Journal of Mathematical Extension
Vol. 14, No. 4, (2020), 1-20
ISSN: 1735-8299
URL: http://www.ijmex.com
Original Research Paper

A Fast and Secure RSA Public Key Cryptosystem

M. Mohammadi
Tehran Branch, Islamic Azad University

A. Zolghadrasli[*]
Shiraz University

M. A. Pourmina
Tehran Branch, Islamic Azad University

Abstract. RSA is a well-known public-key cryptosystem. It is the most commonly used and currently most important public-key algorithm which can be used for both encryption and signing. RSA cryptosystem involves exponentiation modulo an integer number n that is the product of two large primes p and q. The security of the system is based on the difficulty of factoring large integers in terms of its key size and the length of the modulus n in bits which is said to be the key size. In this paper, we present a method that increases the speed of RSA cryptosystem. Also, an efficient implementation of arithmetic and modular operations are used to increase its speed. The security is also enhanced by using a variable key size space. There exist numerous implementations (hardware or software) of RSA cryptosystem, but most of them are restricted in key size. An important improvement achieved in this paper is that the system is designed flexible in terms of key size according to user security.

AMS Subject Classification: 14G50
Keywords and Phrases: Ryptography, RSA cryptosystem, public key, private key, prime numbers, encryption, decryption, modular reduction

Received: April 2018; Accepted: March 2020
[*]Corresponding author

1. Introduction

Now a day's all works related to banking, ATM card, credit card, marketing, E-commerce, etc. are doing with the help of the internet. So providing security is a necessity for the network. For secure communication, many cryptography techniques are presented [1].

Cryptography is a process that is associated with scrambling plaintext (ordinary text, or clear text) into cipher text (a process called encryption), then back again to plain text (known as decryption) [2]. Cryptosystems can be divided into two types, secret-key cryptosystem, and public-key cryptosystem.

RSA algorithm is a cryptosystem, which is known as one of the first practicable public-key cryptosystems and is widely used for secure data transmission [3]. The important property of RSA algorithm is that the encryption key is public and differs from the decryption key which is kept secret. This property gives the RSA algorithm asymmetry property which is based on the practical difficulty of factoring the product of two large prime numbers which is known as the factoring problem. The main procedure of the RSA algorithm is that allows the user to create and then publish the product of two large prime numbers, along with an auxiliary value, as their public key. For more security, the prime factors must be kept secret. Any user can use the public key to encrypt a message (M), but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message [4]. Breaking RSA encryption is known as the RSA problem. It is an open question whether it is as hard as the factoring problem [5, 6, 7].

The efficient implementation of the long integer modular arithmetic primarily determines the performance of RSA. Grobschadl [8] presented the multiplier architecture of the RSA crypto chip for long integer modular arithmetic which utilized Barret's modular reduction method. Hasenplaugh et al. [9] proposed a modification to Barrett's algorithm that leads to a reduction in multiplications and additions. Cao et al. [10] put forth a lookup-table-based modular reduction method which partitions the binary string of an integer to be reduced into blocks according to its runs. Its complexity depends on the number of runs in the binary string.

Sharma et al. [11] proposed The Short Range Natural Number (SRNN) algorithm which was similar to RSA algorithm with some modification. In the algorithm, an extremely large number that has two prime factors (similar to RSA) is used. In addition, two natural numbers in pair of keys (public, private) are used. These natural numbers increase the security of the cryptosystem.

Patidar et al. [1] suggested an algorithm concept to present the modified form of RSA algorithm to speed up the implementation of RSA algorithm during data exchange across the network. This includes the architectural design and enhanced form of RSA algorithm through the use of the third prime number to make a modulus n which is not easily decomposable by intruders. A database system is used to store the key parameters of RSA cryptosystem before it starts the algorithm.

In this paper, we propose a new approach for the efficient implementation of RSA. A combination of Montgomery [12] and Barret [13] modular reduction techniques are used to get an efficient implementation of modular multiplication. Also applying a variable key size space improves security as well.

This paper is organized as follows. In Section 2, some of the elementary theorems in number theory related to public-key algorithms are considered. In Section 3, RSA Public-Key cryptosystem is introduced. In Section 4, we present an efficient implementation of RSA. Section 5 shows test results and comparison. A brief conclusion is presented in Section 6.

2. Some Mathematical Theorems Used in Public-Key Cryptosystem

2.1 Fermat's and Euler's theorems

Two theorems that play important roles in public-key cryptosystems are Fermat's and Euler's theorems. The proof of these theorems can be found in any book in number theory [12].

(I) Fermat's little theorem:
Let p be a prime and a be any integer. Then $a^p \equiv a \pmod{p}$, if $gcd(p, a) = 1$ then $a^{p-1} \equiv 1 \pmod{p}$.
Fermat's theorem can be constructed as saying that the congruence $x^{p-1} \equiv 1 \pmod{p}$ has exactly $p - 1$ solutions ($x = 1, 2, \ldots, p - 1$) which are distinct mod p. This theorem is mainly used as a primality test in the prime generation which has a crucial part in the cryptographic algorithm.

(II) Euler's theorem:
One of the important quantities in number theory referred to as Euler's totient function $\varphi(n)$, where $\varphi(n)$ is the number of positive integers less than n and relatively prime to n. Obviously for a prime p, $\varphi(p) = p - 1$.

Euler theorem states that for every a and n that are relatively prime $a^{\varphi(n)} \equiv 1$ mod n. Euler's totient function is used in RSA public-key cryptosystem in the key generation phase to determine the public and private exponent (see 3.1).

2.2 Chinese remainder theorem

Let $n = n_1 n_2 \ldots n_k$, where n_i are pair-wise relatively prime. Consider the correspondence

$$a \leftrightarrow (a_1 \, a_2 \ldots a_{2k}),$$

where $a \in Z_n^*$ [10], $a_i \in Z_{n_i}^*$, and

$$a_i = a \bmod n_i, \text{ for } i = 1, 2, \ldots, k.$$

Then the correspondence is a one-to-one mapping between Z_n and the Cartesian product: $Z_{n1} \times Z_{n2} \times \ldots \times Z_{nk}$. As a corollary of this theorem:

If $n_1, n_2, \ldots, n_k$ are pair-wise relatively prime and $n = n_1\, n_2\, \ldots\, n_k$, then for all integers x and a,

$$x \equiv a \pmod{n_i},$$

for $i = 1, 2, \ldots, k$ if and only if

$$x \equiv a \pmod{n}.$$

3. RSA Public-Key Cryptosystem

RSA is the most widely used and currently most important public-key algorithm. Its encryption procedure involves exponentiation which could be a very time-consuming process.

3.1 RSA Key generation algorithm

In the RSA system for each entity, the key pair is generated as follows:

1. Two large primes p and q of the same length are selected randomly.
2. The modulus of the system $n = p.q$ and $\varphi(n) = (p-1).(q-1)$ are calculated.
3. An arbitrary integer e is selected such that $gcd(e, \varphi(n)) = 1$ and $1 < e < \varphi(n)$. (e is usually small to produce a large size d for better security) [12].
4. The integer d is calculated satisfying $1 < d < \varphi$ and $e.d \equiv 1$ (mod $\varphi(n)$).

The public keys (e, n) are published and the private key d is kept secret. The integers e and d in this algorithm are called the *encryption exponent* and the *decryption exponent*, respectively, while n is called the *modulus*.

3.2 RSA Encryption/decryption scheme

(I) Encryption:
To encrypt a message M, the sender calculates $C = M^e \pmod{n}$ using the public encryption key (e, n) of the receiver.

(II) Decryption:
To decrypt the message C, the receiver calculates $M = C^d \pmod{n}$ using his/her own private decryption key.

The decryption is correct because:
$e.d \equiv 1 (\bmod\, \varphi)$, there exist an integer k such that $e.d = 1 + k\,\varphi$ and $gcd(m, p) = 1$ then by Fermat's theorem: $m^{(p-1)} \equiv 1 (\bmod\, p)$.

Raising both sides of the congruence to the power $k\,(q-1)$ and then multiplying both sides by m yields :

$$m^{k\,(p-1)\,(q-1)+1} \equiv m (\bmod\, p).$$

On the other hand, if $gcd(m, p) = p$ then this last congruence is valid. Hence in all cases:

$$m^{ed} \equiv m (\bmod\ p).$$

By the same argument:

$$m^{ed} \equiv m (\bmod\ q).$$

Finally, since p and q are distinct primes, it follows that:

$$m^{ed} \equiv m (\bmod\, n),$$
$$c^d \equiv m^{ed} \equiv m (\bmod\, n).$$

The system relies on the well-known number-theoretic identities such as Euler's, Fermat's and Chinese remainder theorem.

4. Efficient Implementation of RSA

RSA cryptosystem is simply modular exponentiation. The modulus n is the product of two large primes p and q. The security of the system is based on the difficulty of factoring large integers in terms of its key size and the length of the modulus n in bits which is said to be the key size.

4.1 Modular reduction

The main operation of many public-key cryptosystems (PKC) such as RSA is the use of the large integers modular exponentiation which is implemented by repeating modular multiplication [14]. For this reason, the efficiency of many PKCs is determined by the efficiency of the modular multiplication algorithm [15, 16, 17, 18].

In this study, several suggested techniques have been considered to minimize the time for modular multiplication and division, either by reducing the number of multiplication or decreasing the time to multiply two numbers and taking the modular reduction.

One method is to pre-compute a multiplication table for one of the operands and use it to perform the multiplication with a series of additions and table lookup steps. Another one is to perform the modular reduction in parallel with each stage of the multiplication calculation by removing the most significant digit and adding its reminder to the partial sum. Alfred J. Menezes et.al [12] also have a huge discussion on different exponentiation algorithms categorized in terms of having fixed or arbitrary choices of the base or the exponent.

One of the efficient algorithms for modular multiplication is the Montgomery modular multiplication algorithm because it avoids division by the modulus [19, 20, 21, 22].

4.1.1. Montgomery reduction technique

Montgomery reduction is a technique that allows efficient implementation of modular multiplication without explicitly performing the classical modular reduction step. Let $m > 0$, R, T be integers such that: $R > m$, R and m are relatively prime ($gcd(m, R) = 1$) and $0 \leqslant T < mR$, this method computes $TR^{-1} \bmod m$ (which is called a Montgomery reduction of T modulo m concerning to R) without using classical method.

It should be mentioned that $\left(T + \left(-T\overset{-1}{m} \bmod R\right) m\right) / R$ is an integer, because:

$$\begin{aligned}
(-T\overset{-1}{m}\bmod R) &= T(-\overset{-1}{m}\bmod R) + kR,\\
m(-\overset{-1}{m}\bmod R) &= -1 + jR,\\
\left(T + (-T\overset{-1}{m}\bmod R)\, m\right)/R &= (T + (T\ (-\overset{-1}{m}\bmod R) + k\,R)\, m)/R\\
&= T((1 + -1 + jR) + k\,R m)/R\\
&= (Tj + k\,m)R/R = Tj + k\,m
\end{aligned}$$

Also we can write:

$$\left(T + (-T\overset{-1}{m}\bmod R)\, m\right)/R = TR^{-1} \bmod m$$

This is correct, because:

$$\begin{aligned}
T + (-T\overset{-1}{m}\bmod R)m &= T \bmod m,\\
\left(T + (-T\overset{-1}{m}\bmod R)\, m\right)/R \bmod m &= TR^{-1}\bmod m
\end{aligned}$$

Besides:

$T < mR$, $(-T\overset{-1}{m}\bmod R) < R$, then:

$$\left(T + (-T\overset{-1}{m}\bmod R)\, m\right)/R < (mR + mR)/R < 2m$$

$$U = (-T\overset{-1}{m}\bmod R)\ \ (R = b^n, \bmod R = \text{low order bits})$$

So, if all integers are represented in radix b with selecting $R = b^n$ ($n - 1$ is the length of m), $TR^{-1} \bmod m$ can be computed with two multiple-precision multiplications (i.e., $U = Tm'$ where m' is minus modular inverse of m) and simple right shift of $(T + Um)$ is required instead of division by R.

4.1.2. Barret's modular reduction method In modular arithmetic, Barrett reduction is a reduction algorithm introduced in 1987 by P.D. Barrett [13]. A naive way of computing Z mod N would be to use a fast division algorithm. Barrett reduction is an algorithm designed to optimize this operation assuming N is constant and $Z < N^2$, replacing divisions by multiplications. Let $1/N$ be the inverse of N as a floating-point number. Then $Z \bmod N = Z - \lfloor \frac{Z}{N} \rfloor N = Z - qN$, $q = \lfloor \frac{Z}{N} \rfloor$. Where $\lfloor x \rfloor$ denotes the floor function. The result is exact, as long as $1/N$ is computed with sufficient accuracy.

However, division by N can be expensive and, in cryptographic settings, may not be a constant time instruction on some CPUs. Thus Barrett reduction approximates $1/N$ with value $/2^k$, because division by 2^k is just a right-shift and so is cheap [12]. To calculate the best value for m, given 2^k consider:

$$\frac{m}{2^k} = \frac{1}{N} \Leftrightarrow m = \frac{2^k}{N}$$

For m to be an integer, we need to round 2^k somehow. Rounding to the nearest integer will give the best approximation but can result in $m/2^k$ being larger than $1/N$, which can cause underflows. Thus $m = \lfloor 2^k/N \rfloor$ is generally used. $m/2^k$ is only an approximation, and the error of the approximation of $1/N$ is: $e = \frac{1}{N} - \frac{m}{2^k}$.

Let k_0 be the smallest integer such that $2^{k_0} > N$. Take $k_0 + 1$ as a reasonable value for k in the above equations. Let

$$q = \left\lfloor \frac{mZ}{2^k} \right\rfloor \text{ and } r = Z - qN$$

Because of the floor function, q is an integer and $r \equiv Z (\bmod N)$. Also, if $Z < 2^k$ then $r < 2N$. In other words, we can write:

$$Z \bmod N = \begin{cases} r, & \text{if r¡N;} \\ r - N, & \text{otherwise.} \end{cases}$$

The proof that $r < 2N$ follows:

If $Z < 2^k$, then

$$\frac{Z}{2^k} . (2^k \bmod N) < N$$

Since $N.\frac{mZ \bmod 2^k}{2^k} < N$ regardless of Z, it follows that

$$\frac{Z.(2^k \text{mod} N)}{2^k} + N.\frac{mZ \bmod 2^k}{2^k} < 2N$$

$$\begin{aligned}
Z &- \left(Z - \frac{Z.(2^k \bmod N)}{2^k}\right) + \frac{N.(mZ \bmod 2^k)}{2^k} < 2N \\
Z &- \frac{Z}{2^k}.(2^k - (2^k \bmod N)) + \frac{N.(mZ \bmod 2^k)}{2^k} < 2N \\
Z &- \frac{NZ}{2^k}.\lfloor\frac{2^k}{N}\rfloor + \frac{N.(mZ \bmod 2^k)}{2^k} < 2N \\
Z &- \frac{NmZ}{2^k} + \frac{N.(mZ \bmod 2^k)}{2^k} < 2N \\
Z &- \left(\frac{mZ - (mZ \bmod 2^k)}{2^k}\right).N < 2N \\
Z &- \left\lfloor\frac{mZ}{2^k}\right\rfloor.N < 2N \\
Z &- qN < 2N \Rightarrow r < 2N
\end{aligned}$$

As a summary, Barret's basic idea was to replace the division with multiplication by a pre-computed constant which approximates the inverse of the modulus [8]. Thus the calculation of the exact quotient $q = \lfloor\frac{Z}{N}\rfloor$ is avoided by computing the quotient $\tilde{q}$ instead: $\tilde{q} = \left\lfloor\frac{\lfloor\frac{Z}{2^{n-1}}\rfloor\lfloor\frac{2^{2n}}{N}\rfloor}{2^{n+1}}\right\rfloor$

Although the equation of $\tilde{q}$ may look complicated, it can be calculated very efficiently, because the divisions by 2^{n-1} or 2^{n+1}, respectively, are simply performed by truncating the least significant $n-1$ or $n+1$ bits of the operands. The expression $\lfloor\frac{2^{2n}}{N}\rfloor$ depends on the modulus N only and is constant as long as the modulus does not change. This constant can be pre-computed, whereby the modular reduction operation is reduced to two simple multiplications and some operand truncations.

4.1.3. An efficient algorithm for modular reduction

We combined Montgomery and Barret modular reduction techniques to get an efficient implementation of modular multiplication. As an example consider computing $\mathtt{a}^5 \bmod \mathtt{m}$ for an integer $\mathtt{a}$, $1 \leqslant \mathtt{a} < \mathtt{m}$:

First, compute $\tilde{a} = \mathtt{aR} \bmod \mathtt{m}$ using Barret modular reduction method.

Then compute the Montgomery reduction of $\tilde{a}\tilde{a}$, which is $A = \tilde{a}^2\,\mathtt{R}^{-1}$ mod m.

The Montgomery reduction of A^2 is $A^2\,\mathtt{R}^{-1} \bmod \mathtt{m} = \tilde{a}^4\,\mathtt{R}^{-3} \bmod \mathtt{m}$ and the Montgomery reduction of $(A^2\,\mathtt{R}^{-1} \bmod \mathtt{m})\ \tilde{a}$ is $\tilde{a}^5\,\mathtt{R}^{-4} \bmod \mathtt{m} = a^5\,\mathtt{R} \bmod \mathtt{m}$, then multiplying this value by $R^{-1} \bmod \mathtt{m}$ and reducing modulo $\mathtt{m}$ gives $\mathtt{a}^5$ mod $\mathtt{m}$.

So the division needed for computing $a^n \bmod m$ (which is time consuming) is replaced by multiplication and just right shift which can be very fast.

In this way, the computation time is reduced and the speed of encryption is enhanced.

4.2 Effective applying of Chinese remainder theorem

The Chinese remainder theorem (CRT) is used as an efficient technique to speed-up the computation of modular exponentiation for the decryption operation of RSA public-key cryptosystems in which the private exponent is very large.

Using the CRT for a special case of $n = p.q$ (p, q are distinct primes) and using the Garner algorithm [12], to compute $x = C^d \bmod n$, first pre-compute:

$$p^{-1} \bmod q,$$

and then:

$$u = ((x \bmod q - x \bmod p).p^{-1} \bmod q) \bmod q$$

finally:

$$x = x \bmod p + u.p \tag{1}$$

According to Fermat's Little Theorem, let p be a prime, any integer a not divisible by p satisfies $a^{p-1} \equiv 1 \bmod p$.

As a consequence of this theorem, if an integer a is not divisible by p and if $n \equiv m \bmod (p-1)$, then $a^n \equiv a^m \bmod p$. So, $x \bmod p = (C^d \bmod n) \bmod p = C^d \bmod p$ (since $n = p.q$).

Using the consequence of Fermat's Little Theorem, x mod p can be written as:

$$c^{d \bmod (p-1)} \bmod p$$

Furthermore, it is easily observed that the cipher text c can be reduced modulo p before computing x mod p, i.e.:

$$x \bmod p = (c \bmod p)^{d \bmod (p-1)} \bmod p$$

Substituting the last equality into the equation 1 yields a formula for computing x that speeds up the calculation by scaling down the length of all operands by half. So, in this study, for computing $x = C^d \bmod n$, where $n = p.q$ and the primes p and q are known, first pre-compute:

$$\begin{aligned} p^{-1} \bmod q, \\ d \bmod (p-1), \\ d \bmod (q-1) \end{aligned}$$

and then:

$$\begin{aligned} t &= (C \bmod p)^{d \bmod (p-1)} \bmod p, \\ u &= ((C \bmod q)^{d \bmod (q-1)} \bmod q - t).(p - 1 \bmod q) \bmod q \end{aligned}$$

finally:

$$x = t + u.p.$$

The classical algorithm for the CRT typically requires a modular reduction with modulus n, whereas this algorithm does not need this requirement. If p, q are approximately $\frac{n}{2}$-bit integers, a modular reduction by n in multiplication and exponentiation takes $O(n^2)$ bit operations, whereas a modular reduction by p, q takes $O((\frac{n}{2})^2)$ bit operations. If hardware can do n-bit operations, it can do two $\frac{n}{2}$-bit operations in parallel and so the speed of calculations with modulus p, q is approximately 4 times faster than the calculations with modulus n, and so is more efficient.

4.3 Testing for primality and prime generation

An efficient algorithm for the problem Primality (n), where n is a large integer is essential for most of cryptosystems. There are no simple yet efficient means of determining whether a large number is a prime however there are many different stochastic and probabilistic primality test algorithms.

The probabilistic algorithm works as follows:

Given m, choose a random w with $1 \leqslant w < m$. The greatest common divisor (w, m), $gcd(w, m)$, is found by Euclid's algorithm (see e.g. [23]). If $gcd(w, m) > 1$, concludes that m is composite, otherwise compute $u = (w^{m-1} \bmod m)$ by repeated squaring. If $u \neq 1$ it means that m is composite. If $u = 1$, w is a witness for the primality of m, and there is evidence that m could be prime.

The more witnesses we find, the stronger the evidence will be. When we have found k witnesses, the probability of m being composite is at most 2^{-k}, except in the unfortunate cases, that all numbers w with $gcd(w, m) = 1$ and $w < m$ are witnesses. Such numbers are called Carmichael numbers but they are very rare (for more information see [23, 24]).

There are several probabilistic primality test algorithms [24], that test whether a number is prime with a given degree of confidence, including:

Lucas Test;

Solovay-Strassen Test;

Lehmer Test;

Rabin-Miller Test;

Fermat's Test;

The combination of two types of primality tests has been used in this study to determine whether a very large random number is a prime in a fast way and with a large degree of confidence. These tests are based on Fermat's theorem and Miller-Rabin method for primality.

The following algorithm is the method used in implementation to generate a prime number of the required length.

Prime generation algorithm:

Input: a positive integer p of length n (as required) as the start point.

Output: prime number p which is the next first prime after start point integer.

1 Check to make sure p is not divisible by any small primes: $3, 5, 7, \ldots$. The most efficient way is to test for divisibility by all primes less than 2000 (using the pre-computed Eratosthenes sieve [8]).

2 Perform the Miller-Rabin test with some random number a. If p passes, generate another random number a and go through the test again (choose a small value of a to make calculation quicker. If p fails increment p and go to step 1.

3 Return (p).

Another option is to generate a random p each time, instead of incrementing p and search until finding a prime.

5. Experimental Results

5.1 Key generation time

Figure 1 shows the experimental timing function for RSA key generation phase. The results are averaged over 20 different (random) data. As it is shown, it's considerably fast due to the efficient implementation of prime generation.

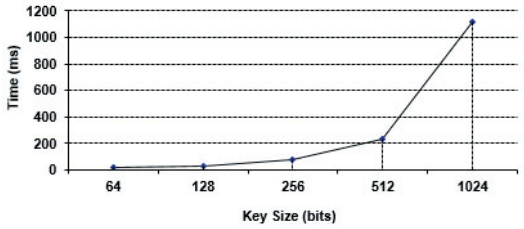

Figure 1. Timing curve for key generation

As it is seen from figure 1, key generation time grows exponentially in RSA system with key size.

5.2 Encryption/Decryption

Figures 2 and 3 show the experimental time for encryption and decryption operation of a file of 2000 bytes. As it is seen from the figures, the encryption and decryption time is decreased using an efficient algorithm for modular reduction introduced in this research (see section 4.1.3). Besides, the speed of decryption is enhanced by applying the optimization method for decryption operation using the Chinese remainder theorem.

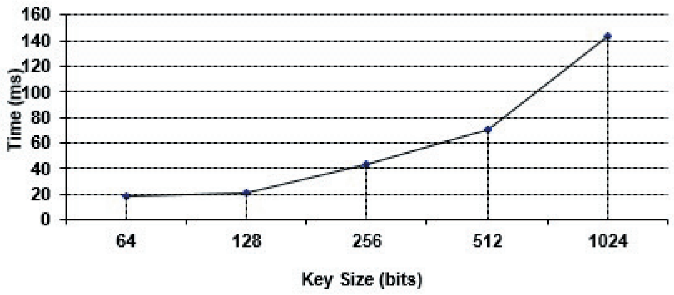

Figure 2. Timing curve for RSA-encryption

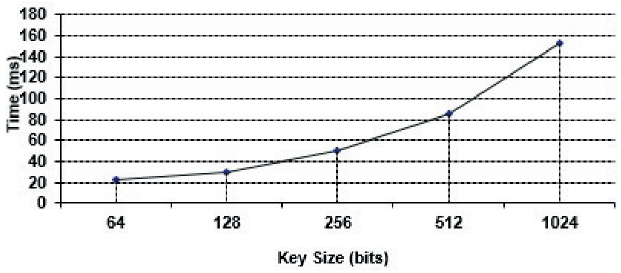

Figure 3. Timing curve for RSA-decryption

5.3 Comparison of the results

Table 1 shows a comparison of the execution time for the reduction of a $2k$-digit number modulo a k-digit modulus m for the Classical, Barrett, and Montgomery methods.

Table 1: execution time for modular reduction for the Classical, Barrett, and Montgomery methods (m sec.)

Length of m in bits	Classical	Barrett	Montgomery
128	5.12	2.13	61.05
256	34.85	3.43	62.94
512	49.34	12.37	63.75
1024	79.05	41.32	64.61
2048	131.65	137.26	65.45

As is seen in the above table, the Montgomery method is independent of the length of the argument. For the size of more than 1024 bits, that is usually used today, Montgomery is faster than the others. For the size of less than 1024 bits, Barrett is the best. This means that applying properly Barrett and Montgomery methods can yield a fast method that is used in the current research.

Table 2 shows a comparison of the results of the proposed algorithm implementation with the one presented by Zhou et al. in [22]. The key size for the key generation is 1024 bits. For encryption and decryption, a file of 1000 bytes is considered.

Table 2: comparison with the others

Technique Key	Generation Time(ms)	Encryption Time(ms)	Decryption Time(ms)
Zhou et al. [22]	1312.7	166.9	157.3
Implemented RSA	1116.3	142.8	152.9

6. Conclusion

In this paper, an efficient implementation of RSA that enhances the speed and security of the system is applied. It has the following three advantages.

First, speeding-up the encryption and decryption operation using the efficient implementation of modular exponentiation (combined Montgomery and Barret modular exponentiation) and optimization method for decryption operation using the Chinese remainder theorem.

Second, efficient implementation of the prime generation that leads to a considerably fast key generation phase.

Third, applying a variable key-size space on RSA which improves its security.

It should be noted that using a variable key-size space is equivalent to finding a prime number in all prime number space. Breaking such a system is impractical.

Using these issues makes a great difference in the implementation of RSA.

References

[1] R. Patidar and R. Bhartiya, *Modified RSA cryptosystem based on offline storage and prime number,* IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), (2013).

[2] J. Abudin, S. K. Keot, G. Malakar, N. M. Borah, and M. Rahman, Modified RSA Public Key Cryptosystem Using Two Key Pairs, *International Journal of Computer Science and Information Technologies,* 5 (3) (2014), 3548–3550.

[3] R. Stinson, *Cryptography: Theory and Practice,* CRC Press, USA, 2005.

[4] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM,* 21 (1978), 120–126.

[5] N. Gura, A. Patel, A. Wander, H. Eberle, and Sh. C. Shantz, *Comparing Elliptic Curve Cryptography and RSA on 8- Bit CPUs,* CHES 2004, (2004).

[6] H. Jeffrey, P. Jill, and H. Joseph, *An Introduction to Mathematical Cryptography,* Springer Science+Business Media, New York, 2008.

[7] D. R. L. Brown, Breaking RSA May Be As Difficult As Factoring, *Journal of Cryptology,* 29 (1) (2016), 220–241.

[8] J. Grobschadl, The Chinese remainder theorem and its application in a high-speed RSA crypto chip, *Proceedings of 16th Computer Security Applications Conference,* (2000), 384–393.

[9] W. Hasenplaugh, G. Gaubatz, and V. Gopal, *Fast modular reduction,* 18th IEEE Symposium on Computer Arithmetic (ARITH '07), (2007).

[10] Zh. Cao, R. Wei, and X. Lin, *A Fast Modular Reduction Method,* Computer Science IACR Cryptology ePrint Archive, (2014)

[11] S. Sharma, J. S. Yadav, and P. Sharma, Modified RSA Public Key Cryptosystem Using Short Range Natural Number Algorithm, *International Journal of Advanced Research in Computer Science and Software Engineering,* 2 (8) (2012), 134–138.

[12] A. J. Menezes, P. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography,* CRC Press, USA, 2003.

[13] P. Barrett, Implementing the Rivest, Shamir and Adleman public-key encryption algorithm on a standard digital signal processor. *Advances in cryptology:CRYPTO*, 86 (263) (1987), 311–323.

[14] N. Nedjah and L. M. Mouller, High-performance hardware of the sliding-window method for parallel computation of modular exponentitions, *International Journal of Parallel Programming,* 37 (2009), 537–555.

[15] A. K. Hussain, A Modified RSA Algorithm for Security Enhancement and Redundant Messages Elimination Using K-Nearest Neighbor Algorithm, *International Journal of Innovative Science, Engineering and Technology,* 2 (1) (2015), 159–163.

[16] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, High performance public-key cryptoprocessor for wireless mobile applications, *Mobile Networks and Applications,* 99 (2007), 245–258.

[17] A. Escala, G. Herold, E. Kiltz, and et al., An Algebraic Framework for Diffie-Hellman Assumptions, *Journal of Cryptology,* 30 (1) (2017), 242–288.

[18] J. S. Coron, T. Holenstein, R. Kunzler, and et al., How to Build an Ideal Cipher: The Indifferentiability of the Feistel Construction, *Journal of Cryptology,* 29 (1) (2016), 61–114.

[19] J. H. Seo, Short Signatures from DiffieHellman: Realizing Almost Compact Public Key, *Journal of Cryptology,* 30 (3) (2017), 735–759.

[20] A. F. Tenca and C. K. Koc, A scalable architecture for modular multiplication based on Montgomery's algorithm, *IEEE Trans. On computer,* 52 (9) (2003), 1215–1221.

[21] N. Pinckney, P. Amberg, and D. Harris, *Parallelized Booth-encoded radix-4 Montgomery multipliers,* Proceeding of 16th IFIP/IEEE International Conferene on Very Large Scale Integration, (2008).

[22] X. Zhou and X. Tang, *Research and implementation of RSA algorithm for encryption and decryption,* 6th International Forum on Strategic Technology (IFOST), (2015).

[23] T. H. Cormen, C. E. Lieserson, and R. L. Rivest, *Introduction to Algorithms,* The MIT Press, London, 2009.

[24] S. William, *Cryptography and Network Security: Principles and Practice,* Prentice Hall, USA, 2017.

Mahnaz Mohammadi
Ph.D of Electrical Engineering
Department of Electrical Engineering
Science and Research Branch, Islamic Azad University
Tehran, Iran
E-mail: mahnazlm@yahoo.com

Alireza Zolghadrasli
Associate Professor of Electrical Engineering
Department of Communication and Electronics
Faculty of Computer and Electrical Engineering
Shiraz University
Shiraz, Iran
E-mail: zolghadr@shirazu.ac.ir

Mohammad Ali Pourmina
Associate Professor of Electrical Engineering
Department of Electrical Engineering
Science and Research Branch, Islamic Azad University
Tehran, Iran
E-mail: pourmina@srbiau.ac.ir