Not all public-key ciphers can be used for signing digital documents in the way described in Lecture 7. However,

- RSA can be used for signing digital documents in this way.

Question: Why RSA can be used for signing digital documents in this way?

The ElGamal Public-Key Cipher

The Discrete Logarithm Problem

The discrete logarithm problem: Let p be a prime, and let α be a primitive root of p. The *discrete logarithm problem* is to find $\log_\alpha a$ for any $1 \le a \le p-1$, which is defined to be the unique integer $0 \le i \le p-2$ such that

$$a = \alpha^i \bmod p.$$

Comment: No polynomial-time algorithm is known for this problem (except for certain special primes p). See

`https://en.wikipedia.org/wiki/Discrete_logarithm`

Comment: If p has 160 or more digits, the DLP is believed to be computationally infeasible to solve in general.

System Parameters of the ElGamal Cipher

Choosing system parameters:

- Choose p to be a large prime, and
- choose α to be a primitive root of p.

Note that both p and α are in the public domain and public parameters.

Key Pairs for the ElGamal Public-Key Cipher

User's key pair:

- Each user chooses a secret number u in $\mathbf{Z}_{p-1}$, as his/her private key $k_d := u$.
- The corresponding public key $k_e = (p, \alpha, \beta)$, where $\beta = \alpha^u \bmod p$.

The relation between the public key and the private key is very clear.

The Four Spaces of the ElGamal Public-Key Cipher

- $\mathcal{M} = \mathbf{Z}_p^* = \{1, \cdots, p-1\}$
- $\mathcal{C} = \mathbf{Z}_p^* \times \mathbf{Z}_p^*$
- $\mathcal{K}_e = \{p\} \times \{\alpha\} \times \mathbf{Z}_p^*$. So $|\mathcal{K}_e| = p - 1$.

  The public key $k_e = (p, \alpha, \beta)$.
- $\mathcal{K}_d = \mathbf{Z}_{p-1}$. Thus $|\mathcal{K}_d| = p - 1$.

  The private key $k_d = u$ such that $\beta = \alpha^u \bmod p$.

The Encryption and Decryption Functions

Encryption: For any public key $k_e = (p, \alpha, \beta)$, and for a (secret) random number $v \in \mathbf{Z}_{p-1}$,

$$E_{k_e}(x, v) = (y_1, y_2),$$

where

$$y_1 = \alpha^v \bmod p, \quad y_2 = x\beta^v \bmod p.$$

Decryption: For any $(y_1, y_2) \in \mathbf{Z}_p^* \times \mathbf{Z}_p^*$,

$$D_{k_d}(y_1, y_2) = y_2 \left(y_1^{k_d}\right)^{-1} \bmod p.$$

Exercise: Prove the correctness of the decryption process above.

Some Features of the ElGamal Public-Key Cipher

- Encryption has data expansion. This is good for security, but bad for cost and performance.
- For decryption, the receiver need not know the secret number v!
- The system is not **deterministic**, since the ciphertext depends on both the plaintext x and the random number v chosen by Alice, the sender. Hence, the encryption is **probabilistic**.
- The ElGamal public-key cipher cannot be used in the digital signature scheme covered in Lecture 7, as the domain and range of the function E_{k_e} are not the same. In fact, they are: $\mathbf{Z}_p^*$ and $\mathbf{Z}_p^* \times \mathbf{Z}_p^*$.
- But it can be used in the key distribution protocol covered in Lecture 7.

Weak Keys in the ElGamal Public-Key Cipher

The following two pairs of keys are weak (in fact, cannot be used):

- $k_e = (p, \alpha, \alpha)$, $k_d = u = 1$.

  Once k_e is published, k_d is easily seen to be 1.

- $k_e = (p, \alpha, 1)$, $k_d = u = 0$.

  Once k_e is published, k_d is easily seen to be 0.

Here we have seen specific examples of weak keys!

Security of the ElGamal Public-Key Cipher

Question: Is it computationally feasible to derive the private key k_d from the public key k_e?

Solution: Note that $k_e = (p, \alpha, \beta)$, where

$$\beta = \alpha^u \bmod p = \alpha^{k_d} \bmod p.$$

It depends on whether there is an efficient algorithm for solving the discrete logarithm problem.

It is believed that there is no polynomial-time algorithm for the DLP in general. So if p is large enough, say with 160 digits, and is not in certain special forms, it is computationally infeasible to derive k_d from k_e.

Security of the ElGamal Public-Key Cipher

Question: Given a ciphertext (y_1, y_2), is it computationally feasible to derive its corresponding plaintext x?

Attack 1: One way is to use $x = y_2\beta^{-v} \bmod p$, where $v \in \mathbf{Z}_{p-1}$ and β is publicly known. Since v is a secret random number, this does not work if p is large enough.

Attack 2: The second way is to use

$$x = D_{k_d}(y_1, y_2) = y_2 \left(y_1^{k_d}\right)^{-1} \bmod p.$$

This does not work either as it is hard to determine k_d.

Answer: It is believed that the answer to this question above in general is NO.

Security of the ElGamal Public-Key Cipher

Summary: Based on the arguments in the previous pages, people believe that the ElGamal public-key cipher satisfies Conditions C1 and C2. But there is no rigorous prof of this belief.