International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 33996-34000

ModBlockExpoA- Enhancement on Speed and Security using Modularization in Public key Cryptograph

Dr.D.I.George Amalarethinam
Asso.Prof, Director-MCA,
Jamal Mohamed College,
Trichy, India.
di_george@ymail.com

J.Sai Geetha
Asst.Prof in Computer Science,
Nehru Memorial College,
Trichy, India .
jsaigeetha99@gmail.com

Dr.K.Mani
Asso.Prof in Computer Science,
Nehru Memorial College,
Trichy, India .
nitishmanik@gmail.com

***Abstract*—**In the speedy development of current information technology, security has become an important aspect in many applications of Internet. Cryptosystem supports secured data transmission. The security of the public key cryptography relies on the size of modular value. Modular exponentiation is one of the basic operations of public key cryptosystem. Whenever the key size increases, the modular value is also increased. It affects the speed of encryption and decryption process. In order to enhance the speed, it is necessary to improve the modular multiplication and exponentiation process. This can be achieved by reducing the number of modular multiplication. In this research work, a new proposed algorithm namely Modular Block Exponentiation Algorithm (ModBlockExpoA) is proposed. It comprises of three basic steps. The first step is to double the modular value(2n). In the next step, the plain text is divided into blocks and in the final step modular exponentiation technique is applied. This approach is a highly secured one, since the hackers cannot derive decryption key from the modular value(2n) and encryption key(e). It also resolves the factorization problem in RSA algorithm. The speed of public key cryptography is enhanced by the implementation of efficient modular multiplication and exponentiation algorithm. Thus the proposed algorithm ModBlockExpoA is suitable for any public key algorithm in order to achieve higher speed and enhanced security. The efficiency of the proposed algorithm is proved by comparing with existing algorithms from the literature.

Keywords— Public key cryptography, Encryption, Decryption, Modular multiplication, Exponentiation, Factorization.

1.INTRODUCTION

Nowadays, Internet technology and e-commerce application have developed rapidly. The main issue for a reliable communication is security. The confidential communication in which e-banking, ATM, business transactions through credit and debit card etc., need adequate security. Cryptography is one of the best ways to overcome the issues. It is a mathematical technique related to aspects such as confidentiality, data integrity, entity authentication and non-repudiation of both communication and information security. The data security plays a critical role in computer and communication systems. To provide such stringent security, most of the systems use public key cryptography. Modular exponentiation and modular multiplication are the essential operations of public-key cryptography. Both the encryption and decryption operations are accomplished by modular exponentiation. The key size is the main factor of security such that public key cryptosystems are typically slower than symmetric key cryptosystem. The large size key makes it difficult to achieve higher throughput. Hence, it is necessary to enhance the speed without affecting the security. In this work, the speed is achieved through the block cipher method. The block size is fixed based on the modulus value. The block value is also fixed to the nearest value of modulus. In ModBlockExpoA, the speed is further enhanced through the minimum difference between the modulus and block value of the plain text. Whenever the block size is increased, the speed is also increased. In general, one of the main constraints of modularisation is that the block size should be less than the modulus value. In the proposed algorithm, the size of modulus has to be increased twice. This approach resolves the constraint of block size increment. The enhanced modulus value also helps to overcome the factorization problem in public key cryptography. Thus the significant parameters of cryptosystem such as speed and security are improved by using the proposed algorithm ModBlockExpoA.

2.RELATED WORK

G.A.V.Rama Chandra Rao et.al.[1], proposed a new algorithm based on the remainder with regards to the modulus value n. It often happened that $2n+1$ and $2n+2$ can easily be factorized, even if it is difficult to factorize the prime factors. This procedure was faster than the existing algorithms and also calculated the computational complexity.

Koon-Shik Cho et.al.[2], introduced a Right-and-Left(RL) binary method for modular exponentiation because the number of clock cycles required to complete the modular exponentiation takes n cycles. Thus, 1024-bit RSA operation was done after $n(n / 2 + 3)$ clock cycles.

Vollala.S et.al.[3], proposed a Bit forwarding (BFW) algorithms to compute a^x mod n, and to compute $a^x b^y$ mod n two algorithms ,namely, Substitute and reward (SRW), Store and forward(SFW) are proposed. This algorithm was suitable for devices with low computational power and limited storage.

Xucheng Yin et.al.[4], introduced a binary modular exponentiation RSA counter measure in order to defend against the comparative power analysis by dividing the private key einto random parts and randomly choosing one of the parts to do one unit operation, each selection till the modular exponentiation of all parts are completed.

Can Xiang et.al.[5] presented two secure outsourcing scheme for modular exponentiations, which enables users to securely outsource modular exponentiations to single untrusted cloud server and detect its dishonest behavior.The first method was a base variable exponent modular exponentiation and the second method was simultaneous modular exponentiation.

Sridevi and Manahajaih[6] proposed a modular exponentiation with Residu Number System(RNS).The modular exponentiation was divided into sequence of modular multiplications. The RNS Montgomery multiplication algorithm needed more computational effort due to the two required base extensions.

3. PROPOSED METHODOLOGY

3.1 Modular Exponentiation

In public key cryptosystem, modular exponentiation is the main component for encryption and decryption process in most of the algorithms particularly in Rivest Shamir Adlemen (RSA) algorithm.

The exponentiation of plain text 'M' is found by using encryption key 'e' and also find the reminder with the modulus value 'n' which is known as encryption process.

$C=M^e \bmod n$

The exponentiation of cipher text 'C' is found by using the decryption key 'd' and also find the reminder with modulus value 'n' which is known as decryption process.

$M=C^d \bmod n$

The speed of the algorithm is affected by the modular exponentiation process in both encryption and decryption.

3.2 Modular Exponentiation Technique

Modular exponentiation is a type of exponentiation performed over a modulus. In this method, an integer 'a' raised to the exponent 'e'. The obtained result is divided by the positive integer modulus 'n' as given in eq.1. The modular exponentiation is

$$c= a^e \bmod n \quad \text{--------- eq.1}$$

where a = 10, e = 3 and n = 13, the solution c = 12 is the reminder of dividing

$10^3 = 1000$ by 13.

Given integers 'a' and 'e', and a positive integer 'n', a unique solution c exists with the property $0 \le c < n$.

In the proposed fast modulus exponentiation technique, the speed is achieved through the subtraction of 'n' from 'a'. The resultant value is raised to the exponentiation 'e' as represented in eq.2. The final result 'c' is the difference between 'c'' and 'n' as given in eq.3.

$$c' \equiv (a - n)^e \bmod n \quad \text{-------- eq.2}$$

where a = 10, e = 3 and n = 13, the solution c'= 1 is the reminder of dividing

$(10-13)^3 \equiv (-3)^3$ by 13 [13]

$\equiv 27$ by 13

$$c \equiv c' - n \quad \text{---------- eq.3}$$

$c \equiv (1-13) = abs(-12) = 12.$

3.3 Reminder Multiplication Algorithm

The reasonable factor '2n' is introduced in place of modulo 'n'. Let 'a', 'b', and 'n' be three n-bit positive binary integers, where a, b < n and gcd (2,2n) = 2. 2n can be divided into products of prime factors. It is proved by using reminder multiplication algorithm [9].

Let y = xu mod n

Then y can be represented as follows:

$y1 \ge y2 \quad y \equiv 2y1 - y2 \pmod n$

$y1 < y2 \quad y \equiv 2y1 - y2 + 2 \pmod n$

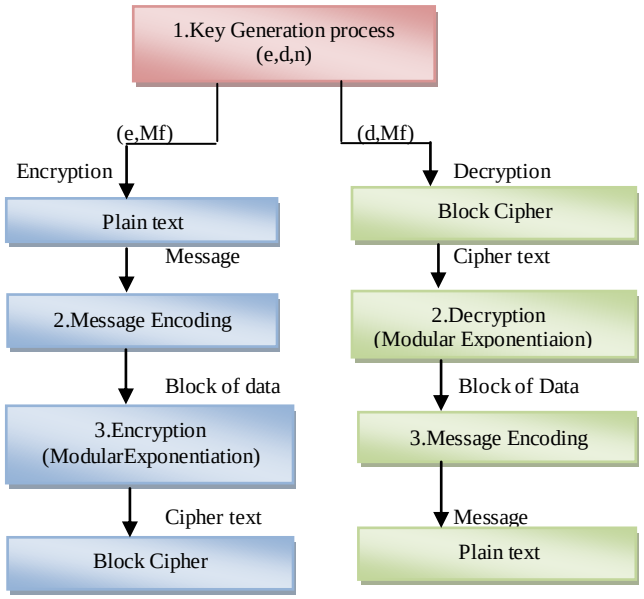
The positive integer 2n is divided into product of mutually prime factors in eq.4.

$$2n=\prod_{i=1}^{m} P_I \text{---------- eq.4}$$

where $P_1, P_2, \ldots, P_m$ are prime factors.

3.4 ModBlockExpoA

The main objective of this algorithm is to enhance the speed as well as security. This algorithm consists of three inter-dependent tasks which is given in Fig.1.



In the key generation process, the value of modulus 'n' is changed into Mf (false modulus). The plaintext is separated into blocks based on the modulus value Mf. Finally, the block value is taken as the plain text and Mf as the modulus in the encryption process. In the same way, the block cipher and modulus Mf is taken for decryption. The modularization performs by applying *ModBlockExpoA*. In this work, speed is dependent on parameters such as the value of Mf, block size and the difference between block value of plain text and Mf. One of the security issues in public key cryptosystem is factorization. It is also resolved by the modulus value Mf. It is illustrated in the pseudo code given in Fig.2.

The encryption key (e), decryption key (d) and false modulus (Mf) are generated in the key generation process by using reminder multiplication method. The plain text M is seperated into blocks $b_1, b_2, \ldots, b_m$. After encryption, these are converted into block cipher $C_1, C_2, \ldots C_m$. It is reversed into block of plain text in decryption. Both encryption and decryption use the new modulus Mf and also using modular multiplication technique. Modulus cannot be calculated for

negative value in general. In this case, it is also measured by applying the Illiac IV routing functions.

Step 1: Key generation
Choose two distinct prime numbers p and q.
Compute n = pq.
Compute φ(n) = (p − 1)(q − 1)
Choose an integer e such that 1 < e < φ(n) and gcd(e, φ(n)) = 1; i.e., e and φ(n) are coprime. e is the public key exponent.

Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; i.e., d is the modular multiplicative inverse of e (modulo φ(n)).

$d \cdot e \equiv 1 \pmod{\varphi(n)}$

Finds the "Mf" Where "Mf" is the modulus factor which is multiplied by "2n". The new modulus "Mf" is used in place of actual modulus "n" and does the process of encryption and decryption. [Remainder Multiplication Algorithm]
Public key = {e, Mf} and private key = {d, Mf}

Step 2: Message Encoding
Plain text (M)=$b_1,b_2,....,b_m$ where $b_1,b_2,....,b_m$ are blocks and $1< b_i < 2n$. [14]

Step 3: Encryption
If (b_i < Mf)
$b_i=(b_i*2 – Mf)$ [13]
else
$b_i=(b_i – Mf)$ [13]
$C'_i = (b_i)^e \bmod Mf$ [Modular exponentiation technique]
$C_i= C'_i – 2n$ where i=1,2, ...m, Block cipher Ci = $C_1,C_2,...C_m$

Step 4: Decryption
If(Ci < Mf)
$C_i=(C_i*2 - Mf)$ [13]
Else
$C_i=(C_i - Mf)$ [13]
$b'_i=(C_i)^d \bmod Mf$
bi= b'_i - 2n where i=1,2, ...m, Plain text M=$b_1,b_2,$ ... b_m

Fig.2.Pseudo code for *ModBlockExpoA*

4. RESULTS AND DISCUSSION

The speed and security are the two important factors of public key cryptography. The proposed algorithm *ModBlockExpoA* measures the speed with various file size and key size by using RSA Algorithm. The *ModBlockExpoA* have been implemented in Java.

In Table 1, the file size is considered as 10 KB and the plain text is 1844627238521774924.The plain text is obtained by applying both n(existing) and Mf(proposed) with different key sizes. The plain text is retained irrespective of cipher text value is either common or different. If anyone tries to factorize the "Mf", it is not possible to find out the original value of decryption key, because it is a false value. The outcome produced by using Mf is not an accurate result.

The results of Table1 support to create high speed cryptosystem in addition to security. It is proved with modular exponentiation method using 10KB size of plain text file and the results are tabulated in Table 2. These experiments are done by using different key size and file size of plain text and also the execution time is analysed. The experimental results are compared with the existing methods such as "Square and multiply" and "MulMod" method [9].

Table 2. Comparison of execution time

Key Size (Bits)	Block size (Bits)	No of blocks	Total Time(ms)		
			RSA (Square and Multiply)	RSA (Mul Mod)	RSA (Mod Block ExpoA)
128	255	327	532	320	265
256	511	163	1562	953	780
512	1023	82	5344	3327	2656
1024	2047	41	19938	16952	9854
2048	4095	21	76656	56129	38094

In the results of Table 2, the execution speed purely depends on the block size. The block size is fixed as twice the size of key and less than the value of modulus. For instance, when the key size is 128 and the modulus value is 256, the block is fixed as 255. It is substantiated that the time taken for the cryptosystem drastically reduced by using *ModblockExpoA* as in the case of 2048 bits key size. The graphical representation of the above is given in Fig.3. An identical comparison is also done with increased plain text file of size 20 KB with different key sizes.

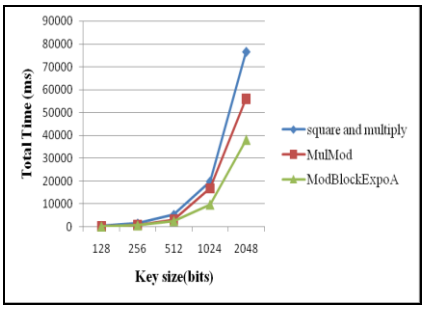


Fig.3. Comparison of execution speed

In Fig.3, the speed of the public key cryptosystem is improved irrespective of the key size and file size. It is obtained by using the appropriate block size and exponentiation method. When the block value of the plain text is reduced without changing its size, it increases the speed of Cryptosystem. Finally, it proves that the *ModBlockExpoA* provides better performance than "Square and multiply" and "MulMod" method. The features of algorithms are compared in the Fig.4.

Table 1.Analysis of modulus value with different key size

Key Size (bits)	Encryption key(e)	Decryption key(d)	Modulus value		Cipher text		Plain text
			N	Mf	n	Mf	
32	2999637769	219304469693489 0257	627538007528806 963	125590760150576 13926	5516696037388227 341	1179623404491703 4304	18446272385217749 24
	4100408393	539762656967981 9057	576080323120066 7267	115216064624013 34534	2237111680584186 310	2237111680584186 310	18446272385217749 24
	3386568233	125345148670091 2505	100341745860421 34783	200683491720842 69566	2404598497107724 257	1243877308314985 9040	18446272385217749 24
40	939830998871	197003887816431 422859431	337340380899817 862494087	674680761799635 724988174	1505853635008614 93547932	1505853635008614 93547932	18446272385217749 24
	1090530539023	642514283512956 556352791	811544530162341 677304373	162308060324683 354608746	7638190261603836 63249053	1575363556322725 340553426	18446272385217749 24
	1012653702083	799904503453625 31145079	498020735248034 033413061	960414704960680 66826122	2594408650011700 23610132	2594408650011700 23610132	18446272385217749 24

RSA with MulMod	RSA with ModBlockExpoA
Publicly announced original modulus	Publicly announced false modulus Mf
e and d computed from public modulus value	e and d computed from modulus Mf
Limit of plain text is 0≤m<n	Limit of plain text is 0≤m<Mf
The strength of variables are only two p and q	The strength of variables are only three p and q and Mf
The plain text is directly applied to the encryption.	Before encryption, the block value of plain text is reduced.
Intruders can easily derive the plain text from cipher text	Deriving the plain text from the cipher text is much difficult for any Intruders.

Fig.4. Comparison of features between MulMod and ModBlockExpoA

In the aspect of security, the key size is an important parameter in any public key algorithm. It is not possible to reduce the key size in view of security threat. In the proposed algorithm *ModBlockExpoA*, the selection of large size key is not affecting the execution speed.

5. CONCLUSION

Security is an important feature in many applications of internet. The basic operation of public key cryptosystem particularly RSA is modular exponentiation which is performed by ModBlockExpoA. The proposed algorithm is based on the idea that the reminder for modulus 'n' can be replaced from the reminder with false modulus Mf. This method makes it more complication to find the decryption key. The block value of the plain text is increased with respect to the value of Mf. Finally, the fast exponentiation method reduces the block value of plain text before finding the exponentiation. The proposed algorithm *ModBlockExpoA* greatly enhances the speed of public key cryptosystem, in addition to security enhancement by using large key size. Incidentally, factorisation problem is also resolved. All the experiments results prove that the *ModBlockExpoA* provides better speed and security than traditional methods such as square and multiply and MulMod algorithm.

References

[1]. Rama Chandra Rao GAV., Lakshmi P.V. and Ravi Shankar.N., 2013, "A New Modular Multiplication Method in Public Key Cryptosystem", International Journal of Network Security, 15(1), PP.23-27.

[2]. Koon-Shi Cho, Je-Hyuk Ryu and Jun-Dong Cho, 2000, "High-Speed Modular Multiplication Algorithm for RSA Cryptosystem", KOSEF (Korea Science and Engineering Foundation) under the grant No.2000-2-30300-004-3 and IDEC (IC Design Education Center)

[3]. Vollala.S., Varadhan.V.V. , Geetha.K. and Ramasubramanian.N., 2014, " Efficient modular multiplication algorithms for public key cryptography", Advance Computing Conference (IACC), 2014 IEEE International , PP:74 – 78.

[4]. Xucheng Yin , Keke Wu , Huiyun Li and Guoqing Xu, 2012, "A randomized binary modular exponentiation based RSA algorithm against the comparative power analysis", Intelligent Control, Automatic Detection and High-End Equipment (ICADE), 2012 IEEE International Conference, PP:160 – 165.

[5]. Can Xiang and Chunming Tang, 2015, "Efficient outsourcing schemes of modular exponentiations with checkability for untrusted cloud server", J Ambient Intell Human Comput 6(1), PP:131-139.

[6]. Sridevi and Manajaih.D.H, 2014,"Modular Arithmetic in RSA Cryptography", International Journal of Advanced Computer Research ,4(17).

[7]. Aggarwal.D., Maurer.U and Shparlinski.I, 2011, "The equivalence of Strong RSA and factoring in the Generic Ring Model of computation," WCC 2011-Workshop on Coding and Cryptography, PP:17-26.

[8]. Basu.S 2012, "A new parallel window based implementation of the Elliptic curve point multiplication in multi-core architectures," International Journal of Network Security, 14(1), PP:52-59.

[9]. Rama Chandra Rao GAV, Lakshmi P.V and Ravi Shankar.N, 2012, "A Novel Modular Multiplication Algorithm and its Application to RSA Decryption", IJCSI International Journal of Computer Science Issues, 9(6).

[10]. Ambika R and Hamsavahini R, 2013, " A Survey on Hardware Architectures for Montgomery Modular Multiplication Algorithm", IJETCAS, PP:13-342.

[11]. Shiann-Rong Kuang, Jiun-Ping Wang, 2013, "Energy-Efficient High-Throughput Montgomery Modular Multipliers for RSA Cryptosystems", IEEE Transactions on very large scale integration (VLSI) systems, 21(11), PP:1999-2009.

[12]. Neto J.C., Tenca.A.F. and Ruggiero W.V., 2011, "A parallel k-partition method to perform Montgomery multiplication" IEEE Int. Conf. Appl.-Specif. Syst., Arch. Process., PP. 251-254.

[13]. Orcutt.S.E, 1976, "Implementation of Permutation Functions in Illiac IV –Type Computers", IEEE Transactions on Vol C-25(9), PP:929-936.

[14]. Amalarethinam, D. G., Geetha, J. S., & Mani, K. (2015). Analysis and Enhancement of Speed in Public Key Cryptography using Message Encoding Algorithm. Indian Journal of Science and Technology, 8(16).