A New Lattice for Implicit Factoring

Yanbin Pan and Yingpu Deng
Key Laboratory of Mathematics Mechanization
Academy of Mathematics and Systems Science,Chinese Academy of Sciences
Beijing 100190, China
{panyanbin,dengyp}@amss.ac.cn

2009.3.20

Abstract

In PKC 2009, Alexander May and Maike Ritzenhofen[7] proposed an ingenious lattice-based method to factor the RSA moduli $N_1 = p_1q_1$ with the help of the oracle which outputs $N_2 = p_2q_2$, where p_1 and p_2 share the t least significant bits and t is large enough when we query it with N_1. They also showed that when asking $k-1$ queries for RSA moduli with α-bit q_i, they can improve the bound on t to $t \geq \frac{k}{k-1}\alpha$. In this paper, we propose a new lattice for implicit factoring in polynomial time, and t can be a little smaller than in [7]. Moreover, we also give a method in which the bound on t can also be improved to $t \geq \frac{k}{k-1}(\alpha - 1 + \frac{1}{2}\log\frac{k+3}{k})$ but with just only one query. Moreover we can show that our method reduces the running time of the implicit factoring for balanced RSA moduli much efficiently and makes it practical.

Keywords: Implicit Factoring, Lattice, RSA moduli

1 Introduction

Although after the invention of the public key cryptosystem RSA[12], factoring large integers becomes more and more important and many efforts including the Quadratic Sieve [10], the Elliptic Curve Method[4] and the Number Field Sieve[5] are made to improve the factorization complexity, there still have not been a polynomial factorization algorithm on classical computing model while Shor[13] has showed there exists a polynomial time algorithm for factorization on quantum computing platforms.

In classical domain, Rivest and Shamir[11] showed that $N = pq$ can be factored efficiently with an oracle that outputs $\frac{3}{5}\log p$ most significant bits of p at Eurocrypt 1985. Later at Eurocrypt 1996, Coppersmith[2] improved the bound to $\frac{1}{2}\log p$.

In PKC 2009, Alexander May and Maike Ritzenhofen[7] proposed an ingenious lattice-based method to factor the RSA moduli with the help of an oracle. Considering the RSA moduli $N_1 = p_1q_1$, they can factor N_1 efficiently if the oracle outputs $N_2 = p_2q_2$, where p_1, p_2 share t least significant bits and t is large enough. More precisely, let q_1, q_2 be α-bit prime numbers, then if $t > 2(\alpha + 1)$, they can find q_1, q_2 in polynomial time. To further improve upon the bound on t, they make $k-1$ queries, and prove that if $t \geq \frac{k}{k-1}\alpha$, they can factor N_1 efficiently under some assumption. In their paper, they also proposed an algorithm to factor balanced RSA moduli with $k-1$ queries and $2^{\frac{n}{4}}$ guesses, where N_1 is an n-bit number. However, $2^{\frac{n}{4}}$ guesses is too large to make the algorithm practical and by Coppersmith's method[3], we can also directly factor N_1 with $2^{\frac{n}{4}}$ guesses without the oracle's help.

In this paper, we propose another lattice-based algorithm which at least has the advantages below:

- We also reach the bound $\frac{k}{k-1}\alpha$ but just using only one query, this can be realized more easily in the real life.
- The vector we will find in our lattice is shorter than the one in [7]. By the Gaussian Heuristic and experiences, our vector can be found more easily.
- For the balanced RSA moduli, our algorithm uses just only one query and $2^{\frac{n}{2k}}$ guesses, which make our algorithm practical. This shows that the oracle does help us to factor N_1.

The paper is organized as follows. In Section 2, we give some results we need. In Section 3, we give our new $2-$ *dimensional* lattice with one query. In Section 4, we present a $k-$ *dimensional* version to improve the bound on t. At last, we give the method to factor the balanced RSA moduli with the help of the oracle in Section 5.

2 Preliminaries

An integer lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{Z}^n$ and can be also defined as below:
Definition of Lattice: Let $b_1, b_2 \cdots, b_d \in \mathbb{Z}^n$ be linearly independent vectors, the lattice $\mathcal{L}$ spanned by them is

$$\mathcal{L} = \{\sum_{i=1}^{d} a_i b_i | a_i \in \mathbb{Z}\}$$

$B = \begin{pmatrix} b_1 \\ \vdots \\ b_d \end{pmatrix}$ is called the basis of $\mathcal{L}$.

A lattice is full rank if $d = n$. If $\mathcal{L}$ is full rank, the determinant $det(\mathcal{L})$ is equal to the absolute value of determinant of the basis B.

Denote $\|v\|$ the Euclidean l_2-norm of a vector v and $\lambda_1(\mathcal{L})$ the length of the shortest vector in the lattice $\mathcal{L}$. The Minkowski's Theorem[9] tells us $\lambda_1(\mathcal{L}) \leq \sqrt{n} det(\mathcal{L})^{\frac{1}{n}}$ where $\mathcal{L}$ is an $n - dimensional$ lattice. Moreover, by the Gaussian Heuristic, $\lambda_1(\mathcal{L}) \approx \sqrt{\frac{n}{2\pi e}} det(\mathcal{L})^{\frac{1}{n}}$ for an $n - dimensional$ random lattice $\mathcal{L}$, and experiences tell us the smaller $\frac{\|v\|}{\sqrt{\frac{n}{2\pi e}} det(\mathcal{L})^{\frac{1}{n}}}$, the more easily we can find v in practice.

We can use Gaussian reduction algorithm to find the shortest vector v in polynomial time and the information on Gaussian reduction algorithm can be found in [8]. To find a short vector in a $n - dimensional$ lattice $\mathcal{L}$, we can use LLL algorithm[6] to find $v \in \mathcal{L}$, such that $\|v\| \leq 2^{\frac{n-1}{4}} det(\mathcal{L})^{\frac{1}{n}}$ in polynomial time.

LLL algorithm is used by Coppersmith[3] to find the factorization of $N = pq$ in polynomial time if we know the low order $\frac{n}{4}$ bits of p where N is an n-bit number.

3 The New Lattice for Implicit Factoring

To factor $N_1 = p_1 q_1$, we get N_2 after one query with the oracle, where $N_2 = p_2 q_2$, p_1, p_2 share the t least significant bits and q_1, q_2 are α-bit prime as assumed in [7].

As in [7], let $p_1 = p + 2^t \bar{p}_1$ and $p_2 = p + 2^t \bar{p}_2$, so

$$\begin{aligned} N_1 &= pq_1 + 2^t \bar{p}_1 q_1 \\ N_2 &= pq_2 + 2^t \bar{p}_2 q_2 \end{aligned}$$

Reducing the two equations modulo 2^t, we get

$$\begin{aligned} N_1 &= pq_1 \bmod 2^t \\ N_2 &= pq_2 \bmod 2^t \end{aligned}$$

Eliminating p from the two equations, May and Ritzenhofen[7] get the linear equation

$$(N_1^{-1} N_2) q_1 - q_2 = 0 \bmod 2^t$$

and proves that the set of the solutions of the equation above is a lattice $\mathcal{L}_{MR}$ spanned by the row vectors of the basis matrix

$$B_L = \begin{pmatrix} 1 & N_1^{-1} N_2 \\ 0 & 2^t \end{pmatrix}$$

So the vector $q_{MR} = (q_1, q_2) \in \mathcal{L}_{MR}$, if $\|q_{MR}\|$ is small enough, they can find q_1 by Gaussian reduction algorithm.

Here we use a new lattice as below instead of $\mathcal{L}_{MR}$. Assume $q_1 < q_2$(Notice that this can't hold forever, we give a short comment in the remark on the case $q_1 > q_2$ and we can compute $gcd(N_1, N_2)$ if $q_1 = q_2$), we subtract N_1 from N_2

$$\begin{aligned} N_1 &= pq_1 \bmod 2^t \\ N_2 - N_1 &= p(q_2 - q_1) \bmod 2^t \end{aligned}$$

Since N_1 is an odd number, we can eliminate p and get

$$N_1^{-1}(N_2 - N_1)q_1 - (q_2 - q_1) = 0 \bmod 2^t$$

As in [7], it can be also easily checked that the set of solutions

$$L = \{(x_1, x_2) \in \mathbb{Z}^2 | N_1^{-1}(N_2 - N_1)x_1 - x_2 = 0 \bmod 2^t\}$$

forms a lattice $\mathcal{L}_{new}$ spanned by the rows of the basis matrix

$$B_{new} = \begin{pmatrix} 1 & N_1^{-1}(N_2 - N_1) \\ 0 & 2^t \end{pmatrix}$$

Notice that the vector $q_{new} = (q_1, q_2 - q_1) \in \mathcal{L}_{new}$ and $q_1 \le 2^\alpha$, $q_2 - q_1 \le 2^{\alpha-1}$, so we can get a similar theorem as Theorem 5 in [7].

Theorem 1. *Let $N_1 = p_1q_1, N_2 = p_2q_2$ be two different RSA moduli with α-bit q_i, and $q_1 < q_2$. Suppose that p_1, p_2 share at least $t > 2\alpha + 1$ bits. Then N_1, N_2 can be factored in quadratic time by running Gaussian reduction on $\mathcal{L}_{new}$.*

Using the lattice $\mathcal{L}_{new}$, we can reduce the numbers of the sharing bits by 1.

Remark 1. *If $q_1 > q_2$, we can construct a new lattice $\mathcal{L}'_{new}$ spanned by the rows of matrix*

$$B'_{new} = \begin{pmatrix} 1 & N_2^{-1}(N_1 - N_2) \\ 0 & 2^t \end{pmatrix}$$

Notice that $q'_{new} = (q_2, q_1 - q_2) \in \mathcal{L}'_{new}$ and $\|q'_{new}\| < \|q_{MR}\|$, so we can also find q'_{new} and so q_2 by Theorem 1, then $q_1 = N_2^{-1}N_1 \bmod 2^t$.

Another question is we don't know whether $q_1 < q_2$ or not, so we run Gaussian reduction on the two lattice $\mathcal{L}_{new}$ and $\mathcal{L}'_{new}$, if any of them successes, then we can factor N_1.

Notice that for any unimodular matrix $U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, either $aN_1 + bN_2$ or $cN_1 + dN_2$ is odd. Suppose not, they are both even, then either a, b are even or are *odd, so are c, d. Moreover, so are ad, bc, hence $ad - bc$ is even, which contradicts the fact $ad - bc = 1$. We assume $aN_1 + bN_2$ is odd, then we can construct a lattice spanned by the row vectors of the matrix $\begin{pmatrix} 1 & (aN_1+bN_2)^{-1}cN_1+dN_2 \\ 0 & 2^t \end{pmatrix}$ which contains the vector $q = (aq_1 + bq_2, cq_1 + dq_1)$, if q is short enough, then we may improve the bound on t.*

4 Improving the Bound on t

To further improve upon the bound on t, we needn't query the oracle more than once, we can do it just with only one query.

Assume $q_1 < q_2$, Let the $k-dimensional$ lattice $\mathcal{L}_{new}^k$ be the one spanned by the rows of the matrix below:

$$B_{new}^k = \begin{pmatrix} 1 & N_1^{-1}(N_2-N_1) & N_1^{-1}(N_2-N_1) & \cdots & N_1^{-1}(N_2-N_1) \\ 0 & 2^t & 0 & \cdots & 0 \\ 0 & 0 & 2^t & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2^t \end{pmatrix}_{k\times k}$$

Notice that the vector $q_{new}^k = (q_1, q_2 - q_1, q_2 - q_1, \cdots, q_2 - q_1) \in \mathcal{L}$ and $\|q_{new}^k\| \leq \sqrt{k+3}2^{\alpha-1}$, if q_{new}^k is indeed the shortest vector in $\mathcal{L}$, then by Minkowski's Theorem, we have

$$\|q_{new}^k\| \leq \sqrt{k}2^{\frac{k-1}{k}t}$$

Let

$$\sqrt{k+3}2^{\alpha-1} \leq \sqrt{k}2^{\frac{k-1}{k}t}$$

which implies that

$$t \geq \frac{k}{k-1}(\alpha - 1 + \frac{1}{2}\log\frac{k+3}{k})$$

This also give us a good suggestion for the choice of k, if t is large enough, we can choose small k to reduce the running time and improve the success probability, if t is small, we have to choose large k.

Similar to Assumption 6 in [7], we can also assume that if $t \geq \dfrac{k}{k-1}(\alpha - 1 + \frac{1}{2}\log\dfrac{k+3}{k})$, then q'_{new} may be a shortest vector in $\mathcal{L}$.

Comparing with the lattice $\mathcal{L}_{MR}^k$ in [7] spanned by the rows of the matrix

$$B_{MR}^k = \begin{pmatrix} 1 & N_1^{-1}N_2 & N_1^{-1}N_3 & \cdots & N_1^{-1}N_k \\ 0 & 2^t & 0 & \cdots & 0 \\ 0 & 0 & 2^t & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2^t \end{pmatrix}_{k\times k}$$

and the vector $q_{MR}^k = (q_1, q_2, \cdots, q_k)$, we find that we just use only one query, but the bound on t can be improved to a lower bound. Notice that by the Gaussian Heuristic, we have

$$\lambda_1(\mathcal{L}_{MR}^k) \approx \lambda_1(\mathcal{L}_{new}^k) \approx \sqrt{\frac{k}{2\pi e}} 2^{\frac{k-1}{k}t}$$

However, $\|q_{new}^k\| < \|q_{MR}^k\|$, since $q_2 - q_1 < 2^{\alpha-1} \le q_i$, then $\dfrac{\|q_{new}^k\|}{\sqrt{\frac{k}{2\pi e}} 2^{\frac{k-1}{k}t}} < \dfrac{\|q_{MR}^k\|}{\sqrt{\frac{k}{2\pi e}} 2^{\frac{k-1}{k}t}}$, so the experience tells us q_{new}^k can be found more efficiently than q_{MR}^k in practice.

Next we present a case in which Assumption 6 in [7] is not true, but our assumption may have a little more advantage. To ensure Assumption 6, we show that k can't be too large.

Claim: Let $t = \alpha + t_0$ where $t_0 \ge 0$, and choose $k_0 \in \mathbb{R}$ such that $\frac{1}{2}\log k_0 - \frac{k_0}{k_0-1}\alpha + \alpha - 1 = t_0$, then for $k \ge k_0$, Assumption 6 is not true.

Proof. Let $f(x) = \frac{1}{2}\log x - \frac{x}{x-1}\alpha + \alpha - 1$, notice that $f(x)$ is a strictly monotonic increasing function, and $f(2) < 0$, $\lim_{x\to+\infty} f(x) = +\infty$, so for any $t_0 \ge 0$, we can find k_0 such $f(k_0) = t_0$.

for any $k \ge k_0$, let $t = \frac{k}{k-1}\alpha + t'$ where $0 \le t' < t_0$, so $f(k) \ge f(k_0) > t'$, i.e. $\frac{1}{2}\log k - \frac{k}{k-1}\alpha + \alpha - 1 > t'$, this implies that

$$t = \frac{k}{k-1}\alpha + t' < \frac{1}{2}\log(k) + \alpha - 1$$

which can also be written as

$$2^t < \sqrt{k}2^{\alpha-1} \le \|q_{MR}^k\|$$

So the vector $(0, 2^t, 0, \cdots, 0)$ is shorter than q_{MR}^k. □

Notice that the proof uses the fact that $\|q_{MR}^k\| \ge \sqrt{k}2^{\alpha-1}$. Since q_{new}^k is shorter than q_{MR}^k, hence has much lower bound, so our assumption may be true for larger k.

Remark 2. *To increase the success rate and reduce the running time, we can use a simple strategy. In the process of LLL algorithm, once we get a new basis vector* $q = (x_1, x_2, \cdots, x_k)$*, we compute* $gcd(x_1, N_1)$ *and halt when we get* q_1*. If* $q_1 > q_2$*, then we can use the similar method in Remark 1.*

5 Implicit Factoring of Balanced RSA Moduli

The most attractive advantage of our method is decreasing the running time for balanced RSA moduli much efficiently and making the implicit factoring of balanced RSA moduli practical.

Let $N_i = p_i q_i$ such that p_i and q_i have bitsize $\frac{n}{2}$ each for $i = 1, 2$. As in [7], we assume $t = \frac{n}{4}$ to minimize the time complexity. We first choose $k \in \mathbb{Z}$, such that it is possible to solve the CVP on a $k-$*dimensional* lattice in a reasonable time. Let $\beta = \lfloor \frac{k-1}{k} t \rfloor$. Then we can split $q_i \pmod{2^t}$ into $2^\beta \bar{q}_i + x_i \pmod{2^t}$ for $i = 1, 2$. So the number of the bits of x_i is at most β. Suppose $x_1 < x_2$, and in Section 3 we have

$$N_1^{-1}(N_2 - N_1)q_1 - (q_2 - q_1) = 0 \bmod 2^t$$

So we have

$$N_1^{-1}(N_2 - N_1)x_1 - (x_2 - x_1) = 2^\beta(\bar{q}_2 - \bar{q}_1 - N_1^{-1}(N_2 - N_1)\bar{q}_1) \bmod 2^t$$

Let $c = 2^\beta(\bar{q}_2 - \bar{q}_1 - N_1^{-1}(N_2 - N_1)\bar{q}_1)$, and we guess $\bar{q}_1, \bar{q}_2$, since the size of $\bar{q}_i$ is at most $t - \beta$, so we must guess roughly $\frac{n}{2k}$ bits, which is much less than $\frac{n}{4}$ in [7]. For any pair $(\bar{q}_1, \bar{q}_2 - \bar{q}_1)$ we guess, we compute the corresponding c, and define the lattice $\mathcal{L}_{bal}$ spanned by the rows of the following basis matrix

$$B_{bal} = \begin{pmatrix} 1 & N_1^{-1}(N_2 - N_1) & N_1^{-1}(N_2 - N_1) & \cdots & N_1^{-1}(N_2 - N_1) \\ 0 & 2^t & 0 & \cdots & 0 \\ 0 & 0 & 2^t & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2^t \end{pmatrix}_{k \times k}$$

Denote c_{bal} the target vector $(0, c, c, \cdots, c)$ and notice that the vector $q_{bal} = (x_1, x_2 - x_1 + c, x_2 - x_1 + c, \cdots, x_2 - x_1 + c) \in \mathcal{L}_{bal}$, , then

$$\|q_{bal} - c_{bal}\| = \|(x_1, x_2 - x_1, x_2 - x_1, \cdots, x_2 - x_1)\| \leq \sqrt{k}2^\beta \leq \sqrt{k}2^{\frac{(k-1)n}{4k}}$$

So we also make the assumption, that q_{bal} is the closest vector to c_{bal} in $\mathcal{L}_{bal}$. Under the assumption, since k is fixed, then we can use a polynomial time algorithm to find q_{bal} hence get x_1(see Blömer[1]). So we find the least $\frac{n}{4}$ bits of q_1 (i.e. q_1 mod

2^t) by combining $\bar{q}_1$ and x_1. Then by using Coppersmith's method[3], we can find the complete q_1.

Denote $TIME_{cvp}$ the running time of getting q_{bal} and $TIME_{cop}$ the running time of Coppersmith's method , then in the worst case, we can use our method to get q_1 in $2^{\frac{n}{2k}}(TIME_{cvp} + TIME_{cop})$ instead of $2^{\frac{n}{4}}(TIME_{cvp} + TIME_{cop})$ which is needed when using the original method in [7].

Since $TIME_{cvp}+TIME_{cop}$ is bounded by a polynomial in $(k!, max(\log N_1, \log N_2))$, the number of the guesses must be the most important factor to decide whether a method is practical or not. For example, if $n = 1000$ and $k = 50$, then we only try 1024 guesses which is practical, but not 2^{250} which is impossible to realize nowadays.

Remark 3. *Obviously, a parallel strategy can give us great help to find* q_1 *.*

References

[1] J. Blömer, Closest Vectors, Successive Minima, and Dual HKZ-bases of Lattices, ICALP 2000, Lecture Notes in Computer Science, Volume 1853, Springer-Verlag, pp.248-259, 2000

[2] D.Coppersmith: Finding a Small Root of a Bivariate Integer Equation, Factoring with High Bits Known, Advances in Cryptology (Eurocrypt 96), Lecture Notes in Computer Science, Volume 1070, pp. 178-189, Springer-Verlag, 1996

[3] D. Coppersmith: Small solutions to polynomial equations and low exponent vulnerabilities, Journal of Cryptology, Volume 10(4), pp. 223-260, 1997

[4] H. W. Jr. Lenstra: Factoring Integers with Elliptic Curves, Ann. Math. 126, pp. 649-673, 1987

[5] A. K. Lenstra, H. W. Jr. Lenstra: The Development of the Number Field Sieve, Springer-Verlag, 1993

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

[7] A. May and M. Ritzenhofen: Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint. *In Proc. of PKC 2009*, volume 5443 of *LNCS*, pages 1-14. Springer Berlin / Heidelberg

[8] C. D. Meyer: Matrix Analysis and Applied Linear Algebra, Cambridge University Press, 2000

[9] H. Minkowski: Geometrie der Zahlen, Teubner-Verlag, 1896

[10] C.Pomerance: The Quadratic Sieve Factoring Algorithm. In Advances in Cryptology (Eurocrypt 84), Lecture Notes in Computer Science, Volume 209, pp. 169-182, Springer-Verlag, 1985.

[11] R. Rivest, A.Shamir: Efficient Factoring Based on Partial Information, In Advances in Cryptology (Eurocrypt 85), Lecture Notes in Computer Science, Volume 219 , pp. 31-34, Springer-Verlag, 1986

[12] R.Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, Volume 21(2), pp. 120-126, 1978

[13] P. Shor: Algorithms for Quantum Computation: Discrete Logarithms and Factoring, Proceedings 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, IEEE Computer Science Press pp. 124-134, 1994