Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring

Jean-Sébastien Coron and Alexander May

Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
jean-sebastien.coron@gemplus.com

Faculty of Computer Science, Electrical Engineering and Mathematics
University of Paderborn
33102 Paderborn, Germany
alexx@uni-paderborn.de

Abstract. We address one of the most fundamental problems concerning the RSA cryptosystem: does the knowledge of the RSA public and secret key-pair (e, d) yield the factorization of $N = pq$ in polynomial time? It is well-known that there is a *probabilistic* polynomial time algorithm that on input (N, e, d) outputs the factors p and q. We present the first *deterministic* polynomial time algorithm that factors N provided that $e, d < \phi(N)$. Our approach is an application of Coppersmith's technique for finding small roots of univariate modular polynomials.

Keywords: RSA, Coppersmith's theorem.

1 Introduction

The most basic security requirement for a public key cryptosystem is that it should be hard to recover the secret key from the public key. To establish this property, one usually identifies a well-known hard problem P and shows that solving P is polynomial-time equivalent to recovering the secret key from the public key.

In this paper we consider the RSA cryptosystem [7]. We denote by $N = pq$ the modulus, product of two primes p and q of the same bit-size. Furthermore, we let e, d be integers such that $e \cdot d = 1 \bmod \phi(N)$, where $\phi(N) = (p-1) \cdot (q-1)$ is Euler's totient function. The public key is then (N, e) and the secret key is (N, d).

It is well known that there exists a *probabilistic* polynomial time equivalence between computing d and factoring N. The proof is given in the original RSA paper by Rivest, Shamir and Adleman [7] and is based on a work by Miller [6].

In this paper, we show that the equivalence can actually be made deterministic, namely we present a *deterministic* polynomial-time algorithm that on input (N, e, d) outputs the factors p and q, provided that $e \cdot d \leq N^2$. Since for standard RSA, the exponents e and d are defined modulo $\phi(N)$, this gives $ed < \phi(N)^2 < N^2$ as required. Our technique is a variant of Coppersmith's theorem for finding small roots of univariate polynomial equations [1], which is based on the LLL lattice reduction algorithm [4]. We also generalize our algorithm to the case of unbalanced prime factors p and q. We obtain that the more imbalanced the prime factors are, the larger is the required upper bound on ed. The paper is an extended version of the paper published by A. May at Crypto 2004 [5].

2 Background on Lattices

Let $u_1, \ldots, u_\omega \in \mathbb{Z}^n$ be linearly independent vectors with $\omega \leq n$. The lattice L spanned by $< u_1, \ldots, u_\omega >$ consists of all integral linear combinations of $u_1, \ldots, u_\omega$, that is:

$$L = \left\{ \sum_{i=1}^{\omega} n_i \cdot u_i | \; n_i \in \mathbb{Z} \right\}$$

Such a set $\{u_1, \ldots, u_\omega\}$ of vectors is called a lattice *basis*. All the bases have the same number of elements, called the *dimension* or *rank* of the lattice. We say that the lattice is full rank if $\omega = n$. Any two bases of the same lattice can be transformed into each other by a multiplication with some integral matrix of determinant ± 1. Therefore, all the bases have the same Gramian determinant $\det_{1 \leq i,j \leq d} < u_i, u_j >$. One defines the *determinant* of the lattice as the square root of the Gramian determinant. If the lattice is full rank, then the determinant of L is equal to the absolute value of the determinant of the $\omega \times \omega$ matrix whose rows are the basis vectors $u_1, \ldots, u_\omega$.

The LLL algorithm [4] computes a short vector in a lattice :

Theorem 1 (LLL). *Let L be a lattice spanned by $(u_1, \ldots, u_\omega)$. The LLL algorithm, given $(u_1, \ldots, u_\omega)$, finds in polynomial time a vector b_1 such that:*

$$\|b_1\| \leq 2^{(\omega-1)/4} \det(L)^{1/\omega}$$

3 The Case of Balanced p and q

In this section, we show the *deterministic* polynomial-time equivalence between recovering d and factoring N, when N is the product of two primes p and q of same bit-size; this is the standard RSA setting. We generalize to an $N = pq$ with unbalanced prime factors in the next section.

Theorem 2. *Let $N = p \cdot q$, where p and q are two prime integers of same bit-size. Let e, d be such that $e \cdot d = 1 \mod \phi(N)$. Then assuming that $e \cdot d \leq N^2$, given (N, e, d) one can recover the factorization of N in deterministic polynomial time.*

Proof. Let $U = e \cdot d - 1$ and $s = p + q - 1$. Our goal is to recover s from N and U. Then given N and s it is straightforward to recover the factorization of N.

First, we assume that we are given the high-order bits s_0 of s. More precisely, we let X be some integer, and write $s = s_0 \cdot X + x_0$, where $0 \le x_0 < X$. The integer s_0 will eventually be recovered by exhaustive search. Moreover, we denote $\phi = \phi(N)$. From $\phi = (p-1) \cdot (q-1) = N - s = N - s_0 \cdot X - x_0$ we obtain the following equations :

$$U = 0 \mod \phi \tag{1}$$

$$x_0 - N + s_0 \cdot X = 0 \mod \phi \tag{2}$$

Let m, k be integers. We consider the polynomials :

$$g_{ij}(x) = x^i \cdot (x - N + s_0 \cdot X)^j \cdot U^{m-j}$$

for $0 \le j \le m$ and $i = 0$, and for $j = m$ and $1 \le i \le k$. Then from equations (1) and (2), we have that for all previous (i, j) :

$$g_{ij}(x_0) = 0 \mod \phi^m$$

Our goal is to find a non-zero integer linear combination $h(x)$ of the polynomials $g_{ij}(x)$, with small coefficients. Then $h(x_0) = 0 \mod \phi^m$, and using the following lemma [3], if the coefficients of $h(x)$ are sufficiently small, then $h(x_0) = 0$ over the integers. Then x_0 can be recovered using any standard root-finding algorithm; eventually from x_0 one recovers the factorization of N. Given a polynomial $h(x) = \sum h_i x^i$, we denote by $\|h(x)\|$ the Euclidean norm of the vector of its coefficients h_i.

Lemma 1 (Howgrave-Graham). *Let $h(x) \in \mathbb{Z}[x]$ which is a sum of at most ω monomials. Suppose that $h(x_0) = 0 \mod \phi^m$ where $|x_0| \le X$ and $\|h(xX)\| < \phi^m/\sqrt{\omega}$. Then $h(x_0) = 0$ holds over the integers.*

Proof. We have :

$$\begin{aligned}
|h(x_0)| &= \left|\sum h_i x_0^i\right| = \left|\sum h_i X^i \left(\frac{x_0}{X}\right)^i\right| \\
&\le \sum \left|h_i X^i \left(\frac{x_0}{X}\right)^i\right| \le \sum \left|h_i X^i\right| \\
&\le \sqrt{\omega}\|h(xX)\| < \phi^m
\end{aligned}$$

Since $h(x_0) = 0 \mod \phi^m$, this gives $h(x_0) = 0$. □

We consider the lattice L spanned by the coefficient vectors of the polynomials $g_{ij}(xX)$. One can see that these coefficient vectors form a triangular basis of a full-rank lattice of dimension $\omega = m + k + 1$ (for an example, see Fig. 1). The determinant of the lattice is then the product of the diagonal entries, which gives :

$$\det L = X^{(m+k)(m+k+1)/2} U^{m(m+1)/2} \tag{3}$$

	1	x	x^2	x^3	x^4	x^5	x^6
$g_{00}(xX)$	U^3						
$g_{01}(xX)$	$*$	U^2X					
$g_{02}(xX)$	$*$	$*$	UX^2				
$g_{03}(xX)$	$*$	$*$	$*$	X^3			
$g_{13}(xX)$		$*$	$*$	$*$	X^4		
$g_{23}(xX)$			$*$	$*$	$*$	X^5	
$g_{33}(xX)$				$*$	$*$	$*$	X^6

Fig. 1. The lattice L of the polynomials $g_{ij}(xX)$ for $k = m = 3$. The symbol '$*$' correspond to non-zero entries whose value is ignored.

Using LLL (theorem 1), one obtains a non-zero short vector b whose norm is guaranteed to satisfy :

$$\|b\| \leq 2^{(\omega-1)/4} \cdot (\det L)^{1/\omega}$$

The vector b is the coefficient vector of some polynomial $h(xX)$ with $\|h(xX)\| = \|b\|$. The polynomial $h(x)$ is then an integer linear combination of the polynomials $g_{ij}(x)$, which implies that $h(x_0) = 0 \mod \phi^m$. In order to apply Lemma 1, it is therefore sufficient to have that :

$$2^{(\omega-1)/4} \cdot (\det L)^{1/\omega} < \frac{\phi^m}{\sqrt{\omega}}$$

Using the inequalities $\sqrt{\omega} \leq 2^{(\omega-1)/2}$, $\phi > N/2$ and $\omega - 1 = m + k \geq m$, we obtain the following sufficient condition :

$$\det L \leq N^{m \cdot \omega} \cdot 2^{-2 \cdot \omega \cdot (\omega - 1)}$$

From equation (3) and inequality $U < N^2$, this gives :

$$X^{(m+k)(m+k+1)/2} \leq N^{m \cdot k} \cdot 2^{-2 \cdot \omega \cdot (\omega-1)}$$

which gives the following condition for X :

$$X \leq \frac{N^{\gamma(m,k)}}{16}, \quad \gamma(m,k) = \frac{2 \cdot m \cdot k}{(m+k) \cdot (m+k+1)}$$

Our goal is to maximize the bound X on x_0, so that as few as possible bits will eventually have to be exhaustively searched. For a fixed m, the function $\gamma(m,k)$ is maximal for $k = m$. The corresponding bound for $k = m$ is then :

$$X \leq \frac{1}{16} \cdot N^{\frac{1}{2} - \frac{1}{4m+2}}. \tag{4}$$

In the following we denote by log the logarithm to the base 2. For an X satisfying the previous inequality, the previous algorithm applies the LLL reduction algorithm on a lattice of dimension $2 \cdot m + 1$ and with entries bounded by $\mathcal{O}(N^{2m})$.

Since the running-time of LLL is polynomial in the dimension and in the size of the entries, given s_0 such that $s = s_0 \cdot X + x_0$ with $0 \leq x_0 < X$, the previous algorithm recovers the factorization of N in time polynomial in $(\log N, m)$.

Finally, taking the greatest integer X satisfying (4), and using $s = p+q-1 \leq 3\sqrt{N}$, we obtain :

$$s_0 \leq \frac{s}{X} \leq 49 \cdot N^{1/(4m+2)}$$

Then, taking $m = \lfloor \log N \rfloor$, we obtain that s_0 is upper-bounded by a constant. The previous algorithm is then run for each possible value of s_0, and the correct s_0 enables to recover the factorization of N, in time polynomial in $\log N$. $\square$

4 Generalization to Unbalanced Prime Factors

The previous algorithm fails when the prime factors p and q are unbalanced, because in this case we have that $s = p + q - 1 \gg \sqrt{N}$. This implies that s is much greater than the bound on X given by inequality (4).

In this section, we provide an algorithm which extends the result of the previous section to unbalanced prime-factors. We use a technique introduced by by Durfee and Nguyen in [2], which consists in using two separate variables x and y for the primes p and q, and replacing each occurrence of $x \cdot y$ by N.

The following theorem shows that the factorization of N given (e, d) becomes easier when the prime factors are imbalanced. Namely, the condition on the product $e \cdot d$ becomes weaker. For example, we obtain that for $p < N^{1/4}$, the modulus N can be factored in polynomial time given (e, d) if $e \cdot d \leq N^{8/3}$ (instead of N^2 for prime factors of equal size).

Theorem 3. *Let β and $0 < \delta \leq 1/2$ be real values, such that $2\beta\delta(1-\delta) \leq 1$. Let $N = p \cdot q$, where p and q are two prime integers such that $p < N^{\delta}$ and $q < 2 \cdot N^{1-\delta}$. Let e, d be such that $e \cdot d = 1 \mod \phi(N)$, and $0 < e \cdot d \leq N^{\beta}$. Then given (N, e, d) one can recover the factorization of N in deterministic polynomial time.*

Proof. Let $U = ed - 1$ as previously. Our goal is to recover p, q from N and U. We have the following equations :

$$U = 0 \mod \phi \tag{5}$$

$$p + q - (N+1) = 0 \mod \phi \tag{6}$$

Let $m \geq 1$, $a \geq 1$ and $b \geq 0$ be integers. We define the following polynomials $g_{ijk}(x, y)$:

$$g_{ijk}(x,y) = x^i \cdot y^j \cdot U^{m-k} \cdot (x + y - (N+1))^k$$

$$\begin{cases} i \in \{0,1\}, & j = 0, & k = 0, \ldots, m \\ 1 < i \leq a, & j = 0, & k = m \\ i = 0, & 1 \leq j \leq b, & k = m \end{cases}$$

In the definition of the polynomials $g_{ijk}(x, y)$, we replace each occurrence of $x \cdot y$ by N; therefore, the polynomials $g_{ijk}(x, y)$ contain only monomials of the form x^r and y^r. From equations (5) and (6), we obtain that (p, q) is a root of $g_{ijk}(x, y)$ modulo ϕ^m, for all previous (i, j, k) :

$$g_{ijk}(p, q) = 0 \mod \phi^m$$

Now, we assume that we are given the high-order bits p_0 of p and the high-order bits q_0 of q. More precisely, for some integers X and Y, we write $p = p_0 \cdot X + x_0$ and $q = q_0 \cdot Y + y_0$, with $0 \le x_0 < X$ and $0 \le y_0 < Y$. The integers p_0 and q_0 will eventually be recovered by exhaustive search.

We define the translated polynomials :

$$t_{ijk}(x, y) = g_{ijk}(p_0 \cdot X + x, q_0 \cdot Y + y)$$

It is easy to see that for all (i, j, k), we have that (x_0, y_0) is a root of $t_{ijk}(x, y)$ modulo ϕ^m :

$$t_{ijk}(x_0, y_0) = 0 \mod \phi^m$$

As in the previous algorithm, our goal is to find a non-zero integer linear combination $h(x, y)$ of the polynomials $t_{ijk}(x, y)$, with small coefficients. Then $h(x_0, y_0) = 0 \mod \phi^m$, and using the following lemma, if the coefficients of $h(x, y)$ are sufficiently small, then $h(x_0, y_0) = 0$ over the integers. Then one can define the polynomial $h'(x) = (p_0 \cdot X + x)^{m+b} \cdot h(x, N/(p_0 \cdot X + x) - q_0 \cdot Y)$. Since $h(x, y)$ is not identically zero and $h(x, y)$ contains only x powers and y powers, the polynomial $h'(x)$ cannot be identically zero. Moreover $h'(x_0) = 0$, which enables to recover x_0 using any standard root-finding algorithm, and eventually the primes p and q. Given a polynomial $h(x, y) = \sum h_{ij} x^i y^j$, we denote by $\|h(x, y)\|$ the Euclidean norm of the vector of its coefficients h_{ij}.

Lemma 2 (Howgrave-Graham). *Let $h(x, y) \in \mathbb{Z}[x, y]$ which is a sum of at most ω monomials. Suppose that $h(x_0, y_0) = 0 \mod \phi^m$ where $|x_0| \le X$, $|y_0| \le Y$ and $\|h(xX, yY)\| < \phi^m/\sqrt{\omega}$. Then $h(x_0, y_0) = 0$ holds over the integers.*

Proof. We have:

$$\begin{aligned} |h(x_0, y_0)| &= \left| \sum h_{ij} x_0^i y_0^i \right| = \left| \sum h_{ij} X^i Y^j \left(\frac{x_0}{X}\right)^i \left(\frac{y_0}{Y}\right)^j \right| \\ &\le \sum \left| h_{ij} X^i Y^j \left(\frac{x_0}{X}\right)^i \left(\frac{y_0}{Y}\right)^j \right| \le \sum \left| h_{ij} X^i Y^j \right| \\ &\le \sqrt{\omega} \|h(xX, yY)\| < \phi^m \end{aligned}$$

Since $h(x_0, y_0) = 0 \mod \phi^m$, this gives $h(x_0, y_0) = 0$. □

We consider the lattice L spanned by the coefficient vectors of the polynomials $t_{ijk}(xX, yY)$. One can see that these coefficient vectors form a triangular basis of a full-rank lattice of dimension $\omega = 2m + a + b + 1$ (for an example,

	1	x	y	x^2	y^2	x^3	y^3	x^4	x^5	y^4
$g_{000}(xX, yY)$	U^3									
$g_{100}(xX, yY)$	$*$	U^3X								
$g_{001}(xX, yY)$	$*$	$*$	U^2Y							
$g_{101}(xX, yY)$	$*$	$*$	$*$	U^2X^2						
$g_{002}(xX, yY)$	$*$	$*$	$*$	$*$	UY^2					
$g_{102}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	UX^3				
$g_{003}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	Y^3			
$g_{103}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	X^4		
$g_{203}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	X^5	
$g_{013}(xX, yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	Y^4

Fig. 2. The lattice L of the polynomials $g_{ijk}(xX, yY)$ for $m = 3$, $a = 2$ and $b = 1$. The symbol '$*$' correspond to non-zero entries whose value is ignored.

see Fig. 2). The determinant of the lattice is then the product of the diagonal entries, which gives :

$$\det L = X^{(m+a)(m+a+1)/2} Y^{(m+b)(m+b+1)/2} U^{m(m+1)} \tag{7}$$

As previously, using LLL, one obtains a non-zero polynomial $h(x, y)$ such that:

$$\|h(xX, yY)\| \le 2^{(\omega-1)/4} \cdot (\det L)^{1/\omega}$$

In order to apply Lemma 2, it is therefore sufficient to have that :

$$2^{(\omega-1)/4} \cdot (\det L)^{1/\omega} < \phi^m/\sqrt{\omega}$$

As in the previous section, using $\sqrt{\omega} \le 2^{(\omega-1)/2}$, $\phi > N/2$ and $\omega - 1 \ge m$, it is sufficient to have :

$$\det L \le N^{m\cdot\omega} \cdot 2^{-2\cdot\omega\cdot(\omega-1)} \tag{8}$$

Let $a = \lfloor (u-1)\cdot m - 1 \rfloor$ and $b = \lfloor (v-1)\cdot m - 1 \rfloor$ for some reals u, v. We obtain that $(m+a)(m+a+1) \le m^2u^2$ and $(m+b)(m+b+1) \le m^2v^2$. We denote $X = N^{\delta_x}$ and $Y = N^{\delta_y}$. From equation (7) we obtain that :

$$\frac{\log(\det L)}{\log N} \le m^2 \cdot \left(\delta_x \cdot \frac{u^2}{2} + \delta_y \cdot \frac{v^2}{2} + \beta\right) + \beta \cdot m \tag{9}$$

Moreover, using $m(u+v) - 3 < \omega \le m(u+v)$, we obtain :

$$\log\left(N^{m\cdot\omega} \cdot 2^{-2\cdot\omega\cdot(\omega-1)}\right) \ge m\left(m(u+v) - 3\right)\log N - 2m^2(u+v)^2 \tag{10}$$

Therefore, we obtain from inequalities (8), (9) and (10) the following sufficient condition :

$$u + v - \delta_x\frac{u^2}{2} - \delta_y\frac{v^2}{2} - \beta \ge \frac{\beta+3}{m} + \frac{2}{\log N}(u+v)^2$$

The function $u \to u - \delta_x \cdot u^2/2$ is maximal for $u = 1/\delta_x$, with a maximum equal to $1/(2\delta_x)$. The same holds for the function $v \to v - \delta_y \cdot v^2/2$. Therefore, taking $u = 1/\delta_x$ and $v = 1/\delta_y$, we obtain the sufficient condition :

$$\frac{1}{2\delta_x} + \frac{1}{2\delta_y} - \beta \geq \frac{\beta+3}{m} + \frac{2}{\log N}\left(\frac{1}{\delta_x} + \frac{1}{\delta_y}\right)^2 \tag{11}$$

For $X = N^{\delta_x}$ and $Y = N^{\delta_y}$ satisfying the previous condition, given p_0 and q_0, the algorithm recovers x_0, y_0 and then p and q in time polynomial in $(m, \log N)$.

In the following, we show that p_0 and q_0 can actually be recovered by exhaustive search, while remaining polynomial-time in $\log N$.

Let ε be such that $0 < \varepsilon \leq \delta/2$. We have the following inequalities :

$$\frac{1}{\delta-\varepsilon} = \frac{1}{\delta(1-\frac{\varepsilon}{\delta})} \geq \frac{1}{\delta}\left(1+\frac{\varepsilon}{\delta}\right) \quad \text{and} \quad \frac{1}{1-\delta-\varepsilon} \geq \frac{1}{1-\delta}\left(1+\frac{\varepsilon}{1-\delta}\right)$$

From $2\beta\delta(1-\delta) \leq 1$, we obtain :

$$2\beta \leq \frac{1}{\delta(1-\delta)} = \frac{1}{\delta} + \frac{1}{1-\delta}$$

which gives :

$$\frac{1}{\delta-\varepsilon} + \frac{1}{1-\delta-\varepsilon} - 2\beta \geq \varepsilon\left(\frac{1}{\delta^2} + \frac{1}{(1-\delta)^2}\right)$$

Therefore, taking $\delta_x = \delta - \varepsilon$ and $\delta_y = 1 - \delta - \varepsilon$, we obtain from (11) the following sufficient condition :

$$\frac{\delta}{2} \geq \varepsilon \geq 2 \cdot \left(\frac{\beta+3}{m} + \frac{2}{\log N}\left(\frac{1}{\delta} + \frac{1}{1-\delta}\right)^2\right)\left(\frac{1}{\delta^2} + \frac{1}{(1-\delta)^2}\right)^{-1}$$

Taking $m = \lfloor \log N \rfloor$, this condition can always be satisfied for large enough $\log N$. Taking the corresponding lower-bound for ε, we obtain $\varepsilon = \mathcal{O}(1/\log N)$, which gives $N^\varepsilon \leq C$ for some constant C. Therefore, we obtain that p_0 and q_0 are upper-bounded by the constants C and $2C$:

$$p_0 \leq \frac{p}{X} \leq N^{\delta-\delta_x} \leq N^\varepsilon \leq C$$

$$q_0 \leq \frac{q}{Y} \leq 2N^{1-\delta-\delta_y} \leq 2N^\varepsilon \leq 2C$$

This shows that p_0 and q_0 can be recovered by exhaustive search. The total running-time is still polynomial in $\log N$. $\square$

5 Practical Experiments

We have implemented the two algorithms of sections 3 and 4, using Shoup's NTL library [8]. First, we describe in Table 1 the experiments with prime factors of equal bit-size, with $e \cdot d \simeq N^2$. We assume that we are given the ℓ high-order bits of s; the observed running time for a single execution of LLL is denoted by t. The total running time for factoring N is then estimated as $T \simeq 2^\ell \cdot t$. We obtain that the factorization would take a few days for a 512-bit modulus, and a few years for a 1024-bit modulus.

N	bits given	dimension	t	T
512 bits	14 bits	21	70 s	13 days
512 bits	10 bits	29	7 min	5 days
512 bits	9 bits	33	16 min	5 days
1024 bits	26 bits	21	7 min	900 years
1024 bits	19 bits	29	40 min	40 years
1024 bits	17 bits	33	90 min	23 years

Table 1. Bit-size of N, number of bits to be exhaustively searched, lattice dimension, observed running-time for a single LLL-reduction t, and estimated total running-time T, with $e \cdot d \simeq N^2$. The experiments were performed on a 1.6 GHz PC running under Windows 2000/Cygwin.

The experiments with prime factors of unbalanced size with $e \cdot d \simeq N^2$ are summarized in Table 2. In this case, it was not necessary to know the high-order bits of p and q, and one recovers the factorization of N after a single application of LLL. The table shows that the factorization of N is easier when the prime factors are unbalanced.

N	δ	dimension	t
512 bits	0.25	16	2 s
512 bits	0.3	29	2 min
1024 bits	0.25	16	15 s
1024 bits	0.3	29	10 min

Table 2. Bit-size of the RSA modulus N such that $p < N^\delta$, lattice dimension, observed running-time for factoring N, with $e \cdot d \simeq N^2$. The experiments were performed on a 1.6 GHz PC running under Windows 2000/Cygwin.

6 Conclusion

We have shown the first *deterministic* polynomial time algorithm that factors an RSA modulus N given the pair of public and secret exponents e and d, provided that $e \cdot d < N^2$. The algorithm is a variant of Coppersmith's technique for finding small roots of univariate modular polynomial equations. We have also generalized our algorithm to the case of unbalanced prime factors.

References

1. D. Coppersmith, "Small solutions to polynomial equations and low exponent vulnerabilities", Journal of Cryptology, Vol. 10(4), pp. 223–260, 1997.
2. G. Durfee and P. Nguyen "Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt'99", Proceedings of Asiacrypt 2000.
3. N. Howgrave-Graham, "Finding small roots of univariate modular equations revisited", Proceedings of Cryptography and Coding, Lecture Notes in Computer Science Vol. 1355, Springer-Verlag, pp. 131–142, 1997
4. A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients", Mathematische Annalen, Vol. 261, pp. 513–534, 1982
5. A. May, "Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring", Proceedings of Crypto 2004, LNCS Vol. 3152, pp. 213-219.
6. G. L. Miller, "Riemann's hypothesis and tests for primality", Seventh Annual ACM Symposium on the Theory of Computing, pp. 234–239, 1975
7. R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21(2), pp.120–126, 1978
8. V. Shoup, NTL: A Library for doing Number Theory, available online at `http://www.shoup.net/ntl/index.html`