Design and Implementation of an Improved RSA Algorithm

Yunfei Li
School of Information Science and Engineering
Yunnan University
Kunming, China
e-mail: liyunfei67@163.com

Qing Liu, Tong Li
National Pilot School of Software
Yunnan University
Kunming, China
e-mail: liuqing@ynu.edu.cn

***Abstract*—This paper aims at speeding up RSA decryption and signature. The performance of RSA decryption and signature has direct relationship with the efficiency of modular exponentiation implementation. This paper proposes a variant of RSA cryptosystem (EAMRSA-Encrypt Assistant Multi-Prime RSA) by reducing modules and private exponents in modular exponentiation. The experimental result shows that the speed of the decryption and signature has been substantially improved and the variant can be efficiently implemented in parallel.**

Keywords-RSA ; acceleration; modular exponentiation; parallel

I. INTRODUCTION

The RSA cryptosystem [1] is one of the most widely used public key systems. It is the main operation of RSA to compute modular exponentiation. Since RSA is based on arithmetic modulo large numbers, it can be slow in constraining environments. Especially, when RSA decrypts the ciphertext and generates the signatures, more computation capacity and time will be required. Reducing modulus in modular exponentiation is a technique to speed up the RSA decryption. Multi-Prime RSA [3] [4] which speeds up the RSA decryption reduces the size of the moduli. Another method which speeds up the RSA decryption is to shift some work to the encryption and the exponents of decryption become small numbers. But most of the variants were only considered from one aspect and didn't combine other aspects. This paper proposes a new variant of RSA based on the reduction of the modulus and private exponents in modular exponentiation of the decryption. The variant cannot only speed up RSA decryption but also can guarantee the security of RSA cryptosystem.

The experimental results on the single-core processor computer show that the new variant can achieve an average factor of 5.92 speedup from 1795- to 3072-bit. Now, lots of multi-core devices are being introduced into the market .The architecture for multi-core processors can improve the performance of processors by parallel work [2]. In this paper, the algorithm has obvious parallel features by parallel analysis, and is easy to be efficiently implemented in parallel. The algorithm can execute in parallel on multi-core devices, giving full play to the parallel processing capabilities of these devices to further improve the performance of RSA.

The paper is organized as follows. In section 2, the new variant of RSA is proposed and its security and parallelism are also discussed in this section. In section 3 the experimental results are presented and the performance of the new variant is analyzed. Related work is reviewed in section 4 and section 5 concludes the paper.

II. THE PROPOSED VARIANT

In this section, a new variant of RSA is proposed and is called EAMRSA (Encrypt Assistant Multi-Prime RSA) in the paper. The variant effectively combines Multi-Prime RSA [3] [4] and RSA-S2 system [5]. It can obtain a higher speedup than the basic RSA and the above two RSA variants. The variant also has obvious parallel characteristics and is easy to be implemented in parallel. Before the proposal of optimizing the RSA cryptosystem is presented, the RSA-S2 system will be discussed.

A. RSA-S2 System

The RSA-S2 [5] was originally proposed as a way to reduce load on small devices (smartcards) by shifting some heavy-weight cryptographic computation to more powerful server-host computers equipped with smartcard readers. The detail of the RSA-S2 is as follows:

- The client randomly generates an integer vector $D=(d_1,d_2,...,d_k)$ and two binary vectors $f=(f_1,f_2,...,f_k)$ and $g=(g_1,g_2,...,g_k)$ such that $d_p=\sum_{i=1}^{k} f_i d_i \bmod(p-1)$ $d_q=\sum_{i=1}^{k} g_i d_i \bmod(q-1)$
- The client sends n, D and x to the server.
- The server computes $Z=(z_1,z_2,...,z_k)$ and sends Z back to the client, where $z_i=x^{d_i}(\bmod n)$.
- The client obtains M by computing M as follows: $M_p=\prod_{i=1}^{k} z_i^{f_i}(\bmod p)$ and $M_q=\prod_{i=1}^{k} z_i^{g_i}(\bmod q)$

One then combines the M_i's using the CRT to obtain M.

This paper adopts RSA-S2 to speed up the decryption by re-assigning the roles: encryption becomes "server" in RSA-S2 parlance and decryption becomes "weak client". The main idea is to shift some computational burden from the decryption to the encryption. The resultant technique is called Encrypt Assistant RSA (EARSA) in this paper.

978-1-4244-5517-1/10/$26.00 ©2010 IEEE

B. EAMRSA

The new variant is described as cryptosystem with three algorithms: key generation, encryption and decryption.

Key generation: The key generation algorithm takes a security parameter and three additional parameters b, k and c as input. The key pairs (public and private) are generated according to the following steps:

- Compute b distinct primes $p_1, \cdots, p_b$ each one $\lfloor n/b \rfloor$ bits in length and generate $N = \prod_{i=1}^{b} p_i$.
- Compute $d = e^{-1} \bmod \varphi(N)$, where e picks the same e used in standard RSA [1] public keys, namely e=65537. The e is relatively prime to $\varphi(N) = \prod_{i=1}^{b}(p_i - 1)$.
- Compute $r_i = d \bmod p_i - 1$, for $1 \le i \le b$. Represent the r_i as follows:

$r_1 = d_{1,1} \cdot e_{1,1} + d_{1,2} \cdot e_{1,2} + ... + d_{1,k} \cdot e_{1,k} (\bmod p_1 - 1)$, ……,

$r_i = d_{i,1} \cdot e_{i,1} + d_{i,2} \cdot e_{i,2} + ... + d_{i,k} \cdot e_{i,k} (\bmod p_i - 1)$, ……,

$r_b = d_{b,1} \cdot e_{b,1} + d_{b,2} \cdot e_{b,2} + ... + d_{b,k} \cdot e_{b,k} (\bmod p_b - 1)$,

where the $d_{i,j}$'s and $e_{i,j}$'s, for $1 \le i \le b$ and $1 \le j \le k$, are random vector elements of c and |n| bits, respectively. The choice and security of parameters: b, k, and c are discussed later. The public key is $< N, e, e_{1,1}, ..., e_{1,k}, ..., e_{b,1}, ..., e_{b,k} >$ and the private key is $< N, d_{1,1}, ..., d_{1,k}, ..., d_{b,1}, ..., d_{b,k} >$.

Encryption: The encryption of the new variant RSA includes two steps:

- Given plaintext message $M \in Z_N$, encrypt M as basic RSA. The ciphertext C is computed as $C \leftarrow M^e \bmod N$.
- Compute vector $Z = (z_{1,1}, ..., z_{1,k}, ..., z_{b,1}, ..., z_{b,k})$ and take C as input, where $Z_{i,j} = C^{e_{i,j}} \bmod N$, for $1 \le i \le b, 1 \le j \le k$, and sends it to the decryption part. The ciphertext message is the vector Z.

Decryption: Decryption is done using the Chinese Remainder Theorem. To decrypt the vector Z and execute as the following steps:

- Calculate M_i by the $M_i = \prod_{j=1}^{k} Z_{i,j}^{d_{i,j}} (\bmod p_i)$, for $1 \le i \le b$, where p_i has been computed.
- According to the Chinese Remainder Theorem, Compute $y_i = N / p_i = p_1 p_2 \cdots p_{i-1} p_{i+1} \cdots p_b$ and $n_i = y_i \times (y_i^{-1} (\bmod p_i))$, for each i, $1 \le i \le b$.
- Using the CRT to combine the M_i 's to obtain $M = C^d = M_1 \times n_1 + \cdots + M_i \times n_i (\bmod N)$, for each $1 \le i \le b$.

The variant improves the performance by evaluating a larger number of exponentiations with reduced modules and private exponents.

C. Security and Parameter Selection

It is clear that the security of EAMRSA depends on the size of the primes used and on the security offered by the many private exponents $d_{i,j}$, for $1 \le i \le b$ and $1 \le j \le k$.

Firstly, the size of the primes used is discussed. The security of EAMRSA depends on the difficulty of factoring integers of the form $N = \prod_{i=1}^{b} p_i$ for b>2.The fastest known factoring algorithm cannot take advantage of this special structure of N [4].However, one has to make sure that the prime factors of N do not fall within the range of the Elliptic Curve Method [8], because 256-bit factors would be within the range of RSA-512 factoring project, using Elliptic Curve Method[4].Consequently, with a bit length of 1024 modules, it is not secure anymore to use a decomposition of more than three primes. When the value of b parameter is chosen in the new variant, one should make sure that the prime factors of N are bigger than 256-bit. In the following experiments, the key sizes of tested algorithms are from 1024- to 3072- bit, and b chooses 3 or 4 to make the prime factors of N bigger than 341-bit. So the security level of the new variant can be greatly enhanced.

To guarantee the security of EAMRSA, one also has to require for the variant to be infeasible to deduce values $r_i = d \bmod p_i - 1$, $1 \le i \le b$ via brute force, since an attacker with knowledge of some one would be able to factor modulus N and thereby break RSA. Given a vector, for each i , $1 \le i \le b$, $< N, e, e_{1,1}, ..., e_{b,k} >$,a search through all possible values of $d_{i,k}$ would reveal r_i .Because there are $b \times k$ c-bit vector elements, one has to make sure that the search space of $2^{b \times k \times c}$ values is large enough to prevent such an exhaustive search. When the values of the parameters c, k, and b are chosen, one has to make the difficulty of exhausting the resulting search space at least equivalent (or harder than) to breaking the underlying RSA cryptosystem. Exhaustive search of $2^{b \times k \times c}$ values is equivalent to searching for all possible keys in a symmetric-key cryptosystem. Thus, based on the RSA key size used, one has to make sure that the symmetric key size can provide equivalent security. Lenstra and Verheul give formulas for determining such keys [7]. RSA with 1024- and 1536-bit keys by the above formulas would be roughly equal in strength to a symmetric-key cryptosystem with 72- and 80-bit keys, respectively. The EAMRSA values c, b, and k were selected based on the key size formulas in [7], and $b \times k \times c$ corresponds to a symmetric key comparable in strength to the corresponding RSA key. In the following experiments, the key size of tested algorithms is from 1024- to 3072-bit, parameter b chooses 3 or 4, parameter c chooses 128- or 256-bit, and k=2, making the value of $b \times k \times c$ much larger than the value based on key size formulas in [7] .Thus, the security level of the private keys of the new variant is greatly enhanced by the above method. A conclusion is reached, that is, the variant can provide services of high security and high speed.

D. Parallelism analysis

As the multi-core computers are getting into the market, more and more computers have the parallel environment, but most of the current programs are not the parallel programs. Parallel programs attempt to solve bigger problems in less time by simultaneously solving different parts of the problem on different processing elements. However, this may not work, unless the problem contains exploitable concurrency, that is, multiple activities or tasks can execute at the same time. The key to parallel computing is the exploitable concurrency. Concurrency which exists in a computational problem can be decomposed into sub problems that can safely execute at the same time. The first step in designing a parallel algorithm is to decompose the problem into elements that can execute concurrently. This decomposition can be considered as occurring in two dimensions that are the data-decomposition and the task-decomposition [6].The new variant of the RSA will be analyzed in two dimensions so as to find the exploitable concurrency of the variant.

The data-decomposition dimension focuses on the data required by the tasks and on how the data can be decomposed into distinct chunks [6]. The main data of the variant are public key, private key, plaintext and ciphertext. Compared with the standard RSA, the public key of the new variant RSA becomes $< N, e, e_{1,1}, ..., e_{1,k}, ..., e_{b,1}, ..., e_{b,k} >$ .the private key becomes $< N, d_{1,1}, ..., d_{1,k}, ..., d_{b,1}, ..., d_{b,k} >$ .The public key of standard RSA is decomposed to many elements and the private exponent is also decomposed to many small numbers. The public key and private key are decomposed into a $b \times k$ matrix. Every unit in the matrix is a pair of the public key and private key and is taken as the input of encryption and decryption and these data units can be operated relatively independently. Because each data unit of the variant is independent, it is possible to parallelize the application by associating each unit with a task.

The task–decomposition views the problem as a stream of instructions that can be broken into sequence called task that can execute simultaneously [6]. The main tasks of the RSA focus on encryption and decryption. The operation of the decryption of the standard RSA only includes a modular exponentiation and cannot be broken into any tasks. The operations of the standard RSA employing the Chinese Remainder Theorem for RSA decryption are two modular exponentiations and are not easy to break and extend in the parallel environment. The operation of the encryption is only a modular exponentiation in the standard RSA and cannot execute in parallel. However, the new variant of RSA includes $b \times k + 1$ modular exponentiations in the encryption and $b \times k$ modular exponentiations in the decryption. The big modular exponentiation of standard RSA is decomposed into many smaller modular exponentiations in the variant. The encryption and decryption of the variant can be broken into multiple tasks that can execute simultaneously. These tasks executed simultaneously must be based on the independent input data. The independency of input data is analyzed in the data-decomposition.

Figure 1 shows that there are lots of exploitable concurrency in the encryption and decryption of the variant. Thus, based on the analysis of the data-decomposition and task-decomposition, the variant can be efficiently implemented in parallel. Figure 1 shows that encryption and decryption of the variant can be paralleled by row or by column. The parallel implementation is agile.

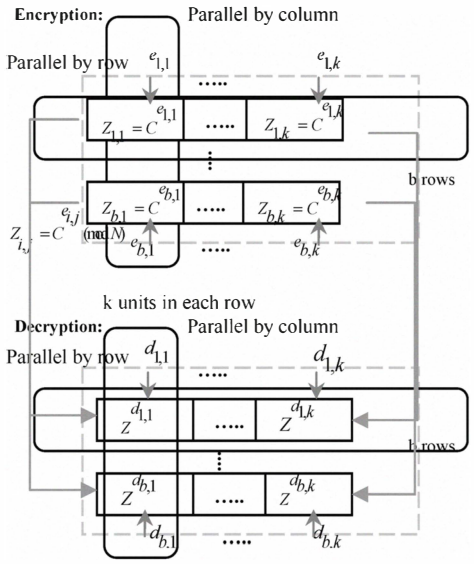

Figure 1. Task Decomposition of Decryption and Encryption.

For the parallel implementation of the new variant, the multi-core computers can be chosen as the parallel hardware platform and the software development platform of parallel programs can choose the OpenSSL [9] cryptographic library and OpenMP [6].

III. PERFORMANCE AND EXPERIMANTAL RESULTS

The speedup is measured by the execution time of RSA decryption in the paper. These test algorithms were written by using the OpenSSL cryptographic library (version 0.9.8 k) and OpenMP. The hardware platform was a 1.5GHz Intel(R) Celeron(R) M processor with 512 MB RAM running Windows XP Professional. In the experiment, the algorithms were called different names according to different parameters and values of the parameters. EA1M3RSA,EA1M4RSA,EA2M3RSA and EA2M4RSA in the paper respectively indicate the EAMRSA of the parameters b=3 and c=128-bit,the parameters b=4 and c=128-bit,the parameters b=3 and c=256-bit and the parameters b=4 and c=256-bit. M3RSA and M4RSA respectively indicate the Multi-Prime RSA of the parameter b=3 and the parameter b=4. The EA1RSA and EA2RSA respectively indicate the EARSA of the parameter b=3 and the parameter b=4. The RSA keys of 1024, 1536, 2048, 2304, 2560, 2816, and 3072 bits were used so as to test EAMRSA performance with both current and future security.

TABLE I. SPEEDUP RELATED TO DECRYPTION

Variant	Speedup					
	2048	*2304*	*2560*	*2816*	*3072*	*Average*
EA1RSA	3.86	4.88	3.41	4.38	5.38	4.14
EA2RSA	1.94	2.44	3.41	2.98	2.73	2.74
M3RSA	2.00	2.00	2.32	2.00	2.00	2.21
M4RSA	2.00	2.44	3.41	3.04	3.66	2.92
EA1M3RSA	4.13	5.20	7.27	5.83	5.55	5.19
EA2M3RSA	3.88	2.52	3.52	3.59	3.66	3.38
EA1M4RSA	4.13	5.20	6.81	8.75	10.75	7.06
EA2M4RSA	3.88	3.39	4.54	4.52	5.55	4.63

Table 1 shows the speedups for the five moduli with eight variants to the standard RSA. The EA1M4RSA got the highest speedup in all variants. The result shows the average EA1M4RSA speedup of 7.06 is from 2048- to 3072-bit.Table 2 lists the improved speedup among the EAMRSA, the EARSA and the Multi-Prime RSA. The results show that the speedup of every algorithm of the EAMRSA is improved to the relative EARSA and Multi-Prime RSA. The highest improved speedup is 4.14 for the EA1M4RSA to the M4RSA and the least improved speedup is 0.64 for the EA2M3RSA to the EA2RSA in table 2.

TABLE II. IMPROVED SPEEDUP RELATED TO DECRYPTION

Variant	Improved Speedup (Average)			
	EA1RSA	*EA2RSA*	*M3RSA*	*M4RSA*
EA1M3RSA	1.05		2.98	
EA2M3RSA		0.64	1.17	
EA1M4RSA	2.92			4.14
EA2M4RSA		1.89		1.71

Figure 2 shows that the process of the EAMRSA encryption in parallel obtains the speedup to the original EAMRSA encryption. The results show that when the key becomes larger, the parallel results will be better.

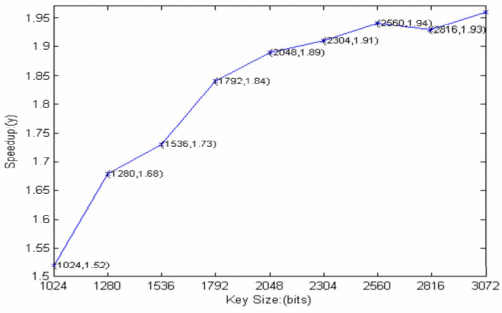

Figure 2. Parallel encryption speedup when varying key size -For k=2

The test hardware platform of the parallel EMRSA encryption was a 1.73GHz Intel Pentium Dual Core with 1 GB RAM running Windows XP Professional. From figure 2, based on double core computers, the speedup of factor 2 which parallel encryption has obtained is the best. The experimental result shows that the speed of the decryption has been substantially improved and the variant can be efficiently implemented in parallel.

IV. DISCUSSION

There are some characteristics for the EAMRSA:

1. High performance in decryption and signature generation.
2. Performance increases with larger moduli, for k is fixed.
3. High security: The private exponents of the EAMRSA, if c=128, k=2, and b=4, can obtain 8×128 -bit strength.

Although EAMRSA reduces the computation load at the decryption, it introduces certain computation cost to the encryption. This added computation cost is negligible and acceptable for most encryption and if the encryption executes on the multi-core computers, the process of the encryption can be implemented in parallel, so the added computation can be well solved.

ACKNOWLEDGMENT

This work has been supported by the National Science Foundation of China under Grant No. 60963007 and by the Middle-aged Backbone Teacher Training Program of Yunnan University No.2113204.

V. CONCLUSION

In this paper, the new RSA variant which can improve the performance of the decryption and signature generation was proposed. The variant can obtain high performance by reducing the modulus and private exponents. The improved variant can be easily implemented in parallel and can get higher security and higher speedup based on current multi-core devices. The next study will focus on: How to combine OpenSSL cryptographic libraries and OpenMP more efficiently and implement the parallel RSA system in the multi-core platform and how to optimize the parameters of the new variant to make the variant get higher performance and security.

REFERENCES

[1] R. Rivest,A. Shamir, L. Aldeman, "A Methoed for Obtaining DigitalSignatures and Public-key Cryptosystems,"J. Communications of the ACM, 1978, 21(2): 120-126.

[2] V .Paxson, R .Sommer,"An architecture for exploiting multi-core processors to parallelize network intrusion prevention,"C.a Proceedings of the IEEE Sarnoff Symposium. 2007. 1-7.

[3] T. Collins, D. Hopkins, and M. Sabin, "Public Key Cryptographic Apparatus and Method,"US Patent #5,848,159. Jan. 1997.

[4] D.Boneh,H.Shacham, "Fast Variants of RSA,"R.RSA Laboratories Cryptobytes,2002,5(1):1-8.

[5] T.Matsumoto,K.Kato, "Speeding up secret computations with insecure auxiliary device,"C.Proc of the 8th Annual International Crypto Conference on Advances in Cryptology.London:Springer-Verlag,1988.

[6] G.Timothy,A.Beverly,"Patterns for Parallel Programming,"M. Addison-Wesley Professional.2005.11.

[7] A. K. Lenstra and E. R. Verheul, " Selecting cryptographic key sizes,"J. the journal of the International Association for Cryptologic Research, vol. 14, no. 4, pp. 255-293, 2001.

[8] R. Silverman and S. Wagstaff Jr, " A Practical Analysis of the Elliptic Curve Factoring Algorithm,"M. Comp. 61(203):445–462. Jul. 1993.

[9] J. Viega, M. Messier and P. Chandra, "Network Security with OpenSSL,",M.O'Reilly,2002.