

Lecture 8: RSA Algorithm

Yuan Xue

As we have mentioned in the last lecture, Diffie and Hellman introduced a new approach to cryptography, and challenged cryptologists to design a general-purpose encryption algorithm that satisfies the public-key encryption requirements. One of the first responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, Len Adleman at MIT. Since then, the Rivest-Shamir-Adleman (RSA) scheme has become the most widely accepted and implemented general-purpose approach to public-key encryption¹. In this lecture, we will study the RSA algorithm. In particular, we shall examine the following topics:

- 1) Mathematical preliminaries of RSA;
- 2) RSA algorithm description;
- 3) Computational aspects of RSA;
- 4) Threats to RSA;
- 5) Public-Key Cryptography Standards (PKCS).

Among these topics, we will focus on the following – how RSA operates, why it would work, and why it is secure. Students are encouraged to read the textbook [WS] Section 9.2 to understand the computational aspect and the security of RSA. Additional reading materials on threats to RSA, and PKCS are provided at the discussion board in the blackboard system.

I. MATHEMATICAL PRELIMINARIES

In this section, we will introduce the mathematical background that helps to understand RSA.

A. Modular Addition

Let's start with one of the simplest ciphers: general Caesar cipher. Its encryption and decryption operation can be represented using the following mathematical functions.

$$C = (P + K) \bmod 26 \quad (1)$$

$$P = (C - K) \bmod 26 \quad (2)$$

¹Recently, a competing system has begun to challenge RSA: elliptic curve cryptography (ECC), which offers equal security for a far smaller key size, thereby reducing processing overhead.

P \ K	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

TABLE I
ADDITION MODULO 10

For simplicity, we replace 26 with 10, and show the general Caesar cipher, which is also the modular addition operation, in Table I. Note that the decryption operation, which subtracts the secret key K from ciphertext C modulo 10, can also be done by adding K' , which is the additive inverse of K modulo 10. An additive modular inverse of K is the number which is added to K to get 0 after modular operation. For example, 4's inverse (modulo 10) is 6, because $(4 + 6) \bmod 10 = 0$. If the secret key were 4, then to encrypt in general Caesar cipher, 4 is added to the plaintext; and to decrypt, 6 is added to the ciphertext. Formally, we have

$$C = (P + K) \bmod 26 \quad (3)$$

$$P = (C + K') \bmod 26 \quad (4)$$

where

$$K + K' \bmod 10 = 0. \quad (5)$$

B. Modular Multiplication

Now let's look at the mod 10 multiplication operation as shown in Table II. We note that only when $K = 1, 3, 7, 9$, the modular multiplication operation works as a cipher, because it only performs a one-to-one mapping between the plaintext and the ciphertext in these cases. What is special about the numbers $\{1, 3, 7, 9\}$? The answer is that those numbers are all relatively prime to 10. Generally, a number K is relatively prime to n means

P \ K	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

TABLE II
MULTIPLICATION MODULO 10

$$\gcd(K, n) = 1; 1 \leq K < n \quad (6)$$

where \gcd denotes the greatest common divisor.

For decryption, we can look for multiplicative inverse, and undo the multiplication by multiplying the ciphertext by the multiplicative inverse of the key. Multiplicative inverse of K , denoted by K^{-1} , is the number by which you'd multiply K to get 1 in mod n . Formally, the cryptosystem can be represented as follows.

$$C = (P \cdot K) \bmod n \quad (7)$$

$$P = (C \cdot K^{-1}) \bmod n \quad (8)$$

where

$$K \cdot K^{-1} \bmod n = 1 \quad (9)$$

Note that only those numbers that are relatively prime to n have multiplicative inverses in mod n . It is non-trivial to find a multiplicative inverse in mod n arithmetic, especially when n is very large. But there is an algorithm, known as Euclid's algorithm, which can efficiently find the inverse².

At this point, we observe that the modular multiplication can also be used as a cipher, if the value of K is chosen appropriately. Moreover, K and K^{-1} can be used as a pair of keys for encryption and decryption, which is required by public-key encryption model. The problem is, however, there exists an

²For details, please refer to http://en.wikipedia.org/wiki/Euclidean_algorithm, or step II in the hint.pdf file for homework 1.

algorithm (Euclid's algorithm) to calculate K^{-1} based on K , while in public-key encryption model, the private key can not be derived from knowledge of the public key.

So let's further explore other mathematical functions. Before that, we'd examine the question how many numbers less than n are relatively prime to n ? This number is denoted as $\phi(n)$, and called *totient function*. As we will see later, this number is quite important in the design of RSA. It is obvious that,

- when n is a prime, $\phi(n) = n - 1$;
- when n is the product of two distinct primes p, q , (i.e., $n = p \cdot q$, $p \neq q$ are primes), $\phi(n) = (p - 1)(q - 1)$.

C. Modular Exponentiation

Now let's proceed to consider encryption and decryption using modular exponentiation operation.

$$C = (P^K) \bmod n \quad (10)$$

$$P = (C^{K''}) \bmod n \quad (11)$$

where K'' is the exponentiative inverse of K .

Just like multiplicative inverse, we may ask what kind of values of K has the exponentiative inverse? and how its inverse can be calculated? The answers to these questions lead to the design of RSA. In what follows, we give a description of RSA algorithm first, then discuss how it is related with modular exponentiation.

II. RSA DESCRIPTION

The RSA scheme is a block cipher. Each plaintext block is an integer between 0 and $n - 1$ for some n , which leads to a block size $\leq \log_2(n)$. The typical block size for RSA is 1024 bits. The details of the RSA algorithm are described as follows.

• Key generation

- 1) Pick two large prime numbers p and q , $p \neq q$;
- 2) Calculate $n = p \times q$;
- 3) Calculate $\phi(n) = (p - 1)(q - 1)$;
- 4) Pick e , so that $\gcd(e, \phi(n)) = 1$, $1 < e < \phi(n)$;
- 5) Calculate d , so that $d \cdot e \bmod \phi(n) = 1$, i.e., d is the multiplicative inverse of e in mod $\phi(n)$;

6) Get public key as $K_U = \{e, n\}$;

7) Get private key as $K_R = \{d, n\}$.

- **Encryption**

For plaintext block $P < n$, its ciphertext $C = P^e \bmod n$.

- **Decryption**

For ciphertext block C , its plaintext is $P = C^d \bmod n$.

A. Why RSA works

As we have seen from the RSA design, RSA algorithm uses modular exponentiation operation. For $n = p \cdot q$, e which is relatively prime to $\phi(n)$, has exponential inverse in mod n . Its exponential inverse d can be calculated as the multiplicative inverse of e in mod $\phi(n)$. The reason is illustrated as follows.

Based on Euler's theorem, for y which satisfies $y \bmod \phi(n) = 1$, the following equation holds.

$$x^y \bmod n = x \bmod n \tag{12}$$

As $d \cdot e \bmod \phi(n) = 1$, we have that $P^{ed} \equiv P \bmod n$. So the correctness of RSA cryptosystem is shown as follows.

- **Encryption:** $C = P^e \bmod n$;

- **Decryption:** $P = C^d \bmod n = (P^e)^d \bmod n = P^{ed} \bmod n = P \bmod n = P$.

B. Why RSA is secure

The premise behind RSA's security is the assumption that factoring a big number (n into p , and q) is hard. And thus it is difficult to determine $\phi(n)$. Without the knowledge of $\phi(n)$, it would be hard to derive d based on the knowledge of e .

However factoring n is not the only way to break RSA. Students are encouraged to read the suggested material to find out more threats to RSA.