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

Subhamoy Maitra and Santanu Sarkar

Applied Statistics Unit, Indian Statistical Institute,
203 B T Road, Kolkata 700 108, India.
{subho, santanu_r}@isical.ac.in

Abstract. Let $N = pq$ be the product of two large primes. Consider CRT-RSA with the public encryption exponent e and private decryption exponents d_p, d_q. It is well known that given any one of d_p or d_q (or both) one can factorize N in probabilistic poly$(\log N)$ time with success probability almost equal to 1. Though this serves all the practical purposes, from theoretical point of view, this is not a deterministic polynomial time algorithm. In this paper, we present a lattice based deterministic poly$(\log N)$ time algorithm that uses both d_p, d_q (in addition to the public information e, N) to factorize N.

Ketwords: CRT-RSA, Cryptanalysis, Factorization, LLL Algorithm, RSA.

1 Introduction

RSA [19] is one of the most popular cryptosystems in the history of cryptology. Let us first briefly describe the idea of RSA as follows:

- primes p, q, with $q < p < 2q$;
- $N = pq$, $\phi(N) = (p-1)(q-1)$;
- e, d are such that $ed = 1 + k\phi(N)$, $k \geq 1$;
- N, e are publicly available and message M is encrypted as $C \equiv M^e \bmod N$;
- the secret key d is required to decrypt the cipher as $M \equiv C^d \bmod N$.

The study of RSA is one of the most attractive areas in cryptology research as evident from many excellent works (one may refer [2, 11, 17] and the references therein for detailed information). The paper [19] itself presents a probabilistic polynomial time algorithm that on input N, e, d provides the factorization of N; this is based on the technique provided by [18]. One may also have a look at [20, Page 197]. Recently in [16, 8] it has been proved that given N, e, d, one can factor N in deterministic polynomial time provided $ed \leq N^2$.

Speeding up RSA encryption and decryption is of serious interest and for large N as both e, d cannot be small at the same time. For fast encryption, it is possible to use smaller e and e as small as $2^{16} + 1$ is widely believed to be a good candidate. For fast decryption, the value of d needs to be small. However, Wiener [21] showed that for $d < \frac{1}{3}N^{\frac{1}{4}}$, N can be factor easily. Later, Boneh-Durfee [3] increased this bound up to $d < N^{0.292}$. Thus use of smaller d is in general not recommendable. In this direction, an alternative approach has been proposed by Wiener [21] exploiting the Chinese Remainder Theorem (CRT) for faster decryption. The idea is as follows:

- the public exponent e and the private CRT exponents d_p and d_q are used satisfying $ed_p \equiv 1 \bmod (p-1)$ and $ed_q \equiv 1 \bmod (q-1)$;
- the encryption is same as standard RSA;
- to decrypt a ciphertext C one needs to compute $M_1 \equiv C^{d_p} \bmod p$ and $M_2 \equiv C^{d_q} \bmod q$;
- using CRT, one can get the message $M \in \mathbb{Z}^n$ such that $M \equiv M_1 \bmod p$ and $M \equiv M_2 \bmod q$.

This variant of RSA is popularly known as CRT-RSA.

Given N, e and any one of d_p, d_q (or both), there exists well known solution to factorize N in probabilistic poly$(\log N)$ time with probability almost 1. In practice, this study is enough from the security point of view for CRT-RSA. However, from theoretical point of view, getting a deterministic polynomial time algorithm for factorization of N with the knowledge of N, e, d_p, d_q is important and we solve it using lattice based technique.

Without loss of generality, consider d_p is available. One can take any random integer W in $[2, N-1]$ and then $\gcd(W^{ed_p} - W, N)$ provides p with a probability almost equal to 1. The strategy fails, when q too divides $W^{ed_p} - W$ and this happens with negligible probability. Knowing both d_p, d_q presents the similar scenario.

Another important work in this direction shows that with the availability of decryption oracle under a fault model, one factorize N in poly$(\log N)$ time [4, Section 2.2] and the idea has been improved by A. Lenstra [4, Section 2.2, Reference 16].

The organization of this paper is as follows. Some preliminaries are discussed in Section 2. The lattice based technique is used in Section 3 to show that one can factorize N in deterministic polynomial time from the knowledge of N, e, d_p, d_q. Section 4 concludes the paper.

2 Preliminaries

Let us present some basics in lattice reduction techniques. Consider the linearly independent vectors $u_1, \ldots, u_\omega \in \mathbb{Z}^n$, when $\omega \leq n$. A lattice, spanned by $< u_1, \ldots, u_\omega >$, is the set of all linear combinations of $u_1, \ldots, u_\omega$, i.e., ω is the dimension of the lattice. A lattice is called full rank when $\omega = n$. Let L be a lattice spanned by the linearly independent vectors $u_1, \ldots, u_\omega$, where $u_1, \ldots, u_\omega \in \mathbb{Z}^n$. By $u_1^*, \ldots, u_\omega^*$, we denote the vectors obtained by applying the Gram-Schmidt process to the vectors $u_1, \ldots, u_\omega$.

The determinant of L is defined as $\det(L) = \prod_{i=1}^{w} ||u_i^*||$, where $||.||$ denotes the Euclidean norm on vectors. Given a polynomial $g(x, y) = \sum a_{i,j} x^i y^j$, we define the Euclidean norm as $\| g(x, y) \| = \sqrt{\sum_{i,j} a_{i,j}^2}$ and infinity norm as $\| g(x, y) \|_\infty = \max_{i,j} |a_{i,j}|$.

It is known that given a basis $u_1, \ldots, u_\omega$ of a lattice L, the LLL algorithm [14] can find a new basis $b_1, \ldots, b_\omega$ of L with the following properties.

- $\| b_i^* \|^2 \leq 2 \| b_{i+1}^* \|^2$, for $1 \leq i < \omega$.
- For all i, if $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ then $|\mu_{i,j}| \leq \frac{1}{2}$ for all j.
- $\| b_i \| \leq 2^{\frac{\omega(\omega-1)+(i-1)(i-2)}{4(\omega-i+1)}} \det(L)^{\frac{1}{\omega-i+1}}$ for $i = 1, \ldots, \omega$.

In [5], deterministic polynomial time algorithms have been presented to find small integer roots of (i) polynomials in a single variable mod N, and of (ii) polynomials in two variables over the integers. The idea of [5] extends to more than two variables also, but in that event, the method becomes probabilistic.

Theorem 1. *[5] Let $p(x,y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X, Y be the bounds on the desired solutions x_0, y_0. Define $p'(x,y) = p(xX, yY)$ and let W be the absolute value of the largest coefficient of p'. Given $XY \leq W^{\frac{2}{3\delta}}$, one can find all integer pairs (x_0, y_0) with $p(x_0, y_0) = 0$, $x_0 \leq X$ and $y_0 \leq Y$ in time polynomial in $(\log W, 2^\delta)$.*

In [6], a simpler algorithm than [5] has been presented in this direction, but it was asymptotically less efficient. Later, in [7], a simpler idea than [5] has been presented with the same asymptotic bound as in [5]. Both the works [6, 7] depend on the following theorem.

Theorem 2. *[9] Let $f(x,y) \in \mathbb{Z}[x,y]$ which is a sum of at most w monomials. Suppose that $f(x_0, y_0) \equiv 0 \bmod (N)$ where $|x_0| \leq X$ and $|y_0| \leq Y$ and $||f(xX, yY)|| < \frac{N}{\sqrt{w}}$. Then $f(x_0, y_0)$ holds over integer.*

The work of [16], in finding the deterministic polynomial time algorithm to factorize N from the knowledge of e, d, uses the techniques presented in [5, 6]. Further, the work of [8] exploits the technique presented in [10].

3 Deterministic Polynomial Time Algorithm

In this section we consider that both d_p, d_q are known apart from the public information N, e. In the next result, we use the idea of [5].

Theorem 3. *Let $e < \phi(N)$, $d_p < (p-1)$ and $d_q < (q-1)$. If N, e, d_p, d_q are known then N can be factored in deterministic polynomial time in $\log N$.*

Proof. We can write $ed_p = 1 + k(p-1)$ and $ed_q = 1 + l(q-1)$ where k, l are positive integers. So we can write $ed_p + k - 1 = kp$ and $ed_q + l - 1 = lq$. Now multiplying these we get $(ed_p - 1)(ed_q - 1) + k(ed_q - 1) + l(ed_p - 1) + kl = kplq$. Substituting k, l by x, y respectively, we get the equation $(ed_p - 1)(ed_q - 1) + x(ed_q - 1) + y(ed_p - 1) + xy = Nxy$. Thus, we have to find the roots (x_0, y_0) of $f(x,y) = (1-N)xy + x(ed_q - 1) + y(ed_p - 1) + (ed_p - 1)(ed_q - 1) = 0$.

As p, q are not known, we need some estimate of p, q. Assume $p = N^{\gamma_1}$, $q = N^{\gamma_2}$, where $\gamma_1 + \gamma_2 = 1$. If p, q are of same bit size, we consider $\gamma_1 = \gamma_2 = \frac{1}{2}$. Otherwise, we estimate p, q are of different bit sizes, such that $pq = N$. As the number of bits in p is $\log_2 p$, we need to try at most $\log N$ many estimates for the bit size of p and run the strategy as described below that many times.

Let $e = N^\alpha$, $d_p = N^{\delta_1}$ and $d_q = N^{\delta_2}$. Let us denote $X = N^{\alpha + \delta_1 - \gamma_1}$ and $Y = N^{\alpha + \delta_2 - \gamma_2}$. Clearly one can take (X, Y) as upper bounds of the root (k, l) of f neglecting the constant terms.

Let $W = ||f(xX, yY)||_\infty \geq (ed_p - 1)(ed_q - 1) \approx e^2 d_p d_q$. Following Theorem 1 [5], one can find the root of f in deterministic polynomial time in $\log N$ (as the degree of the polynomial f is 1) if $XY < W^{\frac{2}{3}}$. Thus we need $kl < (e^2 d_p d_q)^{\frac{2}{3}}$ to get the root of f, which is proved below. Thus it guarantees the one can factor N from the knowledge of N, e, d_p, d_q in deterministic polynomial time in $\log N$.

- We have $ed_p > k(p-1)$ and $ed_q > l(q-1)$. So $e^2 d_p d_q > kl(p-1)(q-1)$, i.e., $(e^2 d_p d_q)^{\frac{2}{3}} > (kl(p-1)(q-1))^{\frac{2}{3}}$.
- Thus, to show that $kl < (e^2 d_p d_q)^{\frac{2}{3}}$, we need to prove, $kl < (kl(p-1)(q-1))^{\frac{2}{3}}$, i.e., $kl < (p-1)^2(q-1)^2$.
- Since we assume $d_p < (p-1), d_q < (q-1)$, we have $e > k$ and $e > l$, i.e., $e^2 > kl$. As we take $\phi(N) = (p-1)(q-1) > e$, we get that $(p-1)^2(q-1)^2 > kl$.

This concludes the proof. □

Let us now present some experimental results in Table 1. Our experiments are based on the strategy of [6] as it is easier to implement. According to the formula presented in [6, Theorem 4], the lattice dimension (denote it by LD) is $(\delta + m + 1)^2$, where δ is the degree of the polynomial f (here $\delta = 1$) and m is a non-negative integer related to the shifts of the polynomial (in the proof of [6, Theorem 4], this m is denoted by k). We have written the programs in SAGE 3.1.1 over Linux Ubuntu 8.04 on a computer with Dual CORE Intel(R) Pentium(R) D CPU 1.83 GHz, 2 GB RAM and 2 MB Cache. We take large primes p, q such that N is of 1000 bits. As we experiment with low lattice dimensions, we cannot demonstrate the success of the experiments when d_p, d_q are of the order of p, q respectively.

N	p	q	e	d_p	d_q	LD	m	L^3-time
1000 bit	500 bit	500 bit	1000 bit	240 bit	240 bit	16	2	14.82 sec
1000 bit	400 bit	600 bit	1000 bit	230 bit	265 bit	16	2	16.09 sec
1000 bit	500 bit	500 bit	1000 bit	350 bit	350 bit	49	5	5914.08 sec

Table 1. Experimental results corresponding to Theorem 3.

Now we present a more general form of Theorem 3. The constraints in Theorem 3 are $\alpha < 1, d_p < p-1, d_q < q-1$. In Theorem 4 we try to get rid of these constraints, but naturally that impose some other conditions.

Theorem 4. *Let $e = N^\alpha, d_p \leq N^{\delta_1}, d_q \leq N^{\delta_2}$. Suppose p is estimated[1] as N^{γ_1}. Suppose we know an approximation p_0 of p such that $|p - p_0| < N^\beta$. If both d_p, d_q are known then one can factor N in deterministic poly$(\log N)$ time if $\frac{\alpha^2}{2} + \frac{\alpha(\delta_1+\delta_2)}{2} + \frac{\delta_1\delta_2}{2} + \alpha\beta + \frac{(\delta_1+\delta_2)\beta}{2} - \frac{3\beta^2}{2} - \alpha\gamma_1 - \delta_2\gamma_1 + 3\beta\gamma_1 - 2{\gamma_1}^2 - \frac{\alpha}{2} - \frac{\delta_1}{2} + \frac{3\beta}{2} - \gamma_1 - \frac{1}{2} < 0$.*

[1] As described in the proof of Theorem 3, the bit size of p can be correctly estimated in $\log N$ many attempts.

Proof. We have $ed_p = 1 + k(p-1)$ and $ed_q = 1 + k(q-1)$. So $k = \frac{ed_p - 1}{p-1}$. Let $k_0 = \frac{ed_p}{p_0}$. Then

$$|k - k_0| = |\frac{ed_p - 1}{p-1} - \frac{ed_p}{p_0}| \approx |\frac{ed_p}{p} - \frac{ed_p}{p_0}| = \frac{ed_p|p - p_0|}{pp_0} \leq N^{\alpha+\delta_1+\beta-2\gamma_1}.$$

Considering $q_0 = \frac{N}{p_0}$, it can be shown that $|q - q_0| < N^{1+\beta-2\gamma_1}$, neglecting the small constant. Assume, $q = N^{\gamma_2}$, where $\gamma_2 = 1 - \gamma_1$. So if we take $l_0 = \frac{ed_q}{q_0}$, then

$$|l - l_0| = |\frac{ed_q - 1}{q-1} - \frac{ed_q}{q_0}| \approx |\frac{ed_q}{q} - \frac{ed_q}{q_0}| = \frac{ed_q|q - q_0|}{qq_0} \leq N^{\alpha+\delta_2+1+\beta-2\gamma_1-2\gamma_2}.$$

Let $k_1 = k - k_0$ and $l_1 = l - l_0$. We have $ed_p + k - 1 = kp$. So $ed_p + k_0 + k_1 - 1 = (k_0 + k_1)p$. Similarly, $ed_q + l_0 + l_1 - 1 = (l_0 + l_1)q$. Now multiplying these equations, we get $(ed_p - 1 + k_0)(ed_q - 1 + l_0) + k_1(ed_q - 1 + l_0) + l_1(ed_p - 1 + k_0) + k_1 l_1 = (k_0 + k_1)p(l_0 + l_1)q$. Now if we substitute k_1, l_1 by x, y respectively, then we get $(ed_p - 1 + k_0)(ed_q - 1 + l_0) + x(ed_q - 1 + l_0) + y(ed_p - 1 + k_0) + xy = (k_0 + x)p(l_0 + y)q$. Hence we have to find the solution k_1, l_1 of

$$(ed_p - 1 + k_0)(ed_q - 1 + l_0) + x(ed_q - 1 + l_0) + y(ed_p - 1 + k_0) + xy = (k_0 + x)p(l_0 + y)q,$$

i.e., we have to find the roots of $f(x, y) = 0$, where $f(x, y) = (1 - N)xy + x(ed_q - 1 + l_0 - l_0 N) + y(ed_p - 1 + k_0 - k_0 N) + (ed_p - 1 + k_0)(ed_q - 1 + l_0) - k_0 l_0 N$.

Let $X = N^{\alpha+\delta_1+\beta-2\gamma_1}$ and $Y = N^{\alpha+\delta_2+1+\beta-2\gamma_1-2\gamma_2}$. Clearly X, Y are the upper bounds of (k_1, l_1), the root of f. Thus, $W = ||f(xX, yY)||_\infty \geq (ed_p - 1 + k_0)(ed_q - 1 + l_0) - k_0 l_0 N \approx e^2 d_p d_q$.

In the "Basic Strategy" of [12, Page 273], the set S is the set of all monomials of f^{m-1} for a given positive integer m. The set M is defined as the set of all monomials that appear in $x^i y^j f$, with $x^i y^j \in S$. This strategy will work well when k_1 and l_1 are of the same order, that is not significantly different in magnitude.

Otherwise, without loss of generality, let us assume that k_1 is significantly smaller than l_1. Following the "Extended Strategy" of [12, Page 274], we exploit extra t many shifts of x where t is a non-negative integer (in the "Basic Strategy", $t = 0$). Our aim is to find a polynomial f_0 that share the root (k_1, l_1) over the integers.

From [12], we know that these polynomials can be found by lattice reduction if $X^{s_1} Y^{s_2} < W^s$ for $s_j = \sum_{x^{i_1} y^{i_2} \in M \setminus S} i_j$ where $s = |S|$, $j = 1, 2$ and $W = ||f(xX, yY)||_\infty$. One can check that $s_1 = \frac{3}{2}m^2 + \frac{7}{2}m + \frac{t^2}{2} + \frac{5}{2}t + 2mt + 2$, $s_2 = \frac{3}{2}m^2 + \frac{7}{2}m + t + mt + 2$, and $s = (m+1)^2 + mt + t$.

Let $t = \tau m$. Neglecting the lower order terms we get that $X^{s_1} Y^{s_2} < W^s$ is satisfied when $(\frac{3}{2} + \frac{\tau^2}{2} + 2\tau)(\alpha + \delta_1 + \beta - 2\gamma_1) + (\frac{3}{2} + \tau)(\alpha + \delta_2 + 1 + \beta - 2\gamma_1 - 2\gamma_2) < (1 + \tau)(2\alpha + \delta_1 + \delta_2)$, i.e., when

$(\frac{\alpha}{2} + \frac{\delta_1}{2} + \frac{\beta}{2} - \gamma_1)\tau^2 + (\alpha + \delta_1 + 3\beta - 4\gamma_1 - 1)\tau + (\alpha + \frac{\delta_1 + \delta_2}{2} + 3\beta - 3\gamma_1 - \frac{3}{2}) < 0.$

In this case the value of τ for which the left hand side of the above inequality is minimum is $\tau = \frac{1 + 4\gamma_1 - 3\beta - \delta_1 - \alpha}{\alpha + \delta_1 + \beta - 2\gamma_1}$. Putting this value of τ we get the required condition as $\frac{\alpha^2}{2} + \frac{\alpha(\delta_1 + \delta_2)}{2} + \frac{\delta_1 \delta_2}{2} + \alpha\beta + \frac{(\delta_1 + \delta_2)\beta}{2} - \frac{3\beta^2}{2} - \alpha\gamma_1 - \delta_2\gamma_1 + 3\beta\gamma_1 - 2\gamma_1{}^2 - \frac{\alpha}{2} - \frac{\delta_1}{2} + \frac{3\beta}{2} - \gamma_1 - \frac{1}{2} < 0$.

The strategy presented in [12] works in polynomial time in $\log N$. As we follow the same strategy, N can be factored from the knowledge of N, e, d_p, d_q in deterministic polynomial time in $\log N$. $\square$

For practical purposes, p, q are same bit size and if we consider that no information about the bits of p is known, then we have $\gamma_1 = \gamma_2 = \beta = \frac{1}{2}$. In this case the required condition is $\frac{\alpha^2}{2} + \frac{1}{2}\alpha(\delta_1 + \delta_2) + \frac{\delta_1\delta_2}{2} - \frac{\alpha}{2} - \frac{\delta_1+\delta_2}{4} - \frac{3}{8} < 0$.

As the condition given in Theorem 4 is quite involved, we present a few numerical values in Table 2. What we like to identify here is to show that the bound of e can indeed exceed $\phi(N)$ (and also N) for which deterministic polynomial time equivalence of computing the CRT-RSA secret keys and factoring can be proved. This is also true when d_p, d_q exceeds the bound of $\max\{p-1, q-1\}$. Indeed, in some cases, the knowledge of a few most significant bits (MSBs) of one prime may be required.

α	δ_1	δ_2	β	γ_1
1.02	0.5	0.5	0.49	0.5
1.0	0.5	0.5	0.49	0.49
1.02	0.45	0.5	0.5	0.5
1.01	0.51	0.51	0.49	0.5
0.98	0.51	0.51	0.5	0.5
1.02	0.47	0.47	0.5	0.5

Table 2. Numerical values of $\alpha, \delta_1, \delta_2, \beta, \gamma_1$ following Theorem 4 for which N can be factored in poly($\log N$) time.

Now we present the experimental results corresponding to Theorem 4 in the set-up that has already mentioned earlier in this section. Once again, we like to point out that the experimental results cannot reach the theoretical bounds due to the small lattice dimensions. However, the values in Table 3 clearly demonstrates the cases where

- e exceeds N,
- d_p exceeds $p-1$.

N	p	q	e	d_p	d_q	LD	(m, t)	MSBs of p to be known	L^3-time
1000 bit	500 bit	500 bit	1001 bit	100 bit	500 bit	20	(2, 1)	5	63.40 sec
1000 bit	500 bit	500 bit	1001 bit	100 bit	502 bit	30	(3,1)	5	187.49 sec
1000 bit	500 bit	500 bit	1010 bit	100 bit	510 bit	20	(2, 1)	15	63.55 sec
1000 bit	500 bit	500 bit	1020 bit	100 bit	550 bit	35	(3, 2)	10	269.58 sec
1000 bit	500 bit	500 bit	1050 bit	100 bit	550 bit	35	(3, 2)	20	275.81 sec
1000 bit	500 bit	500 bit	1070 bit	100 bit	550 bit	35	(3, 2)	30	281.14 sec
1000 bit	400 bit	600 bit	1020 bit	100 bit	520 bit	35	(3, 2)	10	262.03 sec
1000 bit	500 bit	500 bit	1070 bit	100 bit	550 bit	48	(4, 2)	10	1227.20 sec
1000 bit	500 bit	500 bit	1001 bit	200 bit	502 bit	35	(3, 2)	20	266.52 sec
1000 bit	500 bit	500 bit	1020 bit	200 bit	520 bit	48	(4, 2)	10	1217.45 sec

Table 3. Experimental results corresponding to Theorem 4. LD is the lattice dimension and m, t are the parameters as explained in the proof of Theorem 4.

4 Conclusion

Towards theoretical interest, we present a deterministic poly($\log N$) time algorithm that can factorize N given e, d_p, d_q. This algorithm is based on lattice reduction techniques.

References

1. J. Blömer and A. May. A generalized Wiener attack on RSA. PKC 2004, LNCS 2947, pp. 1–13, 2004.
2. D. Boneh. Twenty Years of Attacks on the RSA Cryptosystem. Notices of the AMS, 46(2):203–213, February, 1999.
3. D. Boneh and G. Durfee. Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$. IEEE Trans. on Information Theory, 46(4):1339–1349, 2000.
4. D. Boneh, R. A. DeMillo and R. J. Lipton. On the importance of eliminating errors in cryptographic computations. Journal of Cryptology, 14(2):101–119, 2001.
5. D. Coppersmith. Small Solutions to Polynomial Equations and Low Exponent Vulnerabilities. Journal of Cryptology, 10(4):223–260, 1997.
6. J. -S. Coron. Finding Small Roots of Bivariate Integer Equations Revisited. Eurocrypt 2004, LNCS 3027, pp. 492–505, 2004.
7. J. -S. Coron. Finding Small Roots of Bivariate Integer Equations: a Direct Approach. Crypto 2007, LNCS 4622, pp. 379–394, 2007.
8. J. -S. Coron and A. May. Deterministic polynomial-time equivalence of computing the RSA secret key and factoring. Journal of Cryptology, 20(1):39–50, 2007.
9. N. Howgrave-Graham. Finding Small Roots of Univariate Modular Equations Revisited. Proceedings of Cryptography and Coding, LNCS 1355, pp. 131–142, 1997.
10. N. Howgrave-Graham. Approximate integer common divisors. Proceedings of CaLC'01, LNCS 2146, pp. 51–66, 2001.
11. E. Jochemsz. Cryptanalysis of RSA Variants Using Small Roots of Polynomials. Ph. D. thesis, Technische Universiteit Eindhoven, 2007.
12. E. Jochemsz and A. May. A Strategy for Finding Roots of Multivariate Polynomials with new Applications in Attacking RSA Variants. Asiacrypt 2006, LNCS 4284, pp. 267–282, 2006.
13. E. Jochemsz and A. May. A Polynomial Time Attack on RSA with Private CRT-Exponents Smaller Than $N^{0.073}$. Crypto 2007, LNCS 4622, pp. 395–411, 2007.
14. A. K. Lenstra, H. W. Lenstra and L. Lovász. Factoring Polynomials with Rational Coefficients. Mathematische Annalen, 261:513–534, 1982.
15. S. Maitra, S. Sarkar. A New Class of Weak Encryption Exponents in RSA. Indocrypt 2008, LNCS 5365, pp. 337–349, 2008.
16. A. May. Computing the RSA secret key is deterministic polynomial time equivalent to factoring. Crypto 2004, LNCS 3152, pp. 213–219, 2004.
17. A. May. Using LLL-Reduction for Solving RSA and Factorization Problems: A Survey. LLL+25 Conference in honour of the 25th birthday of the LLL algorithm, 2007. Available at http://www.informatik.tu-darmstadt.de/KP/alex.html [last accessed 23 December, 2008].
18. G. L. Miller. Riemann's hypothesis and test of primality. 7th Annual ACM Symposium on the Theory of Computing, pp. 234–239, 1975.
19. R. L. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of ACM, 21(2):158–164, Feb. 1978.
20. D. R. Stinson. Cryptography - Theory and Practice. 2nd Edition, Chapman & Hall/CRC, 2002.
21. M. Wiener. Cryptanalysis of Short RSA Secret Exponents. IEEE Transactions on Information Theory, 36(3):553–558, 1990.