An efficient variant of the RSA cryptosystem

Cesar Alison Monteiro Paixão*
capaixao@ime.usp.br

Institute of Mathematics and Statistics
University of São Paulo - Brasil

Abstract. We describe an efficient combination of two variants of RSA cryptosystem (MPrime and Rebalanced RSA) analysed by Boneh and Schacham [2]. The decryption process resultant is (for 2048-bits moduli) about 8 times faster than that presented by Quisquater and Couvreur [9] and about 27 times faster than original cryptosystem.

1 Introduction

In this article we present an extension of the work of Boneh and Shacham on some variants of the RSA cryptosystem. We review two of the four variants (Batch RSA, Mprime RSA, Mpower RSA, Rebalanced RSA) they analyse in [2], with the goal of reducing the decryption and signature generation times of the original cryptosystem. We briefly discuss the possibility and viability of combining such variants to obtain a new more efficient one. As a result, we describe a new variant that we call RPrime RSA, which combines Rebalanced RSA and MPrime RSA. For private key operations, this method is about 27 times faster than plain RSA and about 8 times faster than the method presented by Quisquater and Couvreur in [9].

This paper is organized as follows. In section 2 we review the RSA cryptosystem and the Quisquater-Couvreur method [9], in section 3 we describe Mprime and Rebalanced RSA, two variants of original RSA. In section 4 we present RPrime RSA, our proposed combination of MPrime and Rebalanced methods. Section 5, presents some theoretical and experimental results, and we conclude in section 6 with some comments on RPrime RSA.

2 RSA Cryptosystem

Before we present the proposals to optimize the RSA cryptosystem, we review the three basic algorithms that constitute the RSA, together with one frequently used optimization technique .

RSA - Key generation:

* Co-sponsored by Scopus Tecnologia S.A.

1 - Generate two primes p and q and compute their product N.

2 - Pick e such that $\gcd(e, \phi(N)) = 1$, where $\phi(n) = (p-1)(q-1)$.

3 - Compute d such that $d = e^{-1} \bmod \phi(N)$.

$\boxed{\text{Public Key} = \langle e, N \rangle}$ $\boxed{\text{Private Key} = \langle d, N \rangle}$

Encryption: Given a plaintext M and the public key $\langle e, N \rangle$, compute the ciphertext $C = M^e \bmod N$.

Decryption: Given a ciphertext C and the private key $\langle d, N \rangle$, compute the plaintext $M = C^d \bmod N$.

In 1982 a new technique that recovers M from C, by preprocessing the private key was introduced by J-J. Quisquater and C. Couvreur [9]. This method consists of calculating two integers $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$, and two texts M_p and M_q, where $M_p = C^{d_p} \mod p$ and $M_q = C^{d_q} \mod q$. Applying the Chinese Reminder Theorem (CRT) [6] on M_p and M_q we recover the plaintext M. In this document we refer to this technique as QC RSA, and to the version created by Rivest, Shamir and Adleman [10] as original RSA.

Using the criterion presented in [2], we make a theoretical cost estimation in terms of the number of exponentiations executed by the two methods. Basic algorithms to compute exponentiations of the form $C^d \bmod N$ take time $O(\log d \log^2 N)$. When d is of the order of N, these algorithms take time $O(\log^3 N)$. Theoretical speedup of QC RSA with relation to the original RSA (S_{RSA}), is given then as:

$$S_{RSA} = \frac{\log^3 N}{2(\log N/2)^3} = 4$$

The result above shows that this technique achieves a gain of about 4 times relatively to the decryption exponentiations of original RSA. For a more complete analysis (on the decryption of this scheme), we would have to take into account the cost of the CRT, which would slightly decrease the presented result. Actual measurements indicate a gain of 3.24 for 768-bits moduli, 3.32 for 1024-bits moduli and 3.47 for moduli of 2048 bits.

The improvement obtained with the use of this method led to its adoption (still in rudimentary form) in PKCS#1 since version 1.5 and today it can be considered the standard implementation of RSA.

3 RSA Variants

In this section we present two of the four algorithms analyzed by Dan Boneh and Hovav Shacham [2] namely, MPrime RSA and Rebalanced RSA. The Batch RSA and MPower RSA variants are not described here for being outside the scope of our work.

3.1 Mprime RSA - (*Multi-prime RSA*)

Mprime RSA was introduced by Collins et al. [3], who modified the RSA modulus so that it consists of k primes ($N = p_1 p_2 ... p_k$) instead of the traditional two primes p and q. The key generation, encryption and decryption algorithms are as follows:

Key generation: 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 $p_1, ..., p_k$ each one $\lfloor \log N/k \rfloor$ bits in length and $N \leftarrow \prod_{i=1}^{k} p_i$.

2 - Compute e and d such that $d = e^{-1} \bmod \phi(N)$, where $\gcd(e, \phi(N)) = 1$. $\phi(N) = \prod_{i=1}^{k}(p_i - 1)$.

3 - For $1 \leqslant i \leqslant k$, compute $d_i = d \bmod (p_i - 1)$.

$\boxed{\text{Public key} = \langle N, e \rangle}$ $\boxed{\text{Private key} = \langle N, d_1, d_2, ...d_k \rangle}$

Encryption: Given a public key $\langle N, e \rangle$ and a message $M \in Z_N$, encrypt M exactly as in the original RSA, thus $C = M^e \bmod N$.

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

Observe that this method considers to reduce the time expense by modular exponentiation evaluating a larger number of exponentiations with reduced moduli and private exponents. In this way, instead of evaluating, in decryption, a single exponentiation using $(\log N)$-bit modulus and with a large private exponent, we will have k exponentiations on moduli of $\lfloor \log N/k \rfloor$ bits and on a reduced private exponent, which is more efficient.

Considering that evaluating $C^d \bmod N$ takes time $O(\log^3 N)$, the theoretical speedup of Mprime RSA with relation to the QC RSA (S_{QC}), is given as follows:

$$S_{QC} = \frac{k^2}{4}$$

Thus, for $k = 3$ we have a theoretical gain of 2.25 relatively to QC RSA.

3.2 Rebalanced RSA

Rebalanced RSA is based on comments by Wiener [2, 12] on the weakness in the use of the private exponent d. This variant improves the performance of the decryption algorithm displacing the work to the encryption algorithm. This is done by randomly choosing a d that it is large (of the order of N), such that $d \bmod p-1$ and $d \bmod q-1$ are small (of the order of s bits; normally $s = 160$ is used), thus reducing the decryption time. We describe this process in more detail in key generation, encryption and decryption algorithms.

Key generation: The key generation algorithm takes $s \leqslant \lceil \log N \rceil / 2$, and executes the following steps:

1 - Generate two distinct random $\lfloor \log N/2 \rfloor$ bits primes p and q with $\gcd(p-1, q-1) = 2$, and calculate $N \leftarrow pq$.

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 $d = d_q \bmod q-1$ (see [2]).

4 - Calculate $e \leftarrow d^{-1} \bmod \phi(N)$.

$\boxed{\text{Public key} = \langle N, e \rangle}$ $\boxed{\text{Private key} = \langle p, q, d_p, d_q \rangle}$

Encryption: Encrypting a plaintext M using the public key $\langle N, e \rangle$ is identical to the original RSA. Remember, however, that the public exponent e is much larger than that used normally in the RSA and thus the entity encrypting the message M must be prepared to face a higher computational cost when using such public keys.

Decryption: This is the same method used in QC RSA.

Considering that evaluating $C^d \bmod N$ take time $O(\log d \log^2 N)$, the theoretical speedup of Rebalanced RSA with respect to the QC RSA (S_{QC}), is given by:

$$S_{QC} = \frac{\log N}{2s}$$

The above result show that for moduli of 2048 bits with $s = 160$, Rebalanced RSA is theorically 6.4 faster than QC RSA (the pratical results are available in section 5.2). Next, we describe the new proposed variant, Rprime RSA.

4 RPrime RSA

The Rebalanced RSA and Mprime RSA methods can be effectively combined [2]. The general ideia 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 new key generation, encryption and decryption algorithms are as follows:

Key generation: The key generation algorithm takes a integer $s \leqslant \lceil \log N \rceil / k$ and executes the following steps:

1 - Generate k distinct random primes of $\lfloor \log N/k \rfloor$ bits $p_1, p_2 ..., p_k$, with $\gcd(p_1 - 1, p_2 - 1, ..., p_k - 1) = 2$; and calculate $N \leftarrow p_1 p_2 ... p_k$.

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

3 - Find d such that $d = d_{p_1} \bmod p_1 - 1$, $d = d_{p_2} \bmod p_2 - 1$, $..., d = d_{p_k} \bmod p_k - 1$ (see [8]).

4 - Calculate $e \leftarrow d^{-1} \bmod \phi(N)$.

$\boxed{\text{Public key} = \langle N, e \rangle}$ $\boxed{\text{Private key} = \langle p_1, p_2, ..., p_k, d_{p_1}, d_{p_2}, ...d_{p_k} \rangle}$

Encryption: Again, encrypting with the public key $\langle N, e \rangle$ is identical to the original RSA. Remember, 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: To decode C, first calculate $M_i = C^{d_{p_i}} \bmod p_i$ for each i, $1 \leqslant i \leqslant k$, and then, combine the M_i's by means of the CRT to get $M = C^d \bmod N$. The use of the CRT takes negligible time if compared with k exponentiations.

The theoretical speedup (S_{QC}), is therefore given by:

$$S_{QC} = \frac{(\log N)k}{4s}$$

4.1 Security of RPrime RSA

Clearly, the security of RPrime RSA, as well as that of Rebalanced RSA, depends on the security offered by the private exponent d (with the described characteristics in the previous section (section 4) and on the size of the used primes (as MPrime RSA). We know that such private exponent d is enough large to become ineffective the attacks of the small private exponents [1]. And, the attacks on the small public exponents are not a problem, due the size of the public exponent e generated by the key generation algorithm.

Using exponents d_p, d_q and d_r of 160 bits each (for $k = 3$), we can guarantee a security of 2^{80} against the factoring of N [2], for the attack mentioned in [2, 12]. Already, to prevent the factoring by NFS and ECM methods, we must use primes larger than 256 bits, hence, for a modulus of 1024 bits we must use in the maximum three primes, and do not have to use more than two primes for a modulus of 768 bits, taken the same caution demanded on Mprime RSA.

M. Jason Hinek [5] made an analysis of the partial key exposure attack on the MPrime RSA and verified that for three and four primes the attack becomes ineffective; getting an experimental evidence that suggests that execution time of the attack is exponential in the size of modulus RSA, which we believe can be extended for the security of Rprime RSA.

5 Results

In order to get a better estimate of the performance of decryption of RPrime RSA, we compare it with other variants. The first one of these, known as Batch RSA [4], decrypts simultaneously b messages with the approximate cost of a single exponentiation (of order N) and some small exponentiations (using public exponents). The second one uses moduli of the form $N = p^{k-1}q$ [11] and is called MPower RSA [2].

5.1 Theoretical results

Using the criterion indicated in section 2 (number of executed exponentiations) we present in the table 1 a theoretical estimation of the speedup for each variant with respect to the decryption of QC RSA.

For 768-bits moduli the variant that exhibits better performance would be Batch RSA, but for 1024 and 2048 bits moduli Rprime RSA presents the best performance. Notice that while the speedup of Batch, MPrime and MPower variants is fixed regardless of the size of the used moduli, speedup of the Rebalanced and the RPrime variants significantly increases with larger moduli. This happens because we consider s fixed and equal to 160 bits (remember that s is the size of the exponent used in decryption algorithm), while this exponent increases for all other variant.

Speedup (S_{QC})			
Variant	768	1024	2048
Batch $= b$	b	b	b
Mprime $= k^2/4$	2,25	2,25	2,25
Mpower $= k^3/8$	3,37	3,37	3,37
Rebalanced $= (\log N)/2s$	2,40	3,20	6,40
Rprime $= (\log N)k/4s$	3,60	4,80	9,60

Table 1. Theoretical speedup related to the decryption exponentiations - For $b = 4$, $k = 3$ e $s = 160$.

5.2 Experimental results

The experimental results obtained for the decryption algorithms are listed in the table 2. Since the observed times include not only the exponentiations but also other operations like multiplications and the evaluation of the CRT, the results differ from the ones presented in the previous section. All measurements were conducted on an AMD Athlon XP 1400+ platform, with 256 MB of RAM and using GNU MP (library GMP [7]). One thousand messages and 20 keys had been generated, for each analyzed modulus being the result below obtained by the arithmetic average of the time in microseconds expense to decrypt the messages (more details, standard deviation, etc - see [8]).

The variant with the most divergent result was Batch RSA, where the use of small exponentiations and multiplications had harmed the gain brought by reduction of the great exponentiations. The others variations had a light fall with regard to table 1 what is natural because decryption also enclosing other operations, as the calculation of the CRT. The variant that got the best ones resulted was RPrime RSA arriving at a gain of 30% on Rebalanced RSA and 783% on QC RSA (for 2048 bits modulus). This represented in practical a gain approximately of 2720% on the original RSA decryption [8].

Speedup (S_QC)			
Variant	768	1024	2048
Batch	2,47	2,78	3,42
Mprime	1,95	1,89	1,97
Mpower	2,49	2,54	2,79
Rebalanced	2,52	3,02	5,98
Rprime	3,00	3,88	7,83

Table 2. Pratical speedup related to decryption process - Para $b = 4$, $k = 3$ e $s = 160$.

Analyzing the variants described in [2], we note that some other combinations might be attempted. A first combination, would be a mix of Mprime RSA or QC RSA, with Batch RSA [4]. In other words, reduce each C_i (of Batch RSA) modulus p_i ($1 \leqslant i \leqslant k$), combining later these results through the CRT[1].

Considering, however, that decryption process of Batch RSA implies the execution of small exponentiations for each prime used[2], we opt to implementing this method with parameter $k = 2$, in other words, our implementation of Batch RSA use the same technique aplied in QC RSA.

One could also consider the combination of the techniques used by Rebalanced RSA and Mpower RSA. But unfortunately, this combination produces a slow variation for both encryption and decryption. This happens because the decryption algorithm used in MPower RSA makes use of the public exponent e, and the exponent e used by Rebalanced RSA is much larger than the standard, increasing the cost of the modular exponentiations in both process (we therefore do not recommend this variant).

5.3 Advantagens, Disadvantages and applications of new method

We cite here some characteristics of our proposed variant:

- Use of the same encryption and decryption algorithms as Mprime RSA, facilitating code reutilisation and portability for the PKCS#1.
- Excellent performance in decryption (gain of 7,83 on QC RSA for $n = 2048$).
- Performance increases with larger moduli (considering s fixed).
- Especially indicated for moduli larger than 768 bits (to prevent the ECM and NFS factoring attacks [8]).
- Low performance in encryption (time equivalent to the decryption of the original RSA [8] - public exponent much larger than usual).

The idea of reducing the decryption time in detriment of the encryption, used by Rebalanced RSA and Rprime RSA, seems first sight not to present advantages in practical terms. However, there are applications where the balancing characteristic of these algorithms is desirable. Consider, for instance, a situation where the signature generation is executed much more often than verification. A bank, for example, can emit many digital signatures in a single day (in documents, receipts), while the user that receives this signature, has usually a much smaller burden. In this situation is reasonable to transfer the computional effort demanded for the signatures generation to the party

[1] In the case of the combination with QC RSA, we have $k = 2$.

[2] Hence the larger the parameter k, the more trees will be used and consequently more small exponentiations will be necessary, compromising the gain obtained in the exponentiation phase [8].

verifying them.

Another example is provided by applications running on *handheld devices* (PDAs), which generally possess limited computational resources. In communications with servers (or even with *notebooks* or *desktop computers*), we could leave the task of decryption (fast) for the small device, and the encryption (slow) for the computers with more computational resources. A still better alternative would be to use an implementation of MPrime RSA with keys of the MPrime and RPrime RSA, with the use depending on the type of communication (desktop/desktop, or, desktop/handheld), in other words, to use a scheme of hybrid keys.

6 Conclusion

Considering the comments and the recommendations presented in section 4.1 and that the confidence with relation to the one security cryptosystem is obtained empirically . We suggest the following use for these algorithms:

If one desires good encryption and decryption performance and interoperability with systems that already implement the PKCS#1 we recommend the use of MPrime RSA.

Although Mpower and Batch RSA achieve better performance than MPrime and hence constitute better option when high speed is desired, they are not specified in PKCS#1. We remember, moreover, the fragility of the keys used for a good performance of Batch RSA and its use of an agglomerate of messages, which certainly will affect the performance.

For the applications that prioritize the performance the decryption and the signature generation, the best choice is RPrime RSA, which for 2048-bits moduli got a gain of 30% with relation to Rebalanced RSA and is therefore about 27 times faster than original RSA and about 8 times faster than QC RSA (table 2). Besides, this variant can interoperate with systems that already use the PKCS #1. Another fact that favors this variation is that current systems that use MPrime RSA can easily be adapted to it, it is enough to modify the key generation algorithm or create a hybrid key system.

7 Acknowledgements

We would like to thank Paulo Barreto and Leon Achjian Jr for their comments and support during the development of this work.

References

1. D. Boneh. Twenty Years of Attacks on the RSA. *Notices of the American Mathematical Society*, vol 46(2):203–213, 1999.
2. D. Boneh and H. Shacham. Fast Variants of RSA. *RSA Laboratories*, 2002.
3. T. Collins, D. Hopkins, S. Langford, and M. Sabin. Public Key Cryptographic Apparatus and Method. *US Patent #5,848,159*, Jan. 1997.
4. A. Fiat. Batch RSA. *Advances in Cryptology: Proceedings of Crypto '89*, 435:175–185, 1989.
5. M. Jason Hinek. Low Public Exponent Partial Key and Low private Exponent Attacks on Multi-prime rsa. *Master Thesis, Waterloo*, Ontario - Canada, 2002.
6. A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
7. GNU MP. GMP. *Version 4.1 -2002, available in* `http://www.swox.com/gmp/`.
8. Cesar A. M. Paixão. Implementação e Análise Comparativa de Variações do criptossistema RSA. *Master thesis, Instituto de Matemática e Estatística*, Universidade de São Paulo - Brasil, 2003.
9. J-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Eletronic Letters*, vol 18:905–907, 1982.
10. R. Rivest, A. Shamir, and L. Adleman. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems.* Commum. Of the ACM 21(2): 120 - 126, 1978.
11. T. Takagi. Fast RSA-type Cryptosystem Modulo $p^k q$. *In H. Krawczyk, ed., Proceedings of Crypto '98*, 1462 of LNCS. Springer-Verlag:318–326, 1998.
12. M. Wiener. Cryptanalysis of Short RSA Secret Exponents. *IEEE Transactions on Information Theory*, pages 36(3):553–558, 1990.