An efficient variant of the RSA cryptosystem

Cesar Alison Monteiro Paixão*[1], Décio Luiz Gazzoni Filho [2]

[1]Instituto de Matemática e Estatística
Universidade de São Paulo
Rua do Matão, 1010
05508-090 – São Paulo – SP

[2]Departamento de Engenharia de Computação e Sistemas Digitais (PCS)
Escola Politécnica
Universidade de São Paulo
Av. Prof. Luciano Gualberto trav. 3 n. 158
05508-900 – São Paulo – SP

capaixao@ime.usp.br,decio@decpp.net

Abstract. *We describe an efficient combination of two variants of the RSA cryptosystem (MPrime and Rebalanced RSA) analyzed by Boneh and Shacham [Boneh and Shacham 2002]. For 2048-bit moduli, the resulting decryption process is about 8 times faster than that presented by Quisquater and Couvreur [Quisquater and Couvreur 1982] and about 27 times faster than the original cryptosystem.*

Resumo. *Descrevemos uma combinação eficiente de duas variantes do criptossistema RSA (MPrime e Rebalanced RSA) analisadas por Boneh e Shacham [Boneh and Shacham 2002]. Para módulos de 2048 bits, o processo de decriptação resultante é cerca de 8 vezes mais rápido que o apresentado por Quisquater e Couvreur [Quisquater and Couvreur 1982], e cerca de 27 vezes mais rápido que o criptossistema original.*

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) analysed in [Boneh and Shacham 2002], with the goal of reducing the decryption and signature generation times of the original cryptosystem. We briefly discuss the feasibility 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 [Quisquater and Couvreur 1982].

This paper is organized as follows. In Section 2. we review the RSA cryptosystem and the Quisquater-Couvreur method [Quisquater and Couvreur 1982]. In Section 3. we describe Mprime and Rebalanced RSA, two variants of plain RSA. In Section 4. we

*Co-sponsored by Scopus Tecnologia S.A.

present RPrime RSA, our proposed combination of MPrime and Rebalanced methods. In Section 5. we present some theoretical and experimental results. In Section 6., we show how to generate a certificate that proves a lower bound on the security of the system to any interested third parties. We conclude in Section 7. with some comments on RPrime RSA.

2. RSA Cryptosystem

Before presenting our proposal to improve the RSA cryptosystem, we review the three basic algorithms that constitute the RSA cryptosystem, together with a frequently used optimization technique.

Algorithm 2..1 (RSA)**.**

1. [Key generation]
   Generate two primes p, q and compute their product $n = pq$;
   Pick e such that $\gcd(e, \phi(n)) = 1$, where $\phi(n) = (p-1)(q-1)$;
   Compute d such that $d = e^{-1} \bmod \phi(n)$;
   The public-key is $\langle n, e\rangle$, while the private key is $\langle n, d\rangle$.
2. [Encryption]
   Given a plaintext M and the public key $\langle n, e\rangle$, compute the ciphertext $C = M^e \bmod n$.
3. [Decryption]
   Given a ciphertext C and the private key $\langle n, d\rangle$, 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 [Quisquater and Couvreur 1982]. Their method splits the private key in two parts, $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$, and recovers the plaintext M from the ciphertext C by first computing modulo each factor of n, i.e. $M_p = C^{d_p} \bmod p$ and $M_q = C^{d_q} \bmod q$, followed by reconstruction of M_p and M_q modulo n using the Chinese Reminder Theorem (CRT) [Jones and Jones 1998]. In this document we refer to this technique as QC RSA, and to the version created by Rivest, Shamir and Adleman [Rivest et al. 1978] as plain RSA.

Using the criterion presented in [Boneh and Shacham 2002], we estimate the complexity of each method as a function of the number of operations required. Basic algorithms to compute exponentiations of the form $C^d \bmod n$ take time $O(\log d M(n))$, where $M(n)$ is the cost of multiplying two n-bit integers and can be taken as $O(\log^2 n)$. When $d = O(n)$, these algorithms take time $O(\log^3 n)$. On the other hand, QC RSA has $dp = dq = O(\sqrt(n))$ (so that $\log dp = \log dq = O(\log(n)/2)$), for an overall cost of $O(2(\log(n)/2)^3)$. Thus, the theoretical speedup of QC RSA with respect to plain RSA (S_{RSA}) is

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

That is, the decryption procedure of QC RSA is about 4 times faster than that of plain RSA. A more refined analysis would take into account the cost of the final CRT and

would arrive at a slightly smaller speedup. Actual measurements suggest a gain of 3.24 for 768-bit moduli, 3.32 for 1024-bit moduli and 3.47 for 2048-bit moduli.

This improvement led to the adoption of QC RSA (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 in [Boneh and Shacham 2002] — namely, MPrime RSA and Rebalanced RSA. The Batch RSA and MPower RSA variants are not described in detail here, although they are mentioned in section 5 for comparison purposes.

3.1. Mprime (multi-prime) RSA

Mprime RSA was introduced by Collins et al. [Collins et al. 1997]. It differs from plain RSA by constructing moduli with k prime factors ($n = p_1 p_2 \cdots p_k$) instead of only two. The key generation, encryption and decryption algorithms are as follows:

Algorithm 3..1 (Mprime RSA)**.**

1. [Key generation]
   Generate k distinct primes $p_1, \ldots, p_k$, each $\lfloor \lg(n)/k \rfloor$ bits in size, and compute $n = \prod_{i=1}^{k} p_i$;
   Pick e such that $\gcd(e, \phi(n)) = gcd(e, \prod_{i=1}^{k}(p_i - 1)) = 1$;
   Compute d such that $d = e^{-1} \bmod \phi(n)$;
   For $1 \leq i \leq k$, compute $d_i \equiv d \pmod{p_i - 1}$;
   The public key is $\langle n, e \rangle$, while the private key is $\langle n, d_1, \ldots, d_k \rangle$.
2. [Encryption]
   Given a public key $\langle n, e \rangle$ and a message $M \in \mathbb{Z}/n\mathbb{Z}$, encrypt M as in plain RSA, i.e. $C = M^e \bmod n$.
3. [Decryption]
   The decryption is an extension of the Quisquater-Couvreur method. To decrypt a ciphertext C, first compute $M_i = C^{d_i} \bmod p_i$ for $1 \leq i \leq k$. Next, apply the CRT to the M_i's to obtain $M = C^d \bmod n$.

Mprime RSA achieves a decryption speedup relative to plain and QC RSA by reducing the size of exponents and moduli, at the cost of extra modular exponentiations. However, a linear increase in the number of exponentiations translates to a cubic decrease in the cost of each exponentiation, for an overall speedup that is quadratic in the number of factors k of the modulus. Formally, evaluating $C^d \bmod n$ for $d = O(n)$ costs $O(\log^3 n)$, whereas Mprime RSA has $d_i = O(n^{1/k})$ (so that $\log d_i = O(\log(n)/k)$) and multiplication cost of $O((\log(n)/k)^2)$ for an overall cost of $O(k(\log(n)/k)^3) = O(\log^3(n)/k^2)$. Since QC RSA is already 4 times as fast as plain RSA, the theoretical speedup of Mprime RSA over QC RSA (S_{QC}) is

$$S_{\text{QC}} = k^2/4.$$

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

3.2. Rebalanced RSA

Rebalanced RSA is based on comments by Wiener [Boneh and Shacham 2002, Wiener 1990] on the weakness in the use of the private exponent d. This variant improves decryption performance at the expense of encryption performance. This is done by choosing d such that $d \bmod p-1$ and $d \bmod q-1$ are small (on the order of s bits; usually $s = 160$). Unfortunately, such a choice of d leads to large values of e (as large as n itself). We now present the key generation, encryption and decryption procedures for Rebalanced RSA.

Algorithm 3..2 (Rebalanced RSA)**.**

1. [Key generation]
   Generate two random primes p, q, each $\lfloor \lg(n)/2 \rfloor$ bits in size, with $\gcd(p-1, q-1) = 2$, and compute $n = pq$;
   Generate two random s-bit integers d_p and d_q, such that $\gcd(d_p, p-1) = gcd(d_q, q-1) = 1$ and $d_p \equiv d_q \pmod 2$;
   Apply the CRT to obtain d such that $d \equiv d_p \pmod{p-1}$ and $d \equiv d_q \pmod{q-1}$;
   Compute $e = d^{-1} \bmod \phi(n)$;
   The public key is $\langle n, e \rangle$, while the private key is $\langle p, q, d_p, d_q \rangle$.
2. [Encryption]
   Apply the encryption procedure of plain RSA. Note though that $e = O(n)$ instead of $O(1)$ as in plain RSA, thus public-key operations will be more costly.
3. [Decryption]
   Apply the decryption procedure of QC RSA.

We now consider the efficiency of Rebalanced RSA. We recall that d_p, d_q have s bits each (hence $\log d_p = \log d_q = O(s)$). The cost of modular multiplications is the same as that of QC RSA, so the only difference is the number of multiplications computed during each modular exponentiation. As $\log d_p = \log d_q = O(\log(n)/2)$ for QC RSA, we arrive at a theoretical speedup of Rebalanced RSA with respect to QC RSA (S_{QC}) of

$$S_{\text{QC}} = \frac{\log n}{2s}.$$

This result implies that, for moduli of 2048 bits with $s = 160$, Rebalanced RSA is theoretically 6.4 faster than QC RSA (practical results are provided in Section 5.). Now we describe our proposed variant, Rprime RSA.

4. RPrime RSA

The Rebalanced RSA and Mprime RSA methods can be efficiently combined [Boneh and Shacham 2002]. The key generation procedure of Rebalanced RSA (modified for k primes) is employed together with the decryption procedure of Mprime RSA. The new key generation, encryption and decryption algorithms are as follows:

Algorithm 4..1 (RPrime RSA)**.**

1. [Key generation]
   Generate k random primes $p_1, \ldots, p_k$, each $\lfloor \lg(n)/k \rfloor$ bits in size, with $\gcd(p_1 - 1, \ldots, p_k - 1) = 2$, and compute $n = \prod_{i=1}^{k} p_i$;
   Generate k random s-bit integers $d_{p_1}, \ldots, d_{p_k}$ such that $\gcd(d_{p_1}, p_1 - 1) = \ldots =$

$\gcd(d_{p_k}, p_k - 1) = 1$ and $d_{p_1} \equiv \ldots \equiv d_{p_k} \pmod 2$;
Apply the CRT to obtain d such that $d \equiv d_{p_i} \pmod{p_i - 1}$ for $1 \leq i \leq k$ [Paixão 2003];
Calculate $e = d^{-1} \bmod \phi(n)$;
The public key is $\langle n, e \rangle$, while the private key is $\langle p_1, \ldots, p_k, d_{p_1}, \ldots, d_{p_k} \rangle$.

2. [Encryption]
   Apply the encryption procedure of plain RSA. As was the case in Rebalanced RSA, we have $e = O(n)$ instead of $O(1)$ as in plain RSA, leading to more costly public-key operations.
3. [Decryption]
   Compute $M_i = C^{d_{p_i}} \bmod p_i$ for $1 \leq i \leq k$;
   Apply the CRT to the M_i's to obtain $M = C^d \bmod n$.

Given that the d_i's have s bits each (hence $\log d_i = O(s)$ for all i), and the cost of multiplication is $(\log(n)/k)^2$, we arrive at an overall cost of $O(ks(\log(n)/k)^2) = O(s \log^2(n)/k)$. The theoretical speedup over QC RSA (S_{QC}) is then

$$S_{QC} = \frac{\log^3(n)/4}{s \log^2(n)/k} = \frac{k \log n}{4s}.$$

4.1. Security of RPrime RSA

The security of RPrime RSA depends on the security offered by the private exponent d (as in Rebalanced RSA), under the constraints described in Section 4., and on the size of the primes employed (as in Mprime RSA). The private exponent d is large enough that attacks on small d are ineffective [Boneh 1999]. Attacks on small public exponents e are not a problem either, due to the large e's produced by the key generation procedure.

For $k = 3$ and using exponents d_{p_1}, d_{p_2} and d_{p_3} of 160 bits each, the complexity of factoring n is $O(2^{80})$ using the attack of [Boneh and Shacham 2002, Wiener 1990]. To prevent factorization by ECM, primes larger than 256 bits must be employed, hence a modulus of 1024 bits can have no more than three prime factors, while a 768-bit modulus should employ at most two factors for the same reason.

M. Jason Hinek [Hinek 2002] analyzed a partial key exposure attack on MPrime RSA, concluding that the attack is ineffective for three and four primes. He also obtained experimental evidence that the attack has running time exponential in the size of the modulus n, which we believe can be extended for the security of Rprime RSA.

5. Experimental Results

In order to get a better estimate of the decryption performance of RPrime RSA, we benchmark it against other variants. The first we'll consider is Batch RSA [Fiat 1989], which simultaneously decrypts 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 $n = p^{k-1}q$ [Takagi 1998] and is called MPower RSA [Boneh and Shacham 2002].

Theoretical results shown in previous sections and experimental results obtained for the decryption algorithms are listed in Table 1. All measurements were performed

Speedup (S_{QC})	Theoretical			Experimental		
Modulus / Variant	768	1024	2048	768	1024	2048
Batch	4	4	4	2,47	2,78	3,42
Mprime	2,25	2,25	2,25	1,95	1,89	1,97
Mpower	3,37	3,37	3,37	2,49	2,54	2,79
Rebalanced	2,40	3,20	6,40	2,52	3,02	5,98
Rprime	3,60	4,80	9,60	3,00	3,88	7,83

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

on an AMD Athlon XP 1400+ platform, with 256 MB of RAM and using the GNU MP library [Granlund 2005] for integer arithmetic. Our methodology consisted of generating 1000 messages and 20 keys for each moduli size, and computing the arithmetic mean of the time spent to decrypt each message. For more details (standard deviations, etc.) see [Paixão 2003].

Experimental results differ from the theoretical results mainly because the observed times include not only exponentiations but some overhead (including CRT computations) that was neglected in our theoretical analyses. Batch RSA had the most noticeable decrease from expected results to experimental measurements. The small exponentiations and multiplications, which weren't taken into account in our analysis, are significant for the actual performance of this variant. The remaining variants showed only a slight decrease from expected results. Another fact to consider is that speed up of Rebalanced and Rprime variants significantly increases with larger moduli while the others variants remains stable. This is because we consider s fixed and equal 160 bits (recall that s is the size of the exponent used in decryption algorithm), while this exponent increases with moduli size for all other variants.

Overall, the best performing variant was RPrime RSA, showing a 30% improvement over Rebalanced RSA and 783% improvement over QC RSA, both for 2048 bits moduli. Compared to the plain RSA system, this represents a gain of approximately 2720% or 27 times [Paixão 2003].

Analyzing the variants described in [Boneh and Shacham 2002], we note that some other combinations might be attempted. One possibility would be a mix of Mprime RSA or QC RSA with Batch RSA [Fiat 1989]. That is, reduce each C_i (from Batch RSA) modulo $p_i, 1 \leq i \leq k$, later reconstructing these results by the CRT[1].

Unfortunately, as the decryption process of Batch RSA requires the execution of small exponentiations for each prime employed[2], it is more advantageous to implement this method with parameter $k = 2$, that is, using QC RSA instead of Mprime RSA.

One could also consider combining Rebalanced RSA and Mpower RSA. Unfortunately, this variant has bad performance for both encryption and decryption. The reason

[1]If Batch RSA is combined 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, reducing the gain obtained in the exponentiation phase [Paixão 2003].

is that MPower RSA's decryption algorithm employs the public exponent e, and in Rebalanced RSA the exponent e is much larger than plain RSA's public exponents, increasing the cost of modular exponentiations for both procedures. We therefore do not recommend this variant.

5.1. Advantagens, disadvantages and applications of new method

We mention here some characteristics of our proposed variant:

- Uses the same encryption and decryption algorithms as Mprime RSA, allowing code reuse and portability for PKCS#1-compliant implementations;
- Achieves excellent decryption performance (gain of 7,83 on QC RSA for $n = 2048$);
- Shows performance increases with larger moduli (if s is fixed);
- Is not recommended for moduli of up to 768 bits (to prevent factorization by ECM [Paixão 2003]);
- Has low encryption performance due to large public exponent e (similar to the decryption performance of plain RSA) [Paixão 2003].

The idea of trading off encryption for decryption performance, used by Rebalanced RSA and Rprime RSA, may not be seen as an advantage in practice. However, there are applications where it is desirable. A bank, for instance, has to digitally sign documents for all of its customers, whereas each customer only has to check their own signature. In this situation, it is reasonable to offset some of the computional effort demanded in generating signatures to the verifier.

A second scenario are applications running on handheld devices such as PDAs, which generally possess limited computational resources. If the handheld device is performing private-key operations while communicating with more powerful devices (such as servers, or when docking to notebooks or desktops), then RPrime RSA will achieve better overall performance, as the computational effort is more evenly distributed according to resources available to each party. An even better alternative would be to choose between Mprime and RPrime RSA keys according to the type of communication being performed and the devices involved.

6. Certification of ordinary and multiprime RSA moduli

The security of multiprime RSA variants is threatened by insecure parameter choices. Since only the private key owner is aware of how many primes were employed and how large they are, it may be tempting to trade off security for performance in a dangerous fashion, relying on the adversary's supposed inability to tell whether a given RSA modulus is comprised of many small primes or a few large primes. If this practice becomes commonplace, then the perceived security of multiprime RSA variants will be reduced.

In this section we propose a technique that can provide a lower bound on the size of the factors in an RSA modulus. As a side effect, the maximum number of prime factors is provided as well[3]. It applies equally well to ordinary RSA and multiprime variants.

[3]While the exact number of factors is not provided, this is of little consequence to the difficulty of factoring the modulus.

Speedup (S_{QC})	Theoretical			Experimental		
Modulus / Variant	768	1024	2048	768	1024	2048
Batch	4	4	4	2,47	2,78	3,42
Mprime	2,25	2,25	2,25	1,95	1,89	1,97
Mpower	3,37	3,37	3,37	2,49	2,54	2,79
Rebalanced	2,40	3,20	6,40	2,52	3,02	5,98
Rprime	3,60	4,80	9,60	3,00	3,88	7,83

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

on an AMD Athlon XP 1400+ platform, with 256 MB of RAM and using the GNU MP library [Granlund 2005] for integer arithmetic. Our methodology consisted of generating 1000 messages and 20 keys for each moduli size, and computing the arithmetic mean of the time spent to decrypt each message. For more details (standard deviations, etc.) see [Paixão 2003].

Experimental results differ from the theoretical results mainly because the observed times include not only exponentiations but some overhead (including CRT computations) that was neglected in our theoretical analyses. Batch RSA had the most noticeable decrease from expected results to experimental measurements. The small exponentiations and multiplications, which weren't taken into account in our analysis, are significant for the actual performance of this variant. The remaining variants showed only a slight decrease from expected results. Another fact to consider is that speed up of Rebalanced and Rprime variants significantly increases with larger moduli while the others variants remains stable. This is because we consider s fixed and equal 160 bits (recall that s is the size of the exponent used in decryption algorithm), while this exponent increases with moduli size for all other variants.

Overall, the best performing variant was RPrime RSA, showing a 30% improvement over Rebalanced RSA and 783% improvement over QC RSA, both for 2048 bits moduli. Compared to the plain RSA system, this represents a gain of approximately 2720% or 27 times [Paixão 2003].

Analyzing the variants described in [Boneh and Shacham 2002], we note that some other combinations might be attempted. One possibility would be a mix of Mprime RSA or QC RSA with Batch RSA [Fiat 1989]. That is, reduce each C_i (from Batch RSA) modulo $p_i, 1 \leq i \leq k$, later reconstructing these results by the CRT[1].

Unfortunately, as the decryption process of Batch RSA requires the execution of small exponentiations for each prime employed[2], it is more advantageous to implement this method with parameter $k = 2$, that is, using QC RSA instead of Mprime RSA.

One could also consider combining Rebalanced RSA and Mpower RSA. Unfortunately, this variant has bad performance for both encryption and decryption. The reason

[1]If Batch RSA is combined 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, reducing the gain obtained in the exponentiation phase [Paixão 2003].

$p = u^2 + v^2$. The theory of complex multiplication states that elliptic curves $E(\mathbb{Z}/p\mathbb{Z})$ exist such that

$$\#E(\mathbb{Z}/p\mathbb{Z}) = p + 1 \pm 2u \text{ or } p + 1 \pm 2v$$

[Atkin and Morain 1993]. It is readily seen that these curve orders can be factored as

$$\#E(\mathbb{Z}/p\mathbb{Z}) = (u \pm 1)^2 + v^2 \text{ or } u^2 + (v \pm 1)^2.$$

The result is more naturally interpreted in terms of Gaussian integers: to each prime $p = u + iv$, we associate four curves with complex multiplication by $\mathbb{Z}[i]$, whose orders are given by the Gaussian integers adjacent to p in the Gaussian plane. In Section 6.3., the reciprocal of this statement will prove useful: if a Gaussian integer r is adjacent to a Gaussian prime, then r is the order of an elliptic curve modulo p with complex multiplication by $\mathbb{Z}[i]$.

Furthermore, these curves can be constructed in a straightforward manner [Atkin and Morain 1993]. If required, the decomposition $p = u^2 + v^2$ is efficiently produced by the algorithm of Cornacchia, with cost less than that of a square root modulo p and a gcd with arguments the size of p. Next a quadratic non-residue g modulo p must be found. Since $(p-1)/2$ integers modulo p are quadratic non-residues, and Jacobi testing by quadratic reciprocity is quite efficient, this step is of little consequence to the overall running time. Then the curves

$$y^2 = x^3 - g^k x, k = 0, 1, 2, 3$$

have complex multiplication by $\mathbb{Z}[i]$ and distinct group orders.

6.3. The certification method

We rely on the following result, which is a direct modification of the Goldwasser-Kilian ECPP theorem [Goldwasser and Kilian 1986] and was stated and proved by David Broadhurst in [Broadhurst 2005a]. Throughout this section, n is a composite that is not a power, and has k factors.

Theorem 6..1. *Let S be a prime. Suppose there exists an elliptic pseudocurve*[6] *$E(\mathbb{Z}/n\mathbb{Z})$ and a point $P = (x, y)$ on E, such that $SP = O$. Given e such that $S > (n^e + 1)^2$, then the prime factors of n are lower bounded by n^{2e}.*

Proof. Let p be any prime factor of n. Then $S \mid \#E(\mathbb{Z}/p\mathbb{Z})$. By Hasse's theorem, $\#E(\mathbb{Z}/p\mathbb{Z}) < (\sqrt{p}+1)^2$. It follows that $S < (\sqrt{p}+1)^2$, but $S > (n^e+1)^2$ by hypothesis. Then $(\sqrt{p}+1)^2 > (n^e+1)^2$, so that $\sqrt{p} > n^e$ and finally $p > n^{2e}$. □

The following corollary bounds the number of prime factors of n.

Corollary 6..2. *If $e > 1/(2j)$ in Theorem 6..1, then n has at most $j - 1$ prime factors.*

[6] An elliptic pseudocurve is an elliptic curve modulo a composite integer n, which doesn't give rise to a valid group of points over the curve (this only happens if the modulus n is prime), but still retains many of the properties of an actual elliptic curve. See [Cohen 1996] and [Crandall and Pomerance 2000] for more information on pseudocurves.

Proof. Let p_i denote the i-th of k prime factors of n. If $e > 1/(2j)$, then $p_i > n^{1/j}$ for all p_i. We have

$$n = \prod_{i=1}^{k} p_i > \prod_{i=1}^{k} n^{1/j} = \left(n^{1/j}\right)^k$$

and this leads to a contradiction if $k \geq j$. □

Note that Theorem 6..1 requires the point $P = (x, y)$ to have order S when considered modulo each of the prime factors of $n = \prod p_i$ — indeed, if $SP = O \pmod{p}_i$ but $SP \neq O \pmod{p}_j$ for some i, j, then $x \equiv 0 \pmod{p}_i$ but $x \not\equiv 0 \pmod{p}_j$, so $\gcd(x, n) = p_i$ provides a factor of the composite.

Recall that $S \mid \#E(\mathbb{Z}/p_i\mathbb{Z}), 1 \leq i \leq k$. To find p_i's that satisfy this relationship, it's easier to devise a search procedure that begins by constructing valid curve orders with shared factor S, then seeks primes p_i such that there exist elliptic curves modulo each p_i with the desired orders. Note that this search procedure is performed backwards in comparison to the ECPP algorithm, which is given a prime p and seeks an elliptic curve with certain properties.

At first, it's not clear how to produce curve orders divisible by S from the defining equations $\#E(\mathbb{Z}/p_i\mathbb{Z}) = (u \pm 1)^2 + v^2, u^2 + (v \pm 1)^2$ and $p_i = u^2 + v^2$. In principle, one can pick v at random and compute u such that $u^2 \equiv -(v \pm 1)^2 \pmod{S}$, until $p_i = u^2 + v^2$ is prime; then $S \mid \#E(\mathbb{Z}/p_i\mathbb{Z})$ by construction. (Clearly this argument is symmetric in u and v.) Doing this for each p_i, the composite n can be constructed. However, since the square root $\sqrt{u^2} \pmod{S}$ is typically as large as S itself, the constructed composite will be too large to fulfill the conditions of Theorem 6..1.

The key insight is to work over $\mathbb{Z}[i]$: choose a rational prime $S \equiv 1 \pmod{4}$ and factor it over $\mathbb{Z}[i]$ as $S = u_S + iv_S$. Then pick Gaussian integers $g = u_g + iv_g$ at random (subject to size restrictions, as will be clear) and compute Sg until an integer adjacent to Sg in the Gaussian plane is prime; we'll call it p_1. (When considered as a rational prime, we'll employ the square of its norm.) Then an elliptic curve $y^2 = x^3 + a_{p_1}x \pmod{p}_1$ exists such that $\#E(\mathbb{Z}/p_1\mathbb{Z}) = |Sg|^2$. Construct a point $P_{p_1} \in E(\mathbb{Z}/p_1\mathbb{Z})$ of order S by generating a random point P_0, computing $P_{p_1} = (\#E(\mathbb{Z}/p_1\mathbb{Z})/S)P_0$ and checking that $P_{p_1} \neq O$ (which will almost always be the case). Repeat this procedure for $i = 2, \ldots, k$ to find other primes p_i with $\#E(\mathbb{Z}/p_i\mathbb{Z})$ divisible by S and a point $P_{p_i} \in E(\mathbb{Z}/p_i\mathbb{Z})$ of order S. Apply the Chinese Remainder Theorem to the parameters $a_{p_1}, \ldots, a_{p_k}$ of the constructed elliptic curves, and coordinatewise to the points $P_{p_1}, \ldots, P_{p_k}$, to construct an elliptic pseudocurve $E(\mathbb{Z}/n\mathbb{Z}) : y^2 = x^3 + a_n x \pmod{n}$ and a point P_n on $E(\mathbb{Z}/n\mathbb{Z})$ of order S. The witness S, the composite n, the pseudocurve $E(\mathbb{Z}/n\mathbb{Z})$ and the point $P_n \in E(\mathbb{Z}/n\mathbb{Z})$ thus obtained satisfy the conditions of Theorem 6..1.

We formalize this search procedure in the following algorithm:

Algorithm 6..3. Given l, k and prime S, $S \equiv 1 \pmod{4}$, this algorithm generates an l-bit k-prime RSA modulus n and a certificate that n's factors are lower-bounded by S.

1. [Factor S over $\mathbb{Z}[i]$]
   Apply Cornacchia's algorithm to S to obtain $u_S^2 + v_S^2 = S$;
   for $(1 \leq j \leq k)$ {

2. [Find suitable curve orders]
   forever{
   Generate random u_j, v_j of size $(l/k - \lg S - 2)/2$ bits each;
   Compute a candidate order $\#E_j = 2S(u_j + iv_j) = u + iv$;
   if (one of $\{(u \pm 1)^2 + v^2, u^2 + (v \pm 1)^2\}$ is prime) {
   Set p_j to the prime value;
   break;
   }
   }
3. [Construct CM curves with desired orders]
   Find a quadratic non-residue g modulo p_j;
   for ($0 \leq k \leq 3$) {
   Let $E(\mathbb{Z}/p_j\mathbb{Z})$ be the curve $y^2 = x^3 - g^k \pmod{p_j}$;
   Find a point P_0 on $E(\mathbb{Z}/p_j\mathbb{Z})$;
   if ($(\#E_j)P_0 = O$) {
   $a_j = g^k$;
   break;
   }
   }
4. [Find points of order S]
   while ($(\#E_j/S)P_0 = O$)
   Find another point P_0 on $E(\mathbb{Z}/p_j\mathbb{Z})$;
   $P_j = (x_j, y_j) = (\#E_j/S)P_0$;
   }
5. [Lift parameters to $\mathbb{Z}/n\mathbb{Z}$]
   Using the Chinese Remainder Theorem:
   Compute a_n such that $a_n \equiv a_j \pmod{p_j}$ for $1 \leq j \leq k$;
   Compute $P_n = (x_n, y_n)$ such that $x_n \equiv x_j \pmod{p_j}$ and $y_n \equiv y_j \pmod{p_j}$ for $1 \leq j \leq k$;
6. [Return results]
   return $\{\{p_1, \ldots, p_k\}, E(\mathbb{Z}/n\mathbb{Z}), P_n\}$;

An explanation is in order concerning the factor 2 in the candidate group orders. In the equation $\#E = u^2 + v^2 = 2S(u_j + iv_j)$, the factor 2 guarantees that $u \equiv v \equiv 0 \pmod{2}$, so that the integers $\{(u \pm 1)^2 + v^2, u^2 + (v \pm 1)^2\}$ being tested for primality are always odd (being the sum of an even and an odd integer), thus twice as likely to be prime as random integers of the same size.

The attentive reader may question the need for CM curves, as it appears that supersingular curves could be used instead. Unfortunately, the simple relationship between a prime p and the curve order $p+1$ of a supersingular curve modulo p works against the goal of hiding the factor p inside a composite n. In fact, given a witness $S \mid p+1$, then trivially $p \equiv -1 \pmod{S}$. If $S > n^{1/4}$, which will often be the case, one can apply Coppersmith's improvement of Lenstra's divisors in residue classes algorithm [Coppersmith et al. 2004], which finds the factors of n in polynomial time with the information provided.

On the other hand, for an elliptic curve $E(\mathbb{Z}/p\mathbb{Z})$ with CM by $\mathbb{Z}[i]$, the relationship between p and $\#E(\mathbb{Z}/p\mathbb{Z})$ is trivial if the decomposition of p as a Gaussian prime is known, but hard to deduce otherwise. Since factorization into Gaussian primes can be no

easier than integer factorization, this provides an argument for the difficulty of factoring a composite given the information provided by a semiprimality certificate.

It is desirable to compress the size of semiprimality certificates. Our first strategy is to reduce the representation of S by picking a prime of special form, say $S = r^s + e$ for small r, s, e. A second strategy is to reduce the size of the curve parameter a. For instance, one could fix $a = \pm 1$. Although a valid curve would eventually be found by trial and error, after looking at 4^k curves on average, it's possible to do better. If $p \equiv 5 \pmod 8$ and $p = u^2 + v^2$, then one can take $u \equiv 1 \pmod 8$ and $v \equiv 4 \pmod 8$. It's known [Morain 1998] that the curves $E : y^2 = x^3 + x \pmod p$ and $E' : y^2 = x^3 - x \pmod p$ have orders $(u-1)^2 + v^2$ and $(u+1)^2 + v^2$, respectively. By considering candidate orders of these two forms only, the pigeonhole principle states that at most $2k - 1$ candidates are required before two of them share the same a (either 1 or -1).

6.4. Parameter selection for certificates

The only remaining question is the size of S relative to n. Recall that n has k prime factors. Then obviously $S < n^{1/k}$, by the argument in the proof of Corollary 6..2. One shouldn't take S too close to this bound, though, since n can be factored with effort $O(n^{1/k}/S)$ by a simple brute-force attack: choose a random Gaussian integer g of size $(\lg(n)/k - \lg S - 2)/2$ bits, compute candidate factors $p = |2Sg + i^j|^2, 0 \leq j \leq 3$ and test if $p \mid n$. To deflect this attack, one must choose S small enough in comparison to $n^{1/2}$ that generic factorization algorithms are easier to apply than this brute-force method.

With this in mind, we suggest the following choice for S: given the rough complexity c of factoring n, obtained by plugging in n or its factors p_i into the complexity estimates of NFS or ECM, choose S such that $c < n^{1/k}/S$ (or equivalently $S < n^{1/k}/c$). It is also prudent to add a small safety factor to S (by reducing its size, we stress), to ensure that factoring n by NFS or ECM is easier than an exhaustive search of factors using the method above.

7. Conclusion

Considering the comments and recommendations of Section 4.1., we suggest the following guidelines for deploying these algorithms:

- For good encryption and decryption performance, and interoperability with systems that already implement PKCS#1, we recommend the use of MPrime RSA.
- Although Mpower and Batch RSA achieve better performance than MPrime, they are not specified in PKCS#1. Moreover, we emphasize the fragility of keys that must be employed to obtain good performance out of Batch RSA, and the use of an agglomerate of messages, which will certainly affect the performance.
- For applications that demand high decryption and signing performance, the best choice is RPrime RSA, which for 2048-bits moduli showed an improvement of 30% over Rebalanced RSA, being therefore about 27 times faster than plain RSA and about 8 times faster than QC RSA (Table 1). Besides, this variant can interoperate with systems that implement PKCS #1. Also, systems that implement Mprime RSA can be easily modified to implement RPrime RSA as well; it suffices to modify the key generation procedure or create a hybrid key system.

We also presented a certification method that lower-bounds the size of prime factors of a multi-prime RSA modulus. The method is practical and easy to implement if an elliptic curve arithmetic library is available, and provides third-parties with an assurance of the security level of a given multi-prime RSA modulus against factoring attacks.

8. Acknowledgements

The first author would like to thank Paulo Barreto and Arthur Wongtschowski for their comments during the development of this work.

The second author would also like to thank Paulo Barreto for his assistance and for putting the two authors in touch in the first place. He is also thoroughly indebted to David Broadhurst, which pioneered the certification method for semiprimes, provided some proofs and optimizations, and was infinitely patient as he investigated many dead-ends, before finally arriving at the present algorithm. Lastly, David holds the current semiprimality certificate record, having produced a certified semiprime of 35 thousand decimal digits [Broadhurst 2005a].

References

Atkin, A. O. L. and Morain, F. (1993). Elliptic curves and primality proving. *Mathematics of Computation*, 61:29–68.

Boneh, D. (1999). Twenty years of attacks on the RSA. *Notices of the American Mathematical Society*, 46(2):203–213.

Boneh, D. and Shacham, H. (2002). Fast variants of RSA. *RSA Laboratories*.

Broadhurst, D. (2005a). A proof of semiprimality at 35205 digits. `http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0503&L=nmbrthry&P=1303`.

Broadhurst, D. (2005b). Proven hard-to-factor semiprime. `http://groups.yahoo.com/group/primeform/message/5449`.

Cohen, H. (1996). *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer, second edition.

Collins, T., Hopkins, D., Langford, S., and Sabin, M. (1997). Public key cryptographic apparatus and method. US Patent #5,848,159.

Coppersmith, D., Howgrave-Graham, N., and Nagaraj, S. V. (2004). Divisors in residue classes, constructively. Cryptology ePrint Archive, Report 2004/339.

Crandall, R. and Pomerance, C. (2000). *Prime Numbers: A Computational Perspective*. Springer-Verlag, New York.

Fiat, A. (1989). Batch RSA. In *Advances in Cryptology: Proceedings of CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 175–185.

Gazzoni Filho, D. L. (2005). Re: Proven hard-to-factor semiprime. `http://groups.yahoo.com/group/primeform/message/5481`.

Goldwasser, S. and Kilian, J. (1986). Almost all primes can be quickly certified. In *Proc. 18th Annual ACM Symp. Theory of Computing*, pages 316–329.

Granlund, T. (2005). GNU MP arithmetic library. `http://www.swox.com/gmp`.

Hinek, M. J. (2002). Low public exponent partial key and low private exponent attacks on multi-prime RSA. Master's thesis, Waterloo University.

Jones, G. A. and Jones, J. M. (1998). *Elementary Number Theory*. Springer Undergraduate Mathematics Series. Springer-Verlag, New York.

Lenstra Jr., H. W. (1987). Factoring integers with elliptic curves. *Ann. Math.*, 2:649–673.

Morain, F. (1998). Primality proving using elliptic curves: an update. In *Alg. Number Theory: Proc. ANTS-III*, volume 1423 of *LNCS*, pages 111–127. Springer-Verlag.

Paixão, C. A. M. (2003). Implementação e análise comparativa de variações do criptossistema RSA. Master's thesis, Inst. de Matemática e Estatística, Univ. de São Paulo.

Quisquater, J.-J. and Couvreur, C. (1982). Fast decipherment algorithm for RSA public-key cryptosystem. *Eletronic Letters*, 18:905–907.

Reble, D. (2005). Untitled mailing list post. `http://www.graysage.com/djr/isp.txt`.

Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126.

Takagi, T. (1998). Fast RSA-type cryptosystem modulo $p^k q$. In *Proceedings of CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 318–326.

Wiener, M. (1990). Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558.