RESEARCH ARTICLE
doi: 10.2306/scienceasia1513-1874.2013.39S.103

ScienceAsia 39S (2013): **103–106**

Numerical analysis of parallel modular exponentiation for RSA using interconnection networks

Masumeh Damrudi*, **Norafida Ithnin**

Department of Computer System and Communication, Faculty of Computing, Universiti Teknologi Malaysia, 81310, UTM, Skudai, Johor Darultakzim, Malaysia

*Corresponding author, e-mail: m.damrudi@gmail.com

Received 7 Jan 2013
Accepted 5 Apr 2013

ABSTRACT: Security is important for transferring confidential information such as passwords through unreliable networks. The prominent way to overcome this issue is cryptographic algorithms. While a longer key consumes more time in cryptosystem operations for attackers, the RSA as a public key is more secure and uses a longer key. This paper focuses on the numerical analysis of using parallel processing as well as interconnection networks for RSA cryptographic algorithms to achieve suitable results in terms of speed. Numerical results are employed to show that the analysed approach is faster than accepted approach which is the binary method.

KEYWORDS: public key, parallel processing, speedup

INTRODUCTION

Cryptography uses mathematical techniques to achieve information security such as confidentiality, integrity, authentication and non-repudiation employing some techniques such as functionality, performance and ease of implementation. Public key was invented mostly to solve the certain problem of secret key distribution[1,2]. While all of the application data transferred throughout the networking environments is ciphered with a secret key, secret key distribution is a tough job. Managing secret key of networks is feasible only for small number of network users. Secret key cryptography by itself is not an option for a network of millions of users, so it cannot support internet e-commerce[3]. It is a big disadvantage on a key exchanging and a limitation on the number of keys which is up to n^2 in symmetric key, where n is the number of participants in a group. A large number of keys needs to be maintained in a large group. In contrast, the number of keys which needs to be maintained in a group is only $2n$ in asymmetric-key cryptography[2]. Among other existing asymmetric algorithms, the RSA algorithm is the most practical public key system[4] in use yet[5,6] because of its security and easy implementation. At first, RSA was secure for 512 bit key length. By the development of technology and attacks, this key length is not secure any more. In the near future 2048 and 4096 will be secure key lengths for RSA cryptography. The problem of using greater key is having more multiplications which leads to less speed. Binary method is the accepted exponentiation algorithm for RSA which needs $2(k-1)$ multiplications in the worst case where k is the length of key in bit[7–9]. There are some parallel approaches to improve modular exponentiation for RSA which are summarized in Ref. 10. This paper employs numerical analysis for the parallel processing approach with tree interconnection network to approve the advantages of cryptography in terms of speed.

BACKGROUND OF THE STUDY

RSA cryptosystem

Two prime numbers p and q are selected and n is achieved by $n = pq$. ϕ is $\phi = (p-1)(q-1)$. Then an integer e should be chosen where $1 < e < \phi$, such that $\gcd(e, \phi) = 1$. Using extended Euclidean algorithm, which is explained as following[6]

Algorithm 1 (Extended Euclidean algorithm) Given A, $B > 0$, set $x_1 = 1$, $x_2 = 0$, $y_1 = 0$, $y_2 = 1$, $a_1 = A$, $b_1 = B$, $i = 1$.

repeat while $b_i > 0$ {
$i = i + 1$
$q_i = a_{i-1} \operatorname{div} b_{i-1}$
$b_i = a_{i-1} - q_i b_{i-1}$
$a_i = b_{i-1}$
$x_{i+1} = x_{i-1} - q_i x_i$
$y_{i+1} = y_{i-1} - q_i y_i$
}

$\forall i : Ax_i + By_i = a_i$
last $a_i = \gcd(A, B)$

d is computed in such a way that $ed = 1 \pmod{\phi}$ and $d = \mathrm{e}^{-1} \pmod{\phi}$ where $1 < d < \phi$. The value set of e and n is known as public key and d is the private key. By considering m as the plaintext in the encryption process, it will be divided into blocks smaller than n. $C_i = m_i^e \bmod n$ is the encryption process and $m_i = C_i^d \bmod n$ is the decryption process. The index i is used for representing block number.

Binary method

Modular exponentiation is a very important operation[11] for the public key cryptography systems such as RSA. Many researchers believe that the low speed of RSA and some other public key cryptographic algorithms is due to the low speed of the exponentiation computation for a large number[12,13] which can be accelerated by making the modular exponentiation operation fast. The classical method of modular exponentiation is binary method. The binary method uses the following algorithm to calculate $C = m^e \bmod n$.

$$e = (e_{k-1}, e_{k-2}, \ldots, e_1, e_0) = \sum_{i=0}^{k-1} e_i 2^i, e_i \in (0, 1) \tag{1}$$

Algorithm 2 (Binary algorithm)
if $e_{k-1} = 1$ then $C = m$ else $C = 1$
for $i = k - 2$ downto 0
$C = C \times C \pmod{n}$
if $e_i = 1$ then $C = C \times M \pmod{n}$

This algorithm checks every bit of the power (key) and for each non-zero bit the multiplication of the plaintext is multiplied to the results provided by the previous step. The number of multiplications for binary method in the best, average, and worst case are $(k-1)$, $1.5(k-1)$, and $2(k-1)$, respectively[9,14].

NUMERICAL ANALYSIS OF THE NEW APPROACH

Parallel processing is a way of performing operations faster especially when the task has the ability to be divided into smaller parallel tasks. Employing interconnection networks provides us a better infrastructure to perform parallel processing. Parallel processors are computer systems consisting of multiple processing units connected by some interconnection networks, including the software required to make the processing units cooperate[15]. The processing units and the interconnection network are two major factors to classify these computer systems. The interconnection network is an important component in a parallel computer. Topology is a mapping function which draws the connection between the set of processors and memories to other processors and memories[15]. Typically the interconnection networks are divided into static and dynamic. Static networks such as mesh, ring, tree, cube. have fixed connections. Due to the different architectures with different specifications, researchers select the topology which is more compatible with the nature of their algorithm and suits better with a reasonable cost. For an interconnection network, the total number of links is expected to be as low as possible[16]. Topological cost of the network depends on the number of links and its diameter[15,16]. The tree architecture is altered to be employed as an interconnection network which has logarithmic diameter as well as less processor elements in comparison to the tree. The conversion of tree to this interconnection network is explained in Ref. 17. Fig. 1 shows the altered architecture.

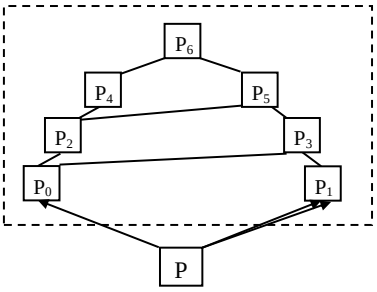

Fig. 1 The altered architecture.

The shape with dashed area illustrates the altered architecture that can be exploited as a crypto-processor.

Algorithm 3 (The new algorithm)
$L =$ levels of architecture
$n = pq$
$n_p = 2^L$
$e_o = e \operatorname{div} n_p$
$e_l = e_o + e \bmod n_p$
P:
$A = m^{e_o} \bmod n$
$B = m^{e_l} \bmod n$
$P_0 = A \times A \bmod n$ $P_1 = A \times B \bmod n$
for $i = 2$ to $N - 2$ step 2 for $i = 3$ to $N - 1$ step 2
$P_i = P_{i-2} \times P_{i-2} \bmod n$ $P_i = P_{i-2} \times P_{i-3} \bmod n$
$P_N = P_{i-1} \times P_{i-2} \bmod n$

Table 1 The numerical example of binary method in new approach.

Step	e_k	New approach(P)
0	$e_7 = 1$	$C = 121 \pmod{217}$
1	$e_6 = 1$	$C = 121 \times 121 \pmod{217} = 102$
2	$e_6 = 1$	$C = 102 \times 121 \pmod{217} = 190$
3	$e_5 = 1$	$C = 190 \times 190 \pmod{217} = 78$
4	$e_5 = 1$	$C = 78 \times 121 \pmod{217} = 107$
5	$e_4 = 1$	$C = 107 \times 107 \pmod{217} = 165$
6	$e_4 = 1$	$C = 165 \times 121 \pmod{217} = 1$
7	$e_3 = 0$	$C = 1 \times 1 \pmod{217} = 1 \Rightarrow A$

Table 2 The numerical example of binary method in new approach(coprocessor).

Step	$P_i(i \bmod 2 = 0)$	$P_i(i \bmod 2 = 1)$
8	$C = 1 \times 1 \pmod{217}$	$C = 1 \times 107 \pmod{217}$
9	$C = 1 \times 1 \pmod{217}$	$C = 1 \times 107 \pmod{217}$
10	$C = 1 \times 107 \pmod{217}$	-

This new algorithm has changed the modular exponentiation process as follows to give better results in terms of speed:

$$r = e \bmod 2^L$$

$$m^e = m^{e/2^L} \times m^{e/2^L} \times \cdots \times m^{e/2^L} \times m^{e/2^L + r}$$

$$m^e = m^r \prod_{i=1}^{2^L} m^{e/2^L}. \qquad (2)$$

This operation is performed in parallel as presented in the pseudocode. The following tables show the operation with a numerical example for new approach in comparison to binary method. Considering that $m = 121$, $p = 7$, $q = 31$ then $n = 217$ and $\phi = 180$. As $\gcd(247, 180) = 1$, therefore, $e = 247 = (11110111)_2$.

The steps for binary method are 13 for the given numerical example as it is shown in Table 3. The steps of this example in the new approach are presented in Tables 1 and 2. Assume the number of levels of the architecture is three. The key is 247, then,

$$n_p = 2^3 = 8$$
$$e_\mathrm{o} = e \text{ div } n_p = 30 = (11110)_2$$
$$e_\mathrm{l} = e_\mathrm{o} + e \bmod n_p = 30 + 7.$$

The operation of $B = 1 \times 107 \pmod{217}$ is performed in the coprocessor in parallel with step 8. The number of steps is decreased in the new approach. It must be considered that if the number of processor elements in the coprocessor becomes more, the speed will be increased particularly when the pipelining is added to the approach. The processor elements are idle until their turn is arrived. The architecture can be used in a way that while the upper level processor elements are encrypting the previous message the following processor elements encrypt the next message.

Table 3 The numerical example for binary method.

Step	e_k	Binary method
0	$e_{k-1} = e_7 = 1$	$C = 121 \pmod{217}$
1	$e_6 = 1$	$C = 121 \times 121 \pmod{217} = 102$
2	$e_6 = 1$	$C = 102 \times 121 \pmod{217} = 190$
3	$e_5 = 1$	$C = 190 \times 190 \pmod{217} = 78$
4	$e_5 = 1$	$C = 78 \times 121 \pmod{217} = 107$
5	$e_4 = 1$	$C = 107 \times 107 \pmod{217} = 165$
6	$e_4 = 1$	$C = 165 \times 121 \pmod{217} = 1$
7	$e_3 = 0$	$C = 1 \times 1 \pmod{217} = 1$
8	$e_2 = 1$	$C = 1 \times 1 \pmod{217} = 1$
9	$e_2 = 1$	$C = 1 \times 121 \pmod{217} = 121$
10	$e_1 = 1$	$C = 121 \times 121 \pmod{217} = 102$
11	$e_1 = 1$	$C = 102 \times 121 \pmod{217} = 190$
12	$e_0 = 1$	$C = 190 \times 190 \pmod{217} = 78$
13	$e_0 = 1$	$C = 78 \times 121 \pmod{217} = 107$

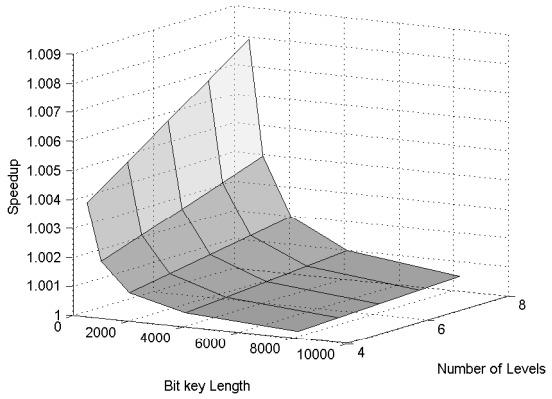

Fig. 2 The speedup of new approach to the binary method.

ANALYSIS

The number of multiplications for binary algorithm in the worst case is $2(k-1)$ which is decreased in the new approach to $2(k-1) - L$ where L is the number of levels in tree. For the numerical example, the 13 steps decreased to 10 due to three levels. The speedup is (old approach time)/(new approach time).

Therefore the speedup is $2(k-1)/(2(k-1)-L)$, which is $13/10 = 1.3$. Fig. 2 shows the speedup of new approach to the accepted method, which is binary method. The speedup is based on different bit key lengths and levels of the architecture.

Fig. 2 indicates that having more number of levels which means more processing elements, the speedup will be increased. The most important part is the pipelining; by employing pipelining the throughput will increase. The throughput of a pipeline approach is $L+1$ times that of the main approach in a time slice.

CONCLUSIONS

This paper illustrates a numerical example of employing tree interconnection network to carry out RSA encryption. The approach employs parallel processing and pipelining to achieve more speed and throughput. As it is seen from Fig. 2, Table 1 and Table 2, the new method decreases the number of multiplications. The new approach also has more throughput than the accepted binary method.

REFERENCES

1. Thorsteinson P, Arun Ganesh G (2004) *.NET Security and Cryptography*, Pearson Education Inc., Prentice Hall.
2. Teerakanok T, Kamolphiwong K (2009) Accelerating asymmetric-key cryptography using parallel-key cryptographic algorithm (PCA). *ECTI-CON 2009*, vol 2, pp 812–5.
3. Mel HX, Baker D (2001) *Cryptography Decrypted*, Addison Wesley.
4. Qing L, Yunfei L, Lin H (2010) On the design and implementation of an efficient RSA variant, *ICACTE 2010*, vol 3, pp 533–6.
5. Smart N (2003) *Cryptography: An Introduction*, McGraw-Hill.
6. Elbirt AJ (2009) *Understanding and Applying Cryptography and Data Security*, Taylor & Francis, New York.
7. Brickell EF (1990) A survey of hardware implementation of RSA. In: *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, Springer-Verlag, pp 368–70.
8. Zhang CN (1993) An improved binary algorithm for RSA. *Comput Math Appl* **25**, 15–24.
9. Koc CK (1994) High-speed RSA implementation. In: *RSA Laboratories 2.0*, RSA Data Security Inc., Redwood City, CA.
10. Damrudi M, Ithnin N (2011) State of the art practical parallel cryptographic approaches. *Aust J Basic Appl Sci* **5**, 660–77.
11. Perin G, Mesquita DG, Herrmann FL, Martins JB (2010) Montgomery modular multiplication on reconfigurable hardware: Fully systolic array vs parallel implementation, *In: VI Southern Programmable Logic Conference (SPL)*, pp 61–6.
12. Hayashi A (2000) A new fast modular multiplication method and its application to modular exponentiation-based cryptography. *Electron Comm Jpn III* **83**, 88–93.
13. Fan W, Chen X, Li X (2010) Parallelization of RSA algorithm based on compute unified device architecture. In: *9th International Conference on Grid and Cooperative Computing (GCC)*, pp 174–8.
14. Sepahvandi S, Hosseinzadeh M, Navi K, Jalali A (2009) An improved exponentiation algorithm for RSA cryptosystem. In: *International Conference on Research Challenges in Computer Science*, pp 128–32.
15. El-Rewini H, Abd-El-Barr M (2005) *Advanced Computer Architecture and Parallel Processing*, Wiley.
16. Gopalakrishna Kini N, Sathish Kumar M, Mruthyunjaya HS (2009) Performance metrics analysis of torus embedded hypercube interconnection network. *Int J Comput Sci Eng* **1**, pp 78–80.
17. Damrudi M, Ithnin N (2012) An optimization of tree topology based parallel cryptography. *Math Probl Eng* **2012**, 1–10.