2009 IEEE International Advance Computing Conference (IACC 2009)
Patiala, India, 6–7 March 2009

Improvement over Public Key Cryptographic Algorithm

Deepak Garg, Seema Verma
Thapar University

Abstract:
In this paper, we have introduced RSA cryptosystem and its improvements. There are many cases when there is the need to enhance the decryption/signature generation speed at the cost of encryption/signature verification speed, e.g., in banks, signature generation can be in huge amount in a single day as compared to only one signature verification in the complete day at receiver side. So here in this paper the main stress is on the improvement of decryption/signature generation cost. Many methods are discussed to improve the same, e.g., Batch RSA, MultiPrime RSA, MultiPower RSA, Rebalanced RSA, RPrime RSA. The proposed approach to improve decryption/signature generation speed is given in the paper. We have tried the improvement by the combination of MultiPower RSA and Rebalanced RSA. Theoretically, the proposed scheme (for key length 2048 bits moduli) is about 14 times faster than that given by RSA with CRT and about 56 times faster than the standard RSA. Tabular and graphical comparison with other variants of RSA is also shown in the paper.

Keywords: ciphertext, encryption, DES

1. Introduction

In the current time, when the Internet provides essential communication between millions of people and is being increasingly used as a tool for commerce, security becomes a tremendously important issue to deal with. There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting passwords. One essential aspect for secure communications is that of cryptography. Cryptography is the science of writing in secret code and is an ancient art. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet.

There are, in general, two types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography. The initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn be decrypted into usable plaintext. With secret key cryptography, a single key is used for both encryption and decryption, e.g., Data Encryption Standard (DES) [1] and Advanced Encryption Standards (AES) [2]. The biggest difficulty with this approach, of course, is the distribution of the key.

Public-key cryptography has been said to be the most significant development in cryptography in the last 300-400 years. In this scheme, a two-key cryptosystem is used in which two parties could engage in a secure communication over a non-secure communication channel without having to share a secret key, e.g., RSA[3], ECC[4] etc.

Here in this paper our main motive is to improve the efficiency of RSA cryptosystem. The first section of the paper describes brief review of RSA Cryptosystem. In Second section various methods for the improvement over standard RSA is reviewed. Then our proposed scheme is shown with the result analysis. The paper is then concluded in the last section.

2. RSA

In cryptography, RSA [3] is a public key algorithm. The algorithm was publicly described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman at MIT. It is the first algorithm known to be suitable for encryption as well as signing, and one of the first great advances in public key cryptography. RSA is widely used in electronic commerce protocols. Given sufficiently long keys and the use of up-to-date implementations; RSA is believed to be totally secure. The concept behind public-key cryptography is to utilize an asymmetric cryptography technique with two keys, one public and one private. The keys are derived from the product of two large prime numbers. The private key can only be deduced from the public key by factoring the large multiple. RSA security depends upon the difficulty of factoring very large numbers. No efficient factoring algorithm has been yet designed which can challenge the security of RSA cryptosystem. Though the techniques for factoring numbers are improving, but the speed depends on the size of the prime numbers, which means they still take significant time. Ofcourse, the possibility is there that one day there will be an extraordinary leap in our ability to factor large numbers, but it is unlikely and offers a minimal threat to RSA, because the improvement over standard RSA is gradually improving day by day.

2.1 Working of RSA

In RSA algorithm [3], each user of the system makes two numbers, e and N (n bits) public and keeps a number d secret. In order for A to send a message to B, A looks up B's public values and, if the message is M (written as a number), then user A makes blocks of the message of size < N and sends $C = M^e \bmod n$. Then receiver B decrypts the text by $M = C^d \bmod N$. The algorithm security depends on the choice of the public and private keys. These must be sufficiently large.

Key Generation of RSA

Generate two large random primes of n/2 bits each, p and q, of approximately equal size such that their product N = pq is of the required bit length, e.g. 1024 bits

Compute $N = pq$ and (φ) phi = $(p-1)(q-1)$.

Choose an integer e, $1 < e < \text{phi}$, such that $\gcd(e, \varphi) = 1$

Compute the secret exponent d, $1 < d < \square$, such that $ed \equiv 1 \pmod{\square}$.

The public key is (N, e) and the private key is (N, d). Keep all the values d, p, q and □ secret.

Encryption of RSA

Sender A does the following:-

Obtains the recipient B's public key (N, e).

Represents the plaintext message as a positive integer M.

Computes the ciphertext $C=M^e \mathrm{mod} N$.

Sends the ciphertext C to B.

Decryption of RSA

Recipient B does the following:-

Uses his private key (N, d) to compute $M = C^d \bmod N$.

Extracts the plaintext from the message representative M.

Here we are concerned with decryption speed of the algorithm. Factors N and d are involved in the computation. Both are large numbers (approx. n bits long). This algorithm takes decryption time as follows: $O(n^3)$.

2.2 Security of RSA Algorithm

The security of the RSA cryptosystem [3] is based on two mathematical problems: the problem of factoring the numbers and the RSA problem. Till now, no efficient algorithm exists for full decryption by the adversary. To avoid partial decryption addition of a secure padding scheme is required.

RSA problem

The RSA problem is defined as the task of taking e^{th} roots modulo a composite n, recovering a value m such that $C=M^e \bmod N$, where (N, e) is an RSA public key and C is an RSA ciphertext. In other words, the problem is to perform the RSA private-key operation given only the public key. A fast means of solving the RSA problem would yield a method for breaking all RSA-based public-key encryption and signing systems. It has long been known that finding the private RSA exponent d is equivalent to factoring N, and that finding square roots modulo N is as hard as factoring N.

The RSA Integer Factoring

Currently the most promising approach to solving the RSA problem is to factor the modulus N. With the ability to recover prime factors, an attacker can compute the secret exponent d from a public key (N, e), then decrypt ciphertext using the standard algorithm. Hence not been proven, but no polynomial-time method for factoring large integers on a classical computer has yet been found. Using software already freely available, with N value 256 bits or shorter, it can be factored in a few hours on a personal computer. Peter Shor, in 1994, published Shor's algorithm [5], showing that a quantum computer could in principle perform the factorization in polynomial time. However, quantum computation is still in its infacy till date and may never prove to be practical. A theoretical hardware device described by Shamir and Tromer in 2003, named TWIRL [6] challenge the security of 1024 bit keys. That's why key of length at least 2048 bits is recommended after that. As of 2005, the largest number factored by a general-purpose factoring algorithm was 663 bits long. It is generally presumed that RSA is secure if N is sufficiently large of the range of 2048 bits or greater.

2.3 Attacks on RSA

Some of the attacks on RSA cryptosystem are discussed here:

Elementary Attacks

These attacks [7] illustrate overt misuse of RSA. There are many such attacks in existence, we have mentioned only of two types here. First is Common Modulus attack, the user may avoid generating a different modulus N = pq for each user in a group, he may wish to fix N once and use the same for all user in the group. The same N is used by all users. A trusted central authority could provide user with a unique pair e, d from which the user forms a public key <N, e> and a secret key <N, d>. The sender b can use his own exponents' e_b, d_b to factor the modulus N. Once N is factored he can recover others private key d, from her public key e. it is clear from this observation that an RSA modulus should never be used by more than one entity. The second example of elementary attack is blinding attack. In this the user blindly signs the data. By this we mean that the adversary takes the signature of the user for malicious data by showing it as innocent by cheating.

Low Private Exponent

To reduce decryption/signature generation time, one may wish to use a small value of d rather than a random d. Since modular exponentiation takes time linear in $\log_2 d$, a small d can improve performance by at least a factor of 10 (for a 1024 bit modulus). Since typically N is 1024 bits, it follows that d must be at least 256 bits long in order to avoid this attack. This is difficult for low-power devices such as smartcards, where a small d would result in big savings. Therefore, one might be tempted to use small secret exponents to speed up the decryption/signing process. Unfortunately, Wiener [8] showed in 1991 that if $d < N^{1/4}$ then the factorization of N can be found in polynomial time using only the public information (N, e). In 1999, Boneh and Durfee [9] improved the bound to $d < N^{0.292}$.

Low Public Exponent

To reduce encryption or signature-verification time, it is customary to use a small public exponent e. The smallest possible value for e is 3, but to avoid certain attacks the value e = 216 +1 = 65537 is recommended. When the value 216 +1 is used, signature verification requires very much less computation as compared to when a random e is used. Attacks due to small value of e are far from a total break. Examples of attacks due to low value of e are coppersmith attack [10], Hastad's Broadcast attack [11], Franklin-Reiter related message attack [12], and partial key exposure attack [13].

Timing Attacks

In 1996, Kocher [14] shows this type of attack. Consider a smartcard that stores a private RSA key. Since the card is tamper resistant, an attacker may not be able to examine its contents and expose the key. Kocher showed that by precisely measuring the time it takes the smartcard to perform an RSA decryption (or signature generation), attacker can quickly discover the private decryption exponent d.

Adaptive chosen ciphertext attacks

In 1998, Daniel Bleichenbacher [15] described this type of attack against RSA-encrypted messages using PKCS#1 v1 padding scheme. Due to flaws with PKCS#1 scheme, Bleichenbacher was able to mount a practical attack against RSA implementation of the secure socket layer protocol and to recover session keys.

3. Improvement over standard RSA

RSA is much slower than DES and other symmetric cryptosystems. In practice, sender typically encrypts a secret message with a symmetric algorithm, encrypts the (comparatively short) symmetric key with RSA, and transmits both the RSA-encrypted symmetric key and the symmetrically-encrypted message to Alice.

Lots of research has been done to enhance the speed of RSA cryptosystem. Here are some approaches which are studied to enhance the decryption speed of RSA.

3.1 RSA with CRT

In 1982, J.J. Quisquater and C. Couvreur [16] introduced a new technique to increase the speed of decryption algorithm of RSA cryptosystem. In this technique two smaller secret keys (d_p, d_q) are calculated from the original secret key(d), decryption is done with these two keys and the result is combined with the help of Chinese Remainder Theorem(CRT). It improves the performance of the basic RSA decryption algorithm by 4.

Key Generation of RSA with CRT

Same as in basic RSA

Encryption of RSA with CRT

Same as in basic RSA

Decryption of RSA with CRT

Calculate d_p=d mod p-1 and d_q=d mod q-1

$M_p=C^{dp}$(mod p)

$M_q=C^{dq}$ (mod q)

Calculate M from M_p & M_q using Chinese Remainder Theorem(CRT)

Performance of RSA with CRT

Theoretical speedup of this method with relation to the original RSA is:

$S_{RSA}= (n^3)/ 2(n/2)^3 = 4$

3.2 Batch RSA

In Batch RSA variant [17] in 1989, if small public exponents are used for the same modulus N, the decryption of the two ciphertext can be done at the cost of one. Suppose C1 and C2 are the two ciphertext from message M1 and M2 respectively. There public keys are <N, 3> and <N, 5> respectively. To decrypt $C_1^{1/3}$modN and $C_2^{1/5}$modN are calculated. According to this variant both decryption processes can be merged to enhance the speed of the decryption algorithm. The scheme can be understood by setting $A= (C_1^5.C_2^3)^{1/15}$

Then, $C_1^{1/3}= A^{10}/ [C_1^3.C_2^2]$

And $C_2^{1/5}= A^6/ [C_1^2.C_2]$

Hence we are able to decrypt both C_1 and C_2 at the cost of computing 15th root (takes the same time as a single RSA decryption) and some additional arithmetic.

But this technique is only valuable when the public exponents' e_1 and e_2 are small. Otherwise it will not increase the decryption speed; rather it will be more expensive. Also the modulus must be the same and public exponents must be distinct for both the messages.

The generalization of this method is given for b RSA ciphertexts. For the implementation of the technique, b relatively prime public keys e1, e2,.....eb sharing common modulus N are used. This technique describes b-batch process using a binary tree. For each i one computes M_i as:

$M_i=C_i^{\ 1/ei}=A^{\alpha i}/ [C_i^{(\alpha i-1)/ei}.\prod_{j\neq i} C_j^{\ \alpha i/ej}$

Where α_i=1 mod e_i

And α_i=0 mod e_j (for $j\neq i$)

Here since b and e_i's are small, the exponents in this equation are also small.

Performance of Batch RSA

Batch RSA decrypts simultaneously b messages with the approximate cost of a single exponentiation (of order of N) and some small exponentiations (using public exponents). According to Fiat [17] with standard 1024-bit keys, batching improves performance significantly. With b=4, RSA decryption is enhanced by a factor of 2.6, with b=8, by a factor of almost 3.5.

3.3 MultiPrime RSA

In this variant [18] in 1998 the RSA modulus was modified so that it consists of k primes $p_1, p_2 \ldots p_k$ instead of using only two. The algorithms of Key generation, Encryption and Decryption are described as:

Key generation of MultiPrime RSA

The key generation algorithm receives as parameter the integer k, indicating the number of primes to be used. The key pairs (public and private) are generated according to the following steps:

1. Compute k distinct primes p1... pk each one [logN/k] bits in length and $N=\prod_{i=1}^{k} p_i$.
2. Compute e and d such that $d = e^{-1}$ mod □ (N), where gcd(e, □(N)) = 1. □(N)$=\prod_{i=1}^{k} (p_i-1)$
3. For $1\leq i\leq k$, compute d_i = d mod (p_i - 1).

Public key = <N, e>, Private key = <N, d_1, d_2 ...dk>

Encryption of MultiPrime RSA

Given a public key <N, e>, encrypt message M exactly as in the original RSA, thus $C = M^e$ mod N.

Decryption of MultiPrime RSA

The decryption is an extension of the Quisquater-Couvreur method. To decrypt a ciphertext C, first calculate $M_i = C^{di}$ mod p_i for each i, $1\leq i\leq k$. Next, apply the CRT to the M_i's to get $M = C^d$ mod N.

Performance of MultiPrime RSA

The theoretical speedup of this variant to the CRT RSA is given as follows:

$S_{CRT}= (2. (n/2)^3)/ (k. (n/k)^3)= k^2/4$

3.4 MultiPower RSA

In this variant [19], N= $p^{k-1}q$ where p and q are n/k bits. The three algorithms are described as below:

Key generation of MultiPower RSA

Generate two primes p and q of [n/k]-bits each and compute N= p^{k-1}.q

Compute $d=e^{-1}$mod(p-1)(q-1)

Compute d1=d mod (p-1) and d2=d mod (q-1).

<N, e> is the public key and <p, q, d1, d2> is the private key.

Encryption of MultiPower RSA :

Same as in standard RSA.

Decryption of MultiPower RSA:

Compute $M_1= C^{d1} \bmod p$ and $M_2=C^{d2} \bmod q$

Thus $M_1^e=C \bmod p$ and $M_2^e=C \bmod q$.

Using Hensel lifting[20], compute M_1', such that $(M_1')^e=C \bmod p^{k-1}$

Using CRT, compute M such that $M= M_1' \bmod p^{k-1}$ and $M= M_2 \bmod q$. Then $M=C^d \bmod N$ is the required result

Performance of MultiPower RSA:

Decryption takes two full exponentiation modulo (n/k)-bit numbers and k-2 Hensel lifting [20]. Hensel lifting is much faster than exponentiation. So the speedup over standard RSA (with CRT) is approximately:

$S_{CRT}= (2. (n/2)^3)/ (2. (n/k)^3)= k^3/8$

For k=3 the theoretical speedup is about 3.38

3.5 Rebalanced RSA

In some applications, user would like to have the reverse behavior of the standard RSA, i.e. fast signature generation/decryption time, e.g., when a cell phone needs to generate an RSA signature that will later be verified on a fast server, one would like signing to be easier than verifying. Rebalanced RSA [21] fulfills this requirement by improving the performance of the decryption/signing algorithm by displacing the work to the encryption/verification algorithm. Because of the security reasons one can't choose small value of d. The values less than $N^{0.292}$ for d are insecure. So d is chosen as a large number, say of the order of N. but d mod p-1 and d mod q-1 are small numbers. Three algorithms of key generation, encryption and decryption are as follows:

Key generation of Rebalanced RSA

Take s<=n/2 bits

1. Generate two distinct random (n/2)-bit prime numbers p & q with gcd(p-1,q-1)=2, and calculate N=pq.
2. Generate two s-bits random numbers d_p and d_q, such that $gcd(d_p,p-1)=1$, $gcd(d_q,q-1)=1$ and $d_p=d_q \bmod 2$.
3. Calculate one d such that $d= d_p \bmod p-1$ and $d=d_q \bmod q-1$
4. Calculate $e=d^{-1} \bmod$ □(N)

Public key =<N,e>

Private key=$<p,q,d_p,d_q>$

Encryption of Rebalanced RSA:

Same as in Standard RSA but with higher computation cost (e is large)

Decryption of Rebalanced RSA:

Same as in RSA with CRT.

Performance of Rebalanced RSA:

Speedup over standard RSA with CRT is:

$S_{CRT}=n/2s$

For s= 160, the theoretical speedup is 6.4 times than RSA with CRT.

Rebalanced approach makes RSA encryption very time-consuming because the public exponent e in Rebalanced RSA-CRT will be of the same order of magnitude as □ (N). Improvement over this was done by Galbraith [22] and Sun [23] to further decrease the public exponent e that is much shorter than □ (N). It is further improved by Hinek in [24].

3.6 RPrime RSA

In 2002 Cesar Alison [25] combined the two RSA variants Rebalanced RSA and Mprime RSA methods to further enhance the decryption speed. The general idea of this scheme is to use the key generation algorithm of Rebalanced RSA (modified for k primes) together with the decryption algorithm of Mprime RSA. The key generation, encryption and decryption algorithms are as follows:

Key generation of RPrime RSA

The key generation algorithm takes an integer s□n/k and executes the following steps:

Generate k distinct random primes of n/k bits p_1, $p_2..., p_k$, with $gcd(p_1 - 1, p_2 - 1, ..., p_k - 1) = 2$; and calculate $N=p_1p_2...p_k$.

Generate k random numbers of s-bits d_{p1} , d_{p2} ..., d_{pk} , such that $gcd(d_{p1} , p_1 - 1) = 1$, $gcd(d_{p2} , p_2 - 1) = 1$, ..., $gcd(d_{pk} , p_k - 1) = 1$ and $d_{p1} = d_{p2} = ... = d_{pk} \bmod 2$.

Find d such that $d = d_{p1} \bmod (p_1 - 1)$, $d = d_{p2} \bmod (p_2 -1)...d = d_{pk} \bmod (p_k - 1)$

Calculate $e =d^{-1} \bmod$ □(N).

Public key = <N, e> and Private key = $<p_1, p_2... p_k, d_{p1}, d_{p2} ...d_{pk}>$

Encryption of RPrime RSA

Again, encrypting with the public key <N, e> is identical to the original RSA. However, that as in Rebalanced RSA, the public exponent e is much larger than the normally used e, and thus, the entity encrypting the message M must be prepared to use such an exponent.

Decryption of RPrime RSA

Same as in MultiPrime RSA decryption

Performance of RPrime RSA

The theoretical speedup (SQC), is therefore given by:

$S_{CRT} = [nk]/4s$

Comparison of all above variants is given in [26].

4. The New Proposed Scheme

In our proposed scheme, we are trying to enhance the speed of RSA decryption by the combination of Rebalanced RSA with MultiPower RSA. The general idea of this scheme is to use the key generation algorithm of Rebalanced RSA with only two primes of n/k bits length and decryption algorithm of MultiPower RSA. The three algorithms for the new scheme are as follows:

Key Generation of new Scheme

Generate two distinct [n/k]-bit primes , p and q, with gcd(p-1,q-1)=2, and calculate $N=p^{k-1}.q$

Generate 2 random numbers d_p and d_q, of s-bits(s<=n/k) such that $gcd(d_p,p-1)=1$, $gcd(d_q,q-1)=1$ and $d_p= d_q \bmod 2$.

Find d such that $d= d_p \bmod p-1$, $d= d_q \bmod q-1$

Calculate $e=d^{-1} \bmod$ □(N)

Public key=<N,e> and Private Key=$<p,q, d_p,d_q>$

Encryption of new Scheme:

Same as in standard RSA. But as we have shifted the cost from decryption side to encryption side, the value of public exponents will get increased. Due to the large value of e RSA encryption must be ready for the higher computational cost.

Decryption of new Scheme:

Compute $M_1= C^{d1}$ mod p and $M_2=C^{d2}$ mod q
Thus $M_1^e=C$ mod p and $M_2^e=C$ mod q.
Using Hensel lifting [20] compute M_1', such that $(M_1')^e=C$ mod p^{k-1}
Using CRT, compute M such that M= M_1' mod p^{k-1} and M= M_2 mod q. Then $M=C^d$ mod N is the required result

Performance of new Scheme

The theoretical speedup of this scheme as with standard RSA with CRT is
$S_{CRT}=nk^2/8s$
For k=3 and s=160 and n=1024, the theoretical speedup over RSA with CRT is 7.20

4.1 Results

We will compare the new scheme with other variants, i.e., Batch RSA, MPrime RSA, MPower RSA, Rebalanced RSA and RPrime RSA. Table1 shows the theoretical comparison between standard RSA and RSA with CRT. RSA with CRT is ideally 4 times faster than the standard RSA.

In Table2 the comparison of all the discussed approaches to that of new approach is shown. All are compared to RSA with CRT. RSA with CRT is 4 times faster than standard RSA. Here we have taken k=3 (no of primes) and s=160. The table is showing results for different key length (different values of n).

Graphical comparison between all the approaches and the new approach is shown. Fig.1 shows the comparison of all the approaches for n=768bits (key length). Fig.2 shows the comparisons of all the approaches for n=1024bits.Fig.3 shows the comparison for n=2048bits

Table 1: Comparison of the RSA with CRT and standard RSA

Speedup	**Key length(n)**		
variant	**768**	**1024**	**2048**
Standard RSA	1	1	1
RSA with CRT	4	4	4

Table 2: Comparison of the new scheme with other variants

Speedup (over RSA with CRT)	**Key length(n)**		
variant	**768**	**1024**	**2048**
Batch RSA=b	b	b	b
MPrime RSA=$k^2/4$	2.25	2.25	2.25
MPower RSA=$k^3/8$	3.37	3.37	3.37
Rebalanced RSA=n/2s	2.40	3.20	6.40
RPrime RSA=nk/4s	3.60	4.80	9.60
New Scheme=$nk^2/8s$	5.40	7.20	14.40

In all the figures the value of k is taken as 3 (no of primes), and s as 160 bits. The batch size (no of messages) in all the figures is 2; ideally in batch variant the speed up is b times as that of standard RSA.

As the theoretical speedup of our scheme is very high as compared to other variant, we can say it will definitely be very effective when decryption cost of RSA is to be lowered. The new approach gives excellent performance in decryption. As shown in the comparison table the theoretical speed gain is about 14 times to that of RSA with CRT and because speed gain of RSA with CRT is 4 times to standard RSA, the new scheme gives the speed gain of approximately 56 times over standard RSA with key length 2048.

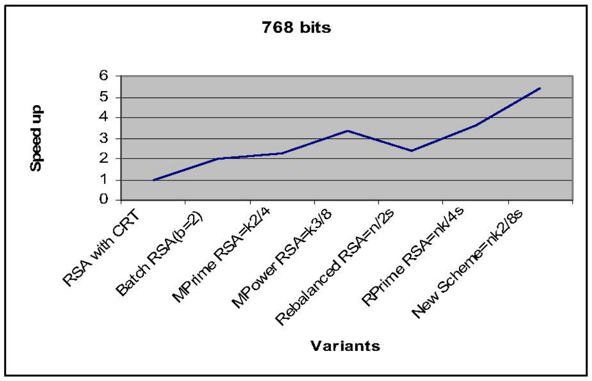

Fig.1 Comparison of RSA Variants with N=768bits

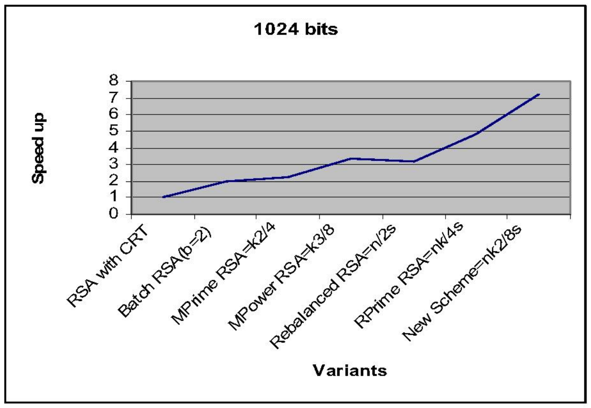

Fig.2 Comparison of RSA Variants with N=1024 bits

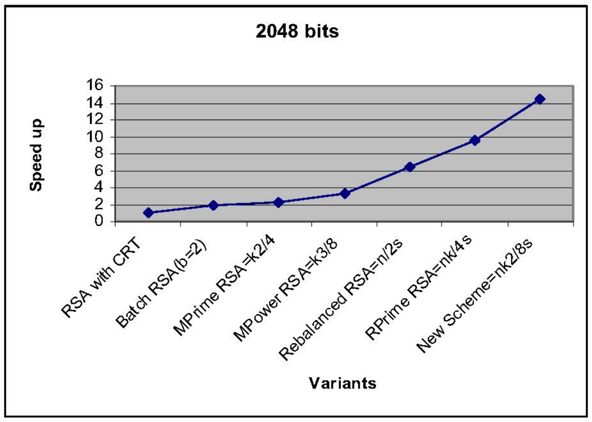

Fig.3 Comparison of RSA Variants with N=2048 bits

5. Conclusion

In this paper we have discussed RSA variants i.e., Batch RSA, MPrime RSA, MPower RSA, Rebalanced RSA and RPrime RSA. All these are focusing their attention on decreasing the decryption/signature generation time. We have also tried to enhance the decryption/signature generation speed by the combination of Rebalanced RSA and MPower RSA. By this combination we are achieving a very good speedup over standard RSA. Rebalanced, RPrime and our new proposed scheme are reducing the decryption/signature generation time but increasing the encryption/signature verification time. At first sight this cost shifting to encryption side does not seem to present advantages in practical terms. However, there are applications where the balancing characteristic of these algorithms is desirable. E.g., banks can emit many digital signatures in a single day, while the user that receives this signature, has usually much smaller burden. So efforts applied can be shifted to the receiver side. On the other hand in case of PDA users (having less resources) the efforts can be shifted to desktop user side (having more resources). So for the applications that prioritize the performance of decryption/signature generation, our proposed scheme will suit the best. It is offering approximately 14 times theoretical speedup over RSA with CRT and approximately 56 times theoretical speedup over standard RSA for key length of 2048 bits. We are working on the implementation of our proposed scheme.

References

[1] Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC (1977).

[2] J. Daemen and V. Rijmen, "Rijndael, the advanced encryption standard", Dr. Dobb's Journal, Vol. 26, No. 3, March 2001, pp. 137-139.

[3] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM Vol. 21, No. 2, pp. 120 - 126, 1978.

[4] Neal Koblitz, "Elliptic Curve Cryptosystems", Mathematics of Computation, Vol. 48, 1987, pp.203–209.

[5]] P.W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring", Proceedings, 35th Annual Symposium on the Foundations of Computer Science, 1994, pp. 124.

[6] Adi Shamir, Eran Tromer, "Factoring Large Numbers with the TWIRL Device", Proceedings, Crypto 2003, LNCS 2729, Springer-Verlag, 2003, pp.1-26.

[7] Dan Boneh, "Twenty Years of Attacks on the RSA Cryptosystem", Notices of the AMS, Vol. 46, No. 2, Feb. 1999, pp. 203-213.

[8] M. Wiener, "Cryptanalysis of Short RSA Secret Exponents", IEEE Transactions on Information Theory, Vol. 36, 1990, pp.553-558.

[9] D.Boneh, G.Durfee, "Cryptanalysis of RSA with private key $d<N^{.292}$", Proceedings of Eurocrypt'98, 1998, pp. 1-11.

[10] D. Coppersmith, "Small Solutions to Polynomial Equations and Low Exponent RSA Vulnerabilities", Journal of Cryptology, Vol. 10, 1997, pp.233-260.

[11] J. Hastad, "Solving Simultaneous Modular Equations of Low Degree", SIAM Journal of Computing, Vol. 17, No. 2, 1988, pp.336-341.

[12] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "Low Exponent RSA with Related Messages", Eurocrypt '96, Vol. 1070, 1996, pp. 1-9.

[13] D. Boneh, G. Durfee, and Y. Frankel, "An Attack on RSA Given a Fraction of the Private Key Bits", Proceedings, AsiaCrypt '98, Vol. 1514, 1998, pp. 25-34.

[14] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", Crypto '96, Vol. 1109, 1996, pp. 104-113.

[15] D. Bleichenbacher, "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1", CRYPTO '98, Vol. 1462, 1998, pp. 1-12.

[16] J-J. Quisquater and C. Couvreur, "Fast Decipherment Algorithm for RSA Public-Key Cryptosystem", Eletronic Letters, Vol. 18, 1982, pp. 905–907.

[17] A. Fiat. Batch RSA. "Advances in Cryptology", Crypto '89, Vol. 435, 1989, pp.175–185.

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

[19] T. Takagi, "Fast RSA-Type Cryptosystem Modulo p^kq", Crypto '98, 1462 of LNCS. 1998, pp. 318–326.

[20] H. Cohen, "A Course in Computational Algebraic Number Theory", Graduate Texts in Mathematics, Vol. 138, p. 137.

[21] M. Wiener, "Cryptanalysis of Short RSA Secret Exponents", IEEE Transactions on Information Theory, Vol. 36, No. 3, 1990, pp. 553–558.

[22] S. D. Galbraith, C. Heneghan, and J. F. McKee, "Tunable balancing of RSA," Proceedings Information Security and Privacy, 2005, Vol. 3574, pp. 280–292.

[23] H.-M. Sun and Mu-En. Wu, "Design of Rebalanced RSA-CRT for Fast Encryption," Information Security Conference 2005. pp. 16-27.

[24] M. J. Hinek, "Another look at small RSA exponents," Topics in Cryptology, CT-RSA 2006, 2006, Vol. 3860, pp. 82–98.

[25] C. A. M. Paixao, "An efficient variant of the RSA cryptosystem", preprints (2003).

[26] A.A. Mamun, M. M. Islam, S.M. M. Romman, A.H.S.U Ahmad, "Performance Evaluation of Several Efficient RSA Variants", IJCSNS VOL.8 No.7, July 2008, pp. 7-11.