RSA Cryptosystem and Methods of Mathematical Analysis

Kaitaku Takeda

June 9, 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*.

Definition 1.5. Let p be an odd prime. An integer a is called *quadratic residue* mod$p$ if $a \equiv x^2 \bmod p$ for some $x \in \mathbb{Z}$. The *Legendre Symbol* $\left(\frac{a}{p}\right)$ is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & a \equiv x^2 \bmod p \\ -1 & a \not\equiv x^2 \bmod p \\ 0 & a \equiv 0 \bmod p \end{cases}$$

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 then 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 have 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 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 such large numbers.

2 Quadratic Sieve

2.1 Background

Before we begin our discussion on Quadratic Sieve, let's briefly look at the background of it's creation. Before the spark of RSA, even a 20-digit number was believed to be difficult to factorize. However by 1980, which was only 3 years after the introduction of RSA, 50-digit numbers were considered as unchallenging numbers to factorize. This was due to work of two mathematicians Brillhart and Morrison, and was known as the *continued fraction factoring algorithm*. In 1990, Carl Pomerance doubled the length of this number with the invention of Quadratic Sieve algorithm. Quadratic Sieve was the first algorithm known to factorize the famous 129-digit RSA number. This number had been estimated to take 40 quadrillion years to factorize in an article written by Martin Gardner in 1976.

2.2 Introduction to Integer Factorization

Despite the difference in the names of these algorithms, they are similar to some extent. In fact all algorithms branch off of one root algorithm invented by the great mathematician, Pierre de Fermat.

2.2.1 Fermat's Algorithm

Fermat used the fact that any odd integer can be written as a difference of two square.

Definition 2.1. (Fermat's Algorithm) Let $n = x^2 - y^2$ for some odd $n \in \mathbb{Z}$. The goal of the algorithm is to find solutions x and y to this equation. Proceed by rewriting the equation:

$$x^2 - n$$

Then take our initial trial x_1 to be the largest square bigger than n which equals $\lceil\sqrt{n}\rceil^2$. Now plug x_1 into the above equation.

$$(x_1)^2 - n$$

$$\lceil\sqrt{n}\rceil^2 - n$$

If the result of such difference is a square, then we have found our y such that $\lceil\sqrt{n}\rceil^2 - n = y^2$. Otherwise we must sequentially continue our trial with our x_i equal to the ith largest square bigger than n.

Example 2.1. Let $n = 2257$. n is clearly odd so proceed with initial trial:

$$x_1 = \lceil\sqrt{2257}\rceil = 48$$

$$48^2 - 2257 = 47$$

47 is not a square so continue the trial.

$$x_2 = \lceil\sqrt{2257}\rceil + 1 = 49$$

$$49^2 - 2257 = 144 = 12^2$$

We have found our x and y to be 49 and 12 respectively. Then,

$$2257 = 49^2 - 12^2$$

$$= (49 - 12)(49 + 12)$$

$$= (37)(61)$$

Thus we have successfully factorized 2257 into 37 and 61.

Although the algorithm is clever, it is still too slow. In fact in it's worst case, the *time complexity* of Fermat's algorithm is equal to that of standard trial division. However in the 1920s, Belgian mathematician Maurice Kraitchik brought improvement to Fermat's algorithm.

2.2.2 Kraitchik's Algorithm

Instead of finding solutions to $n = x^2 - y^2$, Kraitchik's idea was to find solutions to $x^2 \equiv y^2 \bmod n$. His results were then categorized into uninteresting solutions such that $x \equiv \pm y \bmod n$ and interesting solutions where $x \not\equiv \pm y \bmod n$. For solving RSA problems, it turns out that finding such interesting solutions can directly lead to finding prime factors of n using Kraitchik's algorithm.

Definition 2.2. (Kraitchik's Algorithm) Let $x^2 \equiv y^2 \bmod n$ such that $x \not\equiv \pm y \bmod n$. Then $GCD(x - y, n)$ and $GCD(x + y, n)$ will produce non-trivial factors of n. Begin the algorithm in similar fashion as Fermat's by computing the sequence of x_i and it's corresponding sequence of equations, call it $Q(x)$, where $Q(x_i) = x_i^2 - n$.

$$\begin{aligned} x_1 &= \lceil \sqrt{n} \rceil & Q(x_1) &= x_1^2 - n \\ x_2 &= x_1 + 1 & Q(x_2) &= x_2^2 - n \\ x_3 &= x_2 + 1 & Q(x_3) &= x_3^2 - n \\ &\vdots & &\vdots \end{aligned}$$

Notice that $x_i^2 \equiv x_i^2 - n \bmod n$. Therefore, we can write,

$$x_1^2 \cdot x_2^2 \cdot \ldots \cdot x_k^2 \equiv Q(x_1) \cdot Q(x_2) \cdot \ldots \cdot Q(x_k) \bmod n$$

Then if for some k, the product $\prod_{i=1}^{k} Q(x_i)$ is a square, call it y^2, and we let $x = \prod_{i=1}^{k} x_i$, we have our desired equation $x^2 \equiv y^2 \bmod n$. At this point, the only criteria left to check is $x \not\equiv \pm y \bmod n$. Then computing $GCD(x \pm y, n)$ will leave us with factors of n.

It is easy to see that this is far more efficient than Fermat's method since Fermat will still be searching if $Q(x_k)$ is not a square. So how do we find the right sequence of $Q(x)$ so that their product is a square?

2.3 Finding Kraitchik's Sequence

In the definition of Kraitchik's Algorithm, the method of finding the proper sequence was left obscure. However two mathematicians, John Brillhart and Michael Morrison discovered an efficient, yet an extremely simple method for achieving such sequence, using elementary linear algebra. Let us start with few more definitions.

Definition 2.3. An integer $m \in \mathbb{Z}^+$ is called *B-smooth* if none of its prime factors is greater than the given bound B.

Definition 2.4. A *factor base* is a relatively small set of distinct primes such that $p_i \leq B$ for all i. (Note: We will use a special case of factor base which include the element -1)

Now consider the sequence $Q(x)$ once again, where $Q(x_i) = x_i^2 - n$. Then write each $Q(x)$ as it's prime decomposition. That is $Q(x) = \prod p_i^{e_i}$. Now define the *exponent vector* of Q(x), denoted $v(Q(x))$, to be the vector $(e_1, e_2, ..., e_l)$ where e_j is the exponent of its prime decomposition for all $j \in \{1, ..., l\}$ and e_l is the exponent of the largest prime in the decomposition. What significance does the *exponent vector* have in our algorithm? Remark that our goal is to find the sequence of $Q(x)$ such that their product is a square. If we rephrase that statement with respect to its prime decomposition,

$$Q(x_1) \cdot Q(x_2) \cdot ... \cdot Q(x_k) = p_1^{e_1} \cdot p_2^{e_2} ... = y^2$$

Then all non-zero exponents of such decomposition must be even. That is equivalent to stating that $\sum_{i=1}^{k} v(Q(x_i))$ is equal to a vector, such that all entries are even.

Here is where we utilize a very fundamental concept of linear algebra. Since our focus is on the parity of the outcome vector, express each *exponent vector* in terms of mod2, denoted by $v_2(Q(x))$. Now construct a $l \times k$ matrix A such that its ith column is equal to $v_2(Q(x_i))$.

$$A = \begin{bmatrix} \vdots & \vdots & & \vdots \\ v_2(Q(x_1)) & v_2(Q(x_2)) & ... & v_2(Q(x_k)) \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Then in order to find our desired sequence $Q(x)$, finding a non-trivial solution to the equation $A\overrightarrow{w} \equiv 0 \bmod 2$ suffices (where $w_i \equiv 1$ corresponds to the $Q(x_i)$ of particular interest).

In Section 2.4, we will elaborate on the usage of *B-smooth* numbers and *factor base* to make the dimension of matrix A much smaller. In fact, this is where the meat of Quadratic Sieving resides.

2.4 Sieving

We now know that in order to factorize n we must first compute sequence $\{x\}$ and corresponding sequence $Q(x)$ such that the product of $Q(x)$ is a square. We also found out that such sequence can be computed by using minimal linear algebra. However notice that if $Q(x_i)$ decomposes into larger primes, dimension of A will become substantially large since the number of rows in A is determined by the lth prime factor. Thus, we sieve out appropriate $Q(x_i)$ by constructing a *factor base* with respect to a bound B. This method is also known as *Dixon's Algorithm.*

2.4.1 Choosing B

Choice of the bound B is a rather complex topic. In fact, cryptographers think the optimal value for B is approximately $\exp(\sqrt{\log n \log \log n})$. This paper will not elaborate further on the reasoning of this value, however B should be generally chosen to be small since it defines the bound for our matrix.

2.4.2 Constructing a Factor Base

There are two criteria that must be met when constructing a *factor base*, P.

$$1)\ \text{Choose bound } B \text{ such that } \forall p_i \in P : p_i \leq B$$

$$2)\ \left(\frac{n}{p}\right) = 1 \ (n \text{ is the number to be factorized})$$

The reasoning behind the second criteria is trivial.

$$\begin{aligned} \left(\frac{n}{p}\right) \neq 1 \quad &\Leftrightarrow \quad x^2 \not\equiv n \bmod p \\ &\Leftrightarrow \quad p \nmid x^2 - n \\ &\Leftrightarrow \quad p \nmid Q(x) \end{aligned}$$

Since we know that such p will never be in the prime decomposition of $Q(x)$, we do not include it in our *factor base*.

Remark that our *factor base* includes -1. The reason for this is to accommodate for negative values of $Q(x)$. We will shortly see that Pomerance set the interval of the sequence $Q(x)$ to be $[\lfloor\sqrt{n}\rfloor - M, \lfloor\sqrt{n}\rfloor + M]$, whereas Kraitchik used $[\lceil\sqrt{n}\rceil, \lceil\sqrt{n}\rceil + M]$ for some bound M. With the given construction of a *factor base*, we are now ready to sieve the most appropriate sequence of $Q(x)$.

2.4.3 Sieving Algorithm

Recall the sieving technique of *Eratosthenes* to find the prime numbers out of a ordered set of positive integers with a certain bound. The method circles p_i then crosses out every multiple of p_i in the set, with $p_1 = 2$. Then p_{i+1} is the smallest number that is left uncrossed, so we circle it and repeat the process. Those numbers that are circled at the end are your prime numbers within the bound. This ancient technique is what motivated Carl Pomerance's invention of Quadratic Sieve. After construction of a *factor base* P and the sequence $Q(x)$ in the interval $[\lfloor\sqrt{n}\rfloor - M, \lfloor\sqrt{n}\rfloor + M]$ for some M, he took $p_1 \in P$, and divided elements of $Q(x)$ that were multiple of p_1 or its power. Pomerance sequentially continued this for every $p_i \in P$. The sequence now has elements such that $Q(x_i) = 1$ (-1 for negative values) or elements that are not equal to 1 (-1). Surprisingly, those in the former form are the *B-smooth* numbers for some bound B. Once these numbers are sieved, we are only left with the task to put them in their *exponent vector* form, and spot those such that their product is a square.

Example 2.2. (Quadratic Sieve) Let $n = 87463$ and $M = 30$. Our first step is to compute the sequence $Q(x)$ with $x_1 = \lceil\sqrt{87463}\rceil = 296$ and $Q(x_1) = 296^2 - 87463 = 153$. After computing few more elements with respect to our bound, we get

x	Q(x)
265	-17238
266	-16707
⋮	⋮
296	153
⋮	⋮
324	17513
325	18162

Now we will construct our *factor base* P with $B = 37$ (Note: check to see that $37 \approx \exp(\sqrt{\log 87463 \log\log 87463})$). We can observe that not all primes $p_i \leq 37$ meet the criteria $\left(\frac{87463}{p_i}\right) = 1$.

p	2	3	5	7	11	13	17	19	23	29	31	37
$\left(\frac{n}{p}\right)$	1	1	-1	-1	-1	1	1	1	-1	1	-1	-1

Thus we compute our *factor base* to be $P = \{-1, 2, 3, 13, 17, 19, 29\}$. If we sieve using our definition of P, our original table will substantially shrink down to,

x	Q(x)
265	-17238
278	-10179
296	153
299	1938
307	6786
316	12393

We now write $Q(x)$ in its prime decomposition form,

$$Q(x_1) = -17238 = -1^1 \cdot 2^1 \cdot 3^1 \cdot 13^2 \cdot 17^1 \cdot 19^0 \cdot 29^0$$

$$Q(x_2) = -10179 = -1^1 \cdot 2^0 \cdot 3^3 \cdot 13^1 \cdot 17^0 \cdot 19^0 \cdot 29^1$$

$$Q(x_3) = 153 = -1^0 \cdot 2^0 \cdot 3^2 \cdot 13^0 \cdot 17^1 \cdot 19^0 \cdot 29^0$$

$$Q(x_4) = 1938 = -1^0 \cdot 2^1 \cdot 3^1 \cdot 13^0 \cdot 17^1 \cdot 19^1 \cdot 29^0$$

$$Q(x_5) = 6786 = -1^0 \cdot 2^1 \cdot 3^2 \cdot 13^1 \cdot 17^0 \cdot 19^0 \cdot 29^1$$

$$Q(x_6) = 12393 = -1^0 \cdot 2^0 \cdot 3^6 \cdot 13^0 \cdot 17^1 \cdot 19^0 \cdot 29^0$$

and insert their corresponding *exponent vectors* mod2 as columns of matrix A and solve for $A\overrightarrow{w} \equiv 0 \bmod 2$.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

After performing *Gaussian Elimination* on A, we find a non-trivial solution $\overrightarrow{w} = (1, 1, 1, 0, 1, 0)$. Therefore we can conclude that the product of $Q(x_1)$, $Q(x_2)$, $Q(x_3)$, and $Q(x_5)$ is a square. We now put such values into our original congruence.

$$(265 \cdot 278 \cdot 296 \cdot 307)^2 \equiv (-17238 \cdot -10179 \cdot 153 \cdot 6786) \bmod 87463$$

$$34757^2 \equiv 28052^2 \bmod 87463$$

So our $x = 34757$ and $y = 28052$. Then finally compute $GCD(6705, 87463)$ and $GCD(62809, 87463)$ to get prime factors 149 and 587.

As you can see, Quadratic Sieve extends off work of many mathematicians. Similarly, an extension of Quadratic Sieve, the Number Field Sieve was invented, which is the more commonly used algorithm today for larger integer factorization. However, the Quadratic Sieve is still the faster algorithm for numbers up to 110 digits long.

3 Conclusion

As mentioned earlier, Quadratic sieving is an extremely fast algorithm for factorizing certain numbers. In fact, its worst case running time is $O(\exp(\sqrt{\log(n)\log(\log(n))}))$ (where the big-O notation in defines the function for an upper bound). However despite its speed, Quadratic Sieve has become impractical for solving modern RSA challenge. With the rise of Number Field Sieve (NFS), the size of RSA numbers has grown as large as 2048 bits. Yet the largest number to be factorized with NSA is 768 bits, which was achieved by Arjen Lenstra and his colleagues. Due to this rapid growth in size, RSA itself is not typically used to encrypt the actual data anymore. Instead, data is usually encrypted using a *symmetric key*, and RSA plays a role in secure exchange of this key. Despite the rise in popularity of other *asymmetric cryptography* such as the *elliptical curve cryptosystem*, RSA still resides as the most commonly used cryptosystem.

References

[1] Arjen K. Lenstra. *Computational methods in public key cryptology*, Singapore University Press, 2002.

[2] Arjen K. Lenstra. *Integer factoring* (Vol. 19), Springer, Netherlands, (2000), 101-128.

[3] Arjen K. Lenstra, Mark S. Manasse. *Factoring with two large primes*, Springer, Berlin, Heidelberg, (1990), 72-82.

[4] Christof Paar, Jan Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*, Springer-Verlage, Berlin, Heidelberg, 2010.

[5] Carl Pomerance. *Analysis and Comparison of Some Integer Factoring Algorithms, in Computational Methods in Number Theory* (H.W. Lenstra, Jr. and R. Tijdeman, eds.), Math. Centre Tract 154, Amsterdam, 1982.

[6] Carl Pomerance. *Algorithmic Number Theory*, MSRI Publications, 44, (2008), 69-80.

[7] Carl Pomerance. *A Tale of Two Sieves*, `http://www.ams.org/notices/199612/pomerance.pdf`