J. Cryptology (2007) 20: 39–50
DOI: 10.1007/s00145-006-0433-6

Journal of
CRYPTOLOGY

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

Jean-Sébastien Coron
University of Luxembourg,
162a avenue de la Faiencerie, L-1511 Luxembourg
coron@clipper.ens.fr

Alexander May
Department of Computer Science, TU Darmstadt,
64289 Darmstadt, Germany
may@informatik.tu-darmstadt.de

Communicated by Dan Boneh

Received 30 August 2004 and revised 3 January 2006
Online publication 6 October 2006

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 given (e, d) provided that $e, d < \varphi(N)$. Our approach is an application of Coppersmith's technique for finding small roots of univariate modular polynomials.

Key words. 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 recovering the secret key from the public key is polynomial-time equivalent to solving P.

In this paper we consider the RSA cryptosystem [11]. We denote by $N = pq$ the modulus, product of two primes p and q of the same bit-size. Furthermore, we denote by e, d the public and private exponents, such that $e \cdot d = 1 \bmod \varphi(N)$, where $\varphi(N) = (p-1) \cdot (q-1)$ 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 [11] and is based on a work by Miller [8].

In this paper we show that the equivalence can actually be made deterministic, namely we present the first *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 $\varphi(N)$, we have that $ed < \varphi(N)^2 < N^2$ as required. Our result is mainly of theoretical interest, since our deterministic algorithm is much less efficient than the probabilistic one. However, we also present an algorithm that recovers the factors p and q deterministically in time $\mathcal{O}(\log^2 N)$ when $e \cdot d \leq N^{3/2}$; this happens when e is small and $d < \varphi(N)$, which is common in practice.

Our technique is a variant of Coppersmith's theorem for finding small roots of univariate polynomial equations [2]. Coppersmith's theorem is based on the LLL lattice reduction algorithm [6], and has found numerous applications in cryptanalysis (see [10] for a survey). We use a variant in which one considers polynomials modulo an unknown integer (instead of the known modulus). This variant was introduced by Boneh et al. in [1] for factoring moduli of the form $p^r q$ in polynomial time for large r. This approach was also used by Howgrave-Graham in [5] to compute approximate integer common divisors. Our technique is actually a direct application of Howgrave-Graham's algorithm, but for completeness we also provide a full description of our algorithm.

This article is an extended version of a paper published by May [7] at Crypto 2004. The difference with [7] is that our analysis is based on univariate modular polynomials instead of bivariate integer polynomials, which leads to a simpler algorithm. Moreover, we generalize our analysis to the case of unbalanced prime factors p and q. Quite expectedly, we obtain that the upper bound on ed gets larger when the prime factors are more imbalanced. For example, if $p < N^{1/4}$, then the modulus N can be factored in polynomial time given (e, d) for $e \cdot d \leq N^{8/3}$ (instead of N^2 for prime factors of equal size).

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 $\langle u_1, \ldots, u_\omega \rangle$ 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 $\det_{1 \leq i, j \leq d} \langle u_i, u_j \rangle$. 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 [6] computes a short vector in a lattice:

Theorem 1 (LLL). *Let L be a lattice spanned by $(u_1, \ldots, u_\omega) \in \mathbb{Z}^n$, where the Euclidean norm of each of the vectors $u_1, \ldots, u_\omega$ is bounded by B. Given $(u_1, \ldots, u_\omega)$,* *the LLL algorithm finds a vector b_1 such that*

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

in time $\mathcal{O}(\omega^5 n \log^3 B)$

In order to improve the complexity of our algorithm, we use an improved version of LLL, called the L^2 algorithm and due to Nguyen and Stehlé [9]. The L^2 algorithm achieves the same bound on $\|b_1\|$ but in time $\mathcal{O}(\omega^4 n(\omega + \log B) \log B)$.

3. An Algorithm for $ed \leq N^{3/2}$

In this section we consider the standard RSA setting, i.e. we assume that N is the product of two different prime factors p, q of the same bit-size. We also assume that $ed \leq N^{3/2}$. This is a practical case since for RSA one generally uses a small public exponent e (for example, $e = 3$ or $e = 2^{16} + 1$). The following theorem shows that the factorization of N can then be recovered in deterministic time $\mathcal{O}(\log^2 N)$:

Theorem 2. *Let $N = p \cdot q$, where p and q are two prime integers of the same bit-size. Let e, d be such that $e \cdot d = 1 \bmod \varphi(N)$. Then if $1 < e \cdot d \leq N^{3/2}$, there is a deterministic algorithm that given (N, e, d) recovers the factorization of N in time $\mathcal{O}(\log^2 N)$.*

Proof. In the following we assume without loss of generality that $p < q$, which implies

$$p < N^{1/2} < q < 2p < 2N^{1/2}.$$

This gives the following useful estimates:

$$p + q < 3N^{1/2} \quad \text{and} \quad \varphi(N) = N + 1 - (p + q) > \tfrac{1}{2}N. \tag{1}$$

We denote by $\lceil k \rceil$ the smallest integer greater than or equal to k. Furthermore, we denote by $\mathbb{Z}^*_{\varphi(N)}$ the group of invertible integers modulo $\varphi(N)$.

Since $ed = 1 \bmod \varphi(N)$, we know that

$$ed = 1 + k\varphi(N) \qquad \text{for some} \quad k \in \mathbb{N}.$$

We show that k can be recovered up to a small constant when $ed \leq N^{3/2}$. Namely, we define $\tilde{k} = (ed - 1)/N$ as an underestimate of k and we observe that

$$\begin{aligned} k - \tilde{k} &= \frac{ed - 1}{\varphi(N)} - \frac{ed - 1}{N} \\ &= \frac{N(ed - 1) - (N - p - q + 1)(ed - 1)}{\varphi(N)N} \\ &= \frac{(p + q - 1)(ed - 1)}{\varphi(N)N}. \end{aligned}$$

Using (1) we conclude that

$$k - \tilde{k} < 6N^{-3/2}(ed - 1). \tag{2}$$

Then since $ed \leq N^{3/2}$, we obtain that $0 < k - \tilde{k} < 6$. Thus, one of the six values $\lceil \tilde{k} \rceil + i$, $i = 0, 1, \ldots, 5$, must be equal to k. We can test these six candidates successively and for the right choice k, we can compute

$$N + 1 + \frac{1 - ed}{k} = p + q,$$

from which one recovers the factorization of N. Our approach uses only elementary arithmetic on integers of bit-size $\mathcal{O}(\log(N))$. Thus, the running time is $\mathcal{O}(\log^2 N)$, which concludes the proof of the theorem. □

4. The Case of $ed \leq N^2$

As in the previous section, we assume that N is the product of two primes p and q of same bit-size, but here we only assume that $ed \leq N^2$. Under this assumption, we show the *deterministic* polynomial-time equivalence between recovering d and factoring N. We will generalize to an $N = pq$ with unbalanced prime factors in the next section.

Theorem 3. *Let $N = p \cdot q$, where p and q are two prime integers of the same bit-size. Let e, d be such that $e \cdot d = 1 \bmod \varphi(N)$. Then if $1 < e \cdot d \leq N^2$, there is a deterministic algorithm that given (N, e, d) recovers the factorization of N in time $\mathcal{O}(\log^9 N)$.*

Proof. Our technique is a direct application of Howgrave-Graham's algorithm for approximate integer common divisors [5]. Given two integers $a < b$ and $M = b^\alpha$ for some $\alpha \in [0, 1]$, Howgrave-Graham's algorithm outputs all integers $d > M$ dividing both $a + x_0$ and b for some $|x_0| < X$, in time polynomial in $\log b$, where $X = b^\beta$ and $\beta = \alpha^2$.

Letting $U = e \cdot d - 1$ and $s = p + q - 1$, our goal is to recover s from N and U. Then from s it is straightforward to recover the factorization of N. From $U = 0 \mod \varphi(N)$ and $\varphi(N) = (p - 1)(q - 1) = N - s$, we observe that $N - s$ divides both U and $N - s$. Therefore, one can apply Howgrave-Graham's algorithm with $a := N$, $b := U$, $x_0 := -s$ and $M = N/2$. We have that $\alpha \simeq \frac{1}{2}$ and $\beta \simeq \frac{1}{4}$, which enables to recover s and eventually the factorization of N.

In the following, for completeness, we provide the full description of an algorithm for factoring N given (e, d), similar to Howgrave-Graham's algorithm. 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 $0 \leq x_0 < X$. The integer s_0 will eventually be recovered by exhaustive search. Moreover, we denote $\varphi = \varphi(N)$. From $\varphi = (p - 1) \cdot (q - 1) = N - s = N - s_0 \cdot X - x_0$ we obtain the following equations:

$$U = 0 \mod \varphi, \tag{3}$$

$$x_0 - N + s_0 \cdot X = 0 \mod \varphi. \tag{4}$$

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 $1 \le i \le k$, where m, k are fixed parameters. From (3) and (4), we have that for all previous (i, j),

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

For any linear integer combination $h(x)$ of the polynomials $g_{ij}(x)$, we have that $h(x_0) = 0 \bmod \varphi^m$. Our goal is then to find a non-zero $h(x)$ with small coefficients. Namely, using the following lemma from [4], if the coefficients of $h(x)$ are sufficiently small, we have that $h(x_0) = 0$ holds over the integers. The integer x_0 can then 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 4 (Howgrave-Graham). *Let $h(x) \in \mathbb{Z}[x]$ be the sum of at most ω monomials. Suppose that $h(x_0) = 0 \bmod \varphi^m$ where $|x_0| \le X$ and $\|h(xX)\| < \varphi^m/\sqrt{\omega}$. Then $h(x_0) = 0$ holds over the integers.*

Proof. We have

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

Since $h(x_0) = 0 \bmod \varphi^m$, 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

$$\det L = X^{(m+k)(m+k+1)/2} U^{m(m+1)/2}. \tag{5}$$

	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 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 $h(x_0) = 0 \bmod \varphi^m$. In order to apply Lemma 4, it is therefore sufficient to have that

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

Using the inequalities $\sqrt{\omega} \leq 2^{(\omega-1)/2}$, $\varphi > N/2$ 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 (5) 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:

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

Our goal is to maximize the bound X on x_0, so that fewer bits must 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

$$X \leq \tfrac{1}{16} \cdot N^{1/2-1/(4m+2)}. \tag{6}$$

The LLL algorithm is therefore applied on a lattice of dimension $\omega = m + k + 1 = 2\cdot m + 1$ and with entries bounded by $B = \mathcal{O}(N^{2m})$. Since the running time of LLL is polynomial in the lattice 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 (6), and using $s = p + q - 1 \leq 3N^{1/2}$, 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 us to recover the factorization of N. The running time is dominated by the time it takes to run LLL on a lattice of dimension $\omega = 2m + 1$ with entries bounded by $B = \mathcal{O}(N^{2m})$. Since the running time of LLL is bounded by $\mathcal{O}(\omega^6 \log^3 B)$, our algorithm recovers the factorization of N in time $\mathcal{O}(\log^{12} N)$. If one uses the L^2 variant instead of LLL, one obtains a running time of $\mathcal{O}(\log^9 N)$. $\square$

5. 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 $s = p + q - 1 \gg \sqrt{N}$, and s is then much greater than the bound on X given by inequality (6).

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 Durfee and Nguyen in [3], 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. We note that Howgrave-Graham's algorithm for finding approximate integer common divisors does not seem to apply in this case.

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 $e \cdot d \le N^{8/3}$ (instead of N^2 for prime factors of equal size).

Theorem 5. *Let β and $0 < \delta \le \frac{1}{2}$ be real values, such that $2\beta\delta(1-\delta) \le 1$. 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 $e \cdot d = 1 \bmod \varphi(N)$, and $1 < e \cdot d \le N^{\beta}$. Then there is a deterministic algorithm that given (N, e, d) recovers the factorization of N in time $\mathcal{O}(\log^9 N)$.*

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

$$U = 0 \mod \varphi, \tag{7}$$

$$p + q - (N+1) = 0 \mod \varphi. \tag{8}$$

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

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

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

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 that are powers of x or powers of y. From (7) and (8), we obtain that (p, q) is a root of $g_{ijk}(x, y)$ modulo φ^m, for all previous (i, j, k):

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

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:

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

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 $h(x_0, y_0) = 0 \mod \varphi^m$, and, using again Howgrave-Graham's 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 us 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 6 (Howgrave–Graham). *Let $h(x, y) \in \mathbb{Z}[x, y]$ which is the sum of at most ω monomials. Suppose that $h(x_0, y_0) = 0 \bmod \varphi^m$ where $|x_0| \le X$, $|y_0| \le Y$ and $\|h(xX, yY)\| < \varphi^m/\sqrt{\omega}$. Then $h(x_0, y_0) = 0$ holds over the integers.*

Proof. We have

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

Since $h(x_0, y_0) = 0 \bmod \varphi^m$, 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, see Fig. 2). The determinant of the lattice is then the product of the diagonal entries, which gives

$$\det L = X^{(m+a)(m+a+1)/2} Y^{(m+b)(m+b+1)/2} U^{m(m+1)}. \tag{9}$$

As previously, using lattice reduction, 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 6, it is therefore sufficient to have that

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

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

$$\det L \le N^{m \cdot \omega} \cdot 2^{-2 \cdot \omega \cdot (\omega-1)}. \tag{10}$$

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

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

We write $a = \lfloor (u-1) \cdot m - 1 \rfloor$ and $b = \lfloor (v-1) \cdot m - 1 \rfloor$ for some reals u, v. We obtain that $(m+a)(m+a+1) \leq m^2u^2$ and $(m+b)(m+b+1) \leq m^2v^2$. We write $X = N^{\delta_x}$ and $Y = N^{\delta_y}$ for some reals δ_x, δ_y. From (9) and $U \leq N^{\beta}$ we obtain that

$$\frac{\log(\det L)}{\log N} \leq m^2 \cdot \left(\delta_x \cdot \frac{u^2}{2} + \delta_y \cdot \frac{v^2}{2} + \beta \right) + \beta \cdot m, \tag{11}$$

where log denotes the logarithm in base 2. Moreover, using $m(u+v)-3 < \omega \leq m(u+v)$, we have

$$\log(N^{m \cdot \omega} \cdot 2^{-2 \cdot \omega \cdot (\omega - 1)}) \geq m\,(m(u+v) - 3) \log N - 2m^2(u+v)^2. \tag{12}$$

Therefore, combining inequalities (10), (11) and (12), we obtain the following sufficient condition:

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

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

$$\frac{1}{2\delta_x} + \frac{1}{2\delta_y} - \beta \geq \frac{\beta + 3}{m} + \frac{2}{\log N} \left(\frac{1}{\delta_x} + \frac{1}{\delta_y} \right)^2. \tag{13}$$

For $X = N^{\delta_x}$ and $Y = N^{\delta_y}$ satisfying the previous condition and given p_0 and q_0 such that $p = p_0 \cdot X + x_0$ and $q = q_0 \cdot Y + y_0$, the algorithm recovers x_0, y_0 and then p, 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:

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

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

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

Combining the three previous inequalities, we get

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

Therefore, taking $\delta_x = \delta - \varepsilon$ and $\delta_y = 1 - \delta - \varepsilon$, we obtain from (13) the following sufficient condition:

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

Moreover, since $0 < \varepsilon \leq \delta/2$ and $\delta < \frac{1}{2}$, we have

$$\frac{1}{\delta-\varepsilon} \leq \frac{2}{\delta} \quad \text{and} \quad \frac{1}{1-\delta-\varepsilon} \leq 4.$$

Therefore, this gives the following sufficient condition:

$$\frac{\delta}{2} \geq \varepsilon \geq 2 \cdot \left(\frac{\beta+3}{m}+\frac{2}{\log N}\left(\frac{2}{\delta}+4\right)^2\right)\left(\frac{1}{\delta^2}+\frac{1}{(1-\delta)^2}\right)^{-1}.$$

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 while remaining polynomial time in $\log N$. The total running time of our algorithm is then dominated by running the lattice reduction algorithm on a lattice basis of dimension $\omega = \mathcal{O}(m)$ and entries bounded by $B = N^{\mathcal{O}(m)}$. Therefore, using LLL, our algorithm recovers the factorization of N in time $\mathcal{O}(\log^{12} N)$. If one uses the L^2 variant instead of LLL, one obtains a running time of $\mathcal{O}(\log^9 N)$. □

6. Practical Experiments

We have implemented the two algorithms of Sections 4 and 5, using the LLL implementation of Shoup's NTL library [12]. 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 $s = p + q$; 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 $T \simeq 2^{\ell} \cdot t$.

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, when $e \cdot d \simeq N^2$. The experiments were performed on a 1.6 GHz PC running under Windows 2000/Cygwin.

N (bits)	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

We obtain that the factorization of N given (e, d) would take a few days for a 512-bit modulus, and a few years for a 1024-bit modulus. This contrasts with Miller's algorithm whose running time is only a fraction of a second for a 1024-bit modulus.

The experiments with prime factors of unbalanced size and 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 $s = p + q$, and one recovers the factorization of N after a single application of LLL. The results in Table 2 confirm that the factorization of N is easier when the prime factors are unbalanced.

7. 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 provided a generalization to the case of unbalanced prime factors. Finally, we note that the problem of the deterministic polynomial-time equivalence between finding d and factoring N is not entirely solved in this paper, because finding an algorithm for $e \cdot d > N^2$ remains an open problem.

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

N (bits)	δ	Dimension	t
512	0.25	16	2 s
512	0.3	29	2 min
1024	0.25	16	15 s
1024	0.3	29	10 min

References

[1] D. Boneh, G. Durfee and N.A. Howgrave-Graham, Factoring $n = p^r q$ for large r, *Proceedings of Crypto* '99, pp. 326–337. LNCS, Vol. 1666. Springer-Verlag, Berlin, 1999.

[2] D. Coppersmith, Small solutions to polynomial equations and low exponent vulnerabilities, *Journal of Cryptology*, Vol. 10, No. 4, pp. 223–260, 1997.

[3] G. Durfee and P. Nguyen, Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt '99, *Proceedings of Asiacrypt* 2000, pp. 14–29. LNCS, Vol. 1976. Springer-Verlag, Berlin, 2000.

[4] N. Howgrave-Graham, Finding small roots of univariate modular equations revisited, *Proceedings of Cryptography and Coding*, pp. 131–142. LNCS, Vol. 1355. Springer-Verlag, Berlin, 1997.

[5] N. Howgrave-Graham, Approximate integer common divisors, *Proceedings of CALC* '01, pp. 51–66. LNCS, Vol. 2146. Springer-Verlag, Berlin, 2001.

[6] A. K. Lenstra, H. W. Lenstra and L. Lovász, Factoring polynomials with rational coefficients, *Mathematische Annalen*, Vol. 261, pp. 513–534, 1982

[7] A. May, Computing the RSA secret key is deterministic polynomial time equivalent to factoring, *Proceedings of Crypto* 2004, pp. 213–219. LNCS, Vol. 3152. Springer-Verlag, Berlin, 2004.

[8] G. L. Miller, Riemann's hypothesis and tests for primality, *Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing*, pp. 234–239, 1975.

[9] P. Nguyen and D. Stehlé, Floating-point LLL revisited, *Proceedings of Eurocrypt* 2005, pp. 215–233. LNCS, Vol. 3494. Springer-Verlag, Berlin, 2005.

[10] P.Q. Nguyen and J. Stern, The two faces of lattices in cryptology, *Proceedings of CALC* '01, pp. 146–180. LNCS, Vol. 2146. Springer-Verlag, Berlin, 2001.

[11] 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

[12] V. Shoup, NTL: A Library for Doing Number Theory, available online at `http://www.shoup.net/ntl/index.html`