RSA Cryptosystem and Methods of Mathematical Analysis

Kaitaku Takeda

June 16, 2017

Abstract

This paper puts emphasis on RSA Cryptosystem and it's general algorithm along with a simple example. We will then shift our focus to integer factorization methods such as Number Field Sieve and Quadratic Sieve. The paper puts its focus on a particular method know as the Quadratic Sieve algorithm.

1 RSA Cryptosystem

1.1 Introduction

RSA was invented by Ronald Rivest, Adi Shamir, and Lenoard Adleman in 1977 and is known to be the most popular *asymmetric cryptography* used today. It is widely used for the purpose of encryption of small data, digital signatures, and secure exchange of *symmetric* keys.

1.2 Elementary Number Theory

In order to understand RSA, we must first familiarize ourselves with some number theory and modular arithmetics.

Definition 1.1. The *greatest common divisor* of two integers n and m, commonly denoted by $GCD(n, m)$, is the largest positive integer that divides both n and m.

Definition 1.2. Let $a, r, m \in \mathbb{Z}$ and $m > 0$. Then we write,

$$a \equiv r \bmod m$$

if and only if $m \mid a - r$, where m is the *modulus* and r is the *remainder*.

Definition 1.3. The set $\mathbb{Z}_m = \{0, 1, 2, ..., m-1\}$ forms an *integer ring* in which addition and multiplication is *closed*.

Definition 1.4. The number of integers in $\mathbb{Z}_m$ relatively prime to m, denoted $\varphi(m)$ is called *Euler's Phi Function*.

Theorem 1.1. A *multiplicative inverse*, a^{-1} for $a \in \mathbb{Z}_m$ such that,

$$a \cdot a^{-1} \equiv 1 \bmod m$$

exists if and only if $GCD(a, m) = 1$.

Theorem 1.2. Let $m = p_1^{e_1} \cdot p_2^{e_2} \cdot ... \cdot p_n^{e_n}$ where p_i are distinct primes and e_i are positive integers. Then,

$$\varphi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})$$

Now that we laid out some of the fundamental number theory concepts, we are ready to dive into the RSA Cryptosystem.

1.3 Key Generation

Unlike *symmetric cryptography*, RSA utilizes a private key and a public key. As the name suggests, public key is made public and the private key is kept secret. Key generation in RSA is accomplished in 5 steps:

1. Choose 2 large prime numbers p and q
2. Compute $n = p \cdot q$
3. Compute $\varphi(n) = (p - 1)(q - 1)$
4. Arbitrarily select a public key $e \in \{1, 2, ..., \varphi(n) - 1\}$ such that

$$GCD(e, \varphi(n)) = 1$$

5. Compute the private key d such that

$$d \cdot e \equiv \bmod \varphi(n)$$

We have now generated the two necessary keys:

$$k_{pvt} = d$$

$$k_{pub} = (e, n)$$

1.4 Encryption and Decryption

Again, we make k_{pub} public and keep k_{pvt} to yourself. Now suppose your friend wants to send you some data. Given a *plaintext* data x and the public key $k_{pub} = (e, n)$, one can obtain a *ciphertext* y with the following equation:

$$encryption : y = x^e \bmod n$$

The *ciphertext* y can now be sent to you over an insecure channel (e.g. the internet) without letting anyone view the original data x. Once you receive the *ciphertext* y, you can decryt it using your private key, $k_{pvt} = d$:

$$decryption : x = y^d \bmod n$$

Let's demonstrate how the RSA algorithm works with a simple example.

Example 1.1. Suppose your friend wants to send you a message $x = 4$. We must first generate our keys using the five steps explained above. For the sake of this example, we will choose $p = 3$ and $q = 11$ (Note: These numbers must be randomly chosen and must be very large. Today p and q should at least be 512 bits in order to preserve the security of this system). Then our

$$n = 3 \cdot 11 = 33$$

and,

$$\varphi(33) = (3 - 1)(11 - 1) = 20$$

For this example, we will choose $e = 3$, however e should practically be a fairly large number as well. We can check that $GCD(3, 20) = 1$ by using the *Euclidean Algorithm*:

$$20 = 6(3) + 2$$

$$3 = 1(2) + 1$$

$$2 = 2(1) + 0$$

We will use the *Extended Euclidean Algorithm* to find our private key d such that $d \cdot 3 \equiv 1 \bmod 20$. Using the results of the above equations,

$$1 = 3 - 1(2)$$

$$1 = 3 - (20 - 6(3))$$

$$1 = 7(3) - 1(20)$$

Therefore we have,

$$7(3) - 1 = 1(20)$$

$$7 \cdot 3 \equiv 1 \bmod 20$$

Thus, we found our private key $d = 7$. We now have our public key $k_{pub} = (3, 33)$ and our private key $k_{pvt} = 7$. Now your friend can encrypt their message $x = 4$ using the public key:

$$y = 4^3 \bmod 33$$

$$\equiv 64 \bmod 33$$

$$\equiv 31 \bmod 33$$

and you can decrypt the *ciphertext* $y = 31$ using your private key to read the original message $x = 4$:

$$x = 31^7 \bmod 33$$

$$\equiv (-2)^7 \bmod 33$$
$$\equiv -128 \bmod 33$$
$$\equiv 4 \bmod 33$$

We have successfully demonstrated the RSA Cryptosystem, however is it really secure? Remember that the only information that is not made public is your private key d and the two prime numbers p and q. Suppose a hacker wants to steal your private key. In order to do so, they must first compute $\varphi(n) = (p-1)(q-1)$, but they don't know p and q. Of course with our example, $n = 33$ would not take so long to factorize into $p = 3$ and $q = 11$. However, remark that practical RSA uses p and q that are larger than 512 bits which leaves n to be large as 1024 bits. Even with today's fastest computer, factorization of a such large number is believed to be infeasible.

Introduction of RSA suddenly brought attention to integer factorization. Many mathematicians engaged their selves in studying various integer factorization methods and implementing their own algorithms that can quickly factorize larger numbers. In the next section, we will discuss about an algorithm known as Quadratic Sieve that can efficiently factorize large numbers.

2 Quadratic Sieve