Integer Factoring with Extra Information

Domingo Gómez, Jaime Gutierrez, Álvar Ibeas

Faculty of Sciences,
University of Cantabria,
Santander E–39071, Spain

Abstract. In this paper we present an heuristic algorithm and its implementation in C++ program for integer factoring with high-order bits known based on lattice reduction techniques. Our approach is inspired in algorithms for predicting pseudorandom numbers.

1 Introduction

Many very well known and important cryptographic protocols are based on the assumption that factoring large composite integers is computationally difficult. The most famous one is RSA cryptosystem, which is currently used in a wide variety of products, platforms and industries around the world. RSA is incorporated into all of the major protocols for secure Internet communications, including S/MIME and S/WAN.

In this paper, we consider a number N which is product of two primes: P and Q. We analyze the assumption that factoring is computationally difficult when the cryptanalyst has access to extra information.

In cryptographic applications, the cryptanalyst may have available additional information above and beyond the number N itself, see [17]. In practice, Alice or Bob (one of them) typically knows P and Q already, and uses these factors implicitly and/or explicitly during her/his cryptographic computations. The results of these computations may become known to the cryptanalyst, who thereby may find himself at an advantage compared to a pure factoring situation. The necessary information and timing measurements may be obtained by passively eavesdropping on an interactive protocol. The Chinese Remainder Theorem (CRT) is also often used to optimize RSA private key operations. With CRT, $y \bmod P$ and $y \bmod Q$ are computed first (being y is the message to send). These initial modular reduction steps can be vulnerable to timing attacks. The simplest such attack is to choose values of y that are close to P or to Q, and then use timing measurements to determine whether the guessed value is larger or smaller that the actual value of P and Q.

So in practice, extra information may become available to the cryptanalyst, for one of the following reasons:

- loss of the equipment that generated P and Q,
- explicit release of partial extra information as part of a protocol, for instance exchange of secret,

– timing measurements,
– routine usage of P and Q to decrypt mail, sign messages, etc.,
– poor physical security to guard P and Q,
– any other heuristic attack ...

Suppose that an attacker is able to find the high-order h bits of the smallest prime P : *can we recover P and Q in polynomial time in* $\log N$*?*

A directed application of this problem comes from paper [23]. Here, an identity-based variant of RSA in which the user's modulus N is related to his identity is proposed. For example, the high-order bits of N may be the user's name encoded in ASCII. If N is generated in such a way, somewhat more than the high order $\frac{1}{4}\log_2 N$ bits of P are revealed to the public.

From now, given an integer number A, $\log A$ means $\log_2 A$ and polynomial time means polynomial in $\log N$.

Regarding positive answers for this problem, we can mention the work by Rivest and Shamir [22] using a special case of integer programming, which needs about $h = \frac{1}{3}\log N$ bits of P. The paper [10] by Coppersmith used a lattice-based method to factor N using $h = \frac{3}{10}\log N$ bits of P.

The king result is also due to Coppersmith [7–9] which requires only $h = \frac{1}{4}\log N$ bits of P and also uses lattice reduction techniques. We do not know any efficient implementation of the algorithm, (see [16] for special polynomials), which is quite involved.

In practice, if the number of bits from P is near the threshold $\frac{1}{4}\log N$, the size of intermediate steps grows. This is, less bits known imply more computing time. The dimension of the lattice is big, as well as the size of lattice coefficients. Finally, his method requires the resultant of two bivariate integer polynomials. Others variants of Coppersmith's approach are in [16, 11, 13].

In this paper we introduce some ideas to solve this important problem. Our heuristic algorithm is based on the same approach used for predicting non-linear pseudorandom numbers, see [3, 5]. In contrast to algorithm in [7–9]:

– it is asymptotically less efficient,
– it does not require to compute any resultant of integer bivariate polynomials, but it requires to exclude a very small set of primes Q,
– it requires lower size lattice coefficients and smaller lattice dimension,
– it is easy to implement.

The remainder of the paper is structured as follows. We start with a short outline of some basic facts about lattices in Section 2. In Section 3 we introduce Coppersmith's method (Subsection 3.2) and then we give the first approach in Subsection 3.3. In Section 4 we comment details of the C++ implementation using the **NTL** (Number Theory Library) [21] and discuss the results of numerical tests. Finally, the last section states a couple of short conclusions.

2 Background on Lattices

Here we collect several well-known facts about lattices which form the background to our algorithms.

We review several related results and definitions on lattices which can be found in [4, 12]. For more details and more recent references, we also recommend consulting [1, 14, 19, 20].

Let $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$ be a set of linearly independent vectors in $\mathbb{R}^r$. The set

$$\mathcal{L} = \{c_1\boldsymbol{b}_1 + \ldots + c_s\boldsymbol{b}_s, \quad c_1, \ldots, c_s \in \mathbb{Z}\}$$

is called an *s-dimensional lattice* with *basis* $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$. If $s = r$, the lattice L is of *full rank*.

To each lattice $\mathcal{L}$ one can naturally associate its *volume*

$$\mathrm{vol}(\mathcal{L}) = \left(\det\left(\langle \boldsymbol{b}_i, \boldsymbol{b}_j\rangle\right)_{i,j=1}^{s}\right)^{1/2},$$

where $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ denotes the inner product. This definition does not depend on the choice of the basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$.

For a vector $\boldsymbol{u}$, let $\|\boldsymbol{u}\|$ denote its *Euclidean norm*. The famous Minkowski theorem gives the upper bound:

$$\min\{\|\boldsymbol{z}\|: \ \boldsymbol{z} \in \mathcal{L} \setminus \{\boldsymbol{0}\}\} \leq s^{1/2}\mathrm{vol}(\mathcal{L})^{1/s} \tag{1}$$

on the shortest nonzero vector in any s-dimensional lattice $\mathcal{L}$ in terms of its volume.

The Minkowski bound (1) motivates a natural question, named *Shortest Vector Problem (SVP)*: how to find the shortest nonzero vector in a lattice. Unfortunately, there are several indications that this problem is **NP**-hard when the dimension grows. This study has suggested several definitions of a *reduced* basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$ for a lattice, trying to obtain a shortest vector by the first basis element $\boldsymbol{b}_1$. The celebrated *LLL algorithm* of Lenstra, Lenstra and Lovász [18] provides a concept of *reduced* basis and a desirable solution in practice.

Another related problem is the *Closest Vector Problem, CVP*: Given a lattice $\mathcal{L} \subseteq \mathbb{R}^r$ and a shift vector $\boldsymbol{t} \in \mathbb{R}^r$, the goal consists on finding a vector in the set $\boldsymbol{t} + \mathcal{L}$ with minimum norm. It is well-known that the CVP is **NP**-hard when the dimension grows. An approximate polynomial time solution is presented in [2].

However, both computational problems SVP and CVP are known to be solvable in deterministic polynomial time (polynomial in the bit-size of a basis of $\mathcal{L}$) provided that the dimension of $\mathcal{L}$ is fixed (see [15], for example). The lattices in this paper are of fixed (and low) dimension.

The lattices employed in this paper consist of integer solutions $\boldsymbol{x} = (x_0, \ldots, x_{s-1}) \in \mathbb{Z}^s$ of a system of congruences

$$\sum_{i=0}^{s-1} a_{ij}x_i \equiv 0 \bmod q_j, \qquad j = 1, \ldots, m,$$

modulo some integers $q_1, \ldots, q_m$. Typically (although not always) the volume of such a lattice is the product $Q = q_1 \ldots q_m$. Moreover all the aforementioned algorithms, when applied to such a lattice, become polynomial in $\log Q$.

If $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$ is a basis of the above lattice, by the well-known Hadamard inequality we have:

$$\prod_{i=1}^{s} \|\boldsymbol{b}_i\| \geq \mathrm{vol}(\mathcal{L}). \tag{2}$$

3 Integer Factoring with high-order bits known

In this section we present an sketch of an algorithm for integer factoring with high-order bits known.

3.1 Notation

Given a number N which is product of two primes P and Q, we suppose that $P < Q$. Our results involve a parameter $\varDelta$ which measures how many high-order bits of P are known. This parameter is assumed to vary independently of P subject to satisfy the inequality $\varDelta < P$.

More precisely, we say that an integer w is a $\varDelta$-*approximation* to the integer u when $|w - u| \leq \varDelta$.

So, in the case where the high-order h bits of the prime P are given, we can build a $\varDelta$-approximation P_0 to P, by taking the h high-order bits of P and $\lfloor \log P \rfloor + 1 - h$ zeroes. In this case, $\varDelta = 2^{\lfloor \log P \rfloor + 1 - h} - 1$, that is,

$$P - P_0 \leq \varDelta \cong \frac{P}{2^h}.$$

By dividing N into P_0, we obtain a $\varDelta_1$-approximation Q_0 to Q:

$$|Q - Q_0| \leq \varDelta_1 \cong \frac{Q\varDelta}{P}.$$

Let $\varepsilon_0 = P - P_0$ and $\varepsilon_1 = Q - Q_0$. From $N = PQ$ we obtain:

$$f(\varepsilon_0, \varepsilon_1) = 0,$$

where

$$f(\varepsilon_0, \varepsilon_1) = (P_0 + \varepsilon_0)(Q_0 + \varepsilon_1) - N.$$

And with

$$|\varepsilon_0| \leq \varDelta, \quad |\varepsilon_1| \leq \varDelta_1. \tag{3}$$

The main objective is to find roots of this innocent polynomial $f(\varepsilon_0, \varepsilon_1)$.

A big and important result for arbitrary bivariate and irreducible integer polynomial is due to Coppersmith.

3.2 Coppersmith method

We start stating one of the main result in [8]:

Theorem 1 (Theorem 2-[8]). *Let $p(\varepsilon_0, \varepsilon_1)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let Δ, Δ_1 be bounds on the desired solutions x_0, y_0. Define $p^*(\varepsilon_0, \varepsilon_1) = p(\varepsilon_0 \Delta, \varepsilon_1 \Delta_1)$ and let W be the absolute value of the largest coefficient of $p^*(\varepsilon_0, \varepsilon_1)$. If*

$$\Delta \Delta_1 \leq W^{2/(3\delta)-\epsilon} 2^{-14\delta/3},$$

then in polynomial time in $(\log W, \delta, 1/\epsilon)$ we can find all integer pairs (x_0, y_0) with $p(x_0, y_0) = 0$ bounded by $|x_0| \leq \Delta$, $|y_0| \leq \Delta_1$.

In order to apply the above result to the problem of factoring an integer when we know the high-order bits of one of the factors and according Coppersmith [8, 9], we suppose that we know $N = PQ$ and the high-order $h = \frac{1}{4}\log_2 N$ bits of P. We apply Theorem 2 to polynomial $f(\varepsilon_0, \varepsilon_1)$ given by Equation (3.1) and take:

$$\begin{aligned} |\varepsilon_0| &< P_0 N^{-1/4} = \Delta, \\ |\varepsilon_1| &< Q_0 N^{-1/4} = \Delta_1, \\ \delta &= 1, \quad W = N^{3/4}. \end{aligned}$$

As corollary it follows:

Theorem 2 (Theorem 4-[8]). *In polynomial time we can find the factorization of $N = PQ$ if we know the high-order $(\frac{1}{4}\log_2 N)$ bits of P*

The proof of this result is quite involved and uses lattice reduction techniques. In one hand, the proof requires the calculation of the associated lattice volume. The corresponding matrix has $(k+\delta)^2$ rows and $(k+\delta)^2 + k^2$ columns, where k is an integer satisfying

$$k > \frac{2}{3\epsilon}.$$

Fixed the degree δ, a bound for the lattice dimension is $O(k^2)$.

Remark 1. Given $h = (\frac{1}{3}\log_2 N)$ bits of P we want to find the appropriate integer k.

We know that $\Delta\Delta_1 < W^{2/(3\delta)-\epsilon} 2^{-14\delta/3}$ and since $W^{3/4}$, we obtain

$$(\frac{1}{4} + \frac{3}{8}\epsilon)\log_2 N < h = \frac{1}{3}\log_2 N.$$

Then $\epsilon < \frac{2}{9}$. On the other hand, $k > \frac{2}{3\epsilon}$ implies $k > 3$. So the lattice dimension is bigger than 41.

In next subsection we present a new approach where for $\frac{1}{3}\log_2 N$ bits of P we need a lattice of dimension only 4.

3.3 One round lattice reduction

First of all, we remark that if P_0 and Q_0 are given as in Subsection 3.1, they are unique, that is, if we know the high-order h bits of P and N is fixed, then P_0 and Q_0 are uniquely determined. Now we present the main result in this subsection:

Theorem 3. *For a prime P and natural numbers g and h, there is a set $\mathcal{V}(P,g,h) \subseteq \mathbb{Z}_P$ of cardinality $\#(\mathcal{V}(P,g,h)) = O(2^{2\log P+g-3h})$ with the following property. Given $N = PQ$ and the high-order h bits of P, whenever $Q \notin \mathcal{V}(P,g,h)$, there exists an algorithm recovering P and Q on deterministic polynomial time, where $g = \lfloor \log Q \rfloor$.*

Proof. First, we will suppose the number of bits of the prime P is given, then we know g. Let $P_0, Q_0, \Delta_1, \Delta$ as in Subsection 3.1, where P_0 is a Δ-approximation to P and Q_0 is a Δ_1-approximation to Q. We can reformulate the theorem as follows: there is a set $\mathcal{V}(\Delta,\Delta_1) \subset \mathbb{Z}_P$ of cardinality $\#(\mathcal{V}(\Delta,\Delta_1)) = O(\Delta^2\Delta_1)$ with the following property. Whenever $Q \notin \mathcal{V}(\Delta,\Delta_1)$, there exists an algorithm recovering P and Q on deterministic polynomial time.

The result is trivial when $4\Delta^2\Delta_1 \geq P$, and so we assume $4\Delta^2\Delta_1 < P$.

The set $\mathcal{V}(\Delta,\Delta_1)$ of primes Q that we are going to exclude consists of values Q satisfying the following congruence:

$$d_1 Q + E \equiv 0 \mod P, \tag{4}$$

where $|d_1| \leq 4\Delta$ and $|E| \leq 8\Delta\Delta_1$. Note that there are at most $O(\Delta^2\Delta_1)$ choices for d_1 and E. Once these parameters are chosen, there can be at most one choice for Q such that $d_1 Q + E \equiv 0 \mod P$. Hence, $\#(\mathcal{V}(\Delta,\Delta_1)) = O(\Delta^2\Delta_1)$.

Suppose that $Q \notin \mathcal{V}(\Delta,\Delta_1)$. An outline of our proof goes as follows. We aim to show that the integers ε_j occur as certain components of a short vector in an appropriate lattice; this lattice can be constructed from the information that we are given. We find ε_0 and ε_1 by using well-known techniques for finding short vectors in lattices, and then we use the equalities $P = \varepsilon_0 + P_0$ and $Q = \varepsilon_1 + Q_0$ to recover P and Q. We obtain

$$f(\varepsilon_0, \varepsilon_1) = 0.$$

More explicitly, the method is derived from the following construction.

$$A = P_0Q_0 - N,\ B = Q_0\Delta,$$
$$C = P_0\Delta_1,\ D = \Delta\Delta_1,$$

then

$$A\Delta\Delta_1 + B\Delta_1\varepsilon_0 + C\Delta\varepsilon_1 + D\varepsilon_0\varepsilon_1 = 0. \tag{5}$$

Therefore, the lattice $\mathcal{L}$ consisting of integer solutions $\boldsymbol{x} = (x_0, x_1, x_2, x_3) \in \mathbb{Z}^4$ of the system of congruence equations:

$$\begin{aligned} &Ax_0 + Bx_1 + Cx_2 + Dx_3 = 0, \\ &x_0 \equiv 0 \mod \Delta\Delta_1, \\ &x_1 \equiv 0 \mod \Delta_1, \\ &x_2 \equiv 0 \mod \Delta, \end{aligned} \tag{6}$$

contains the vector

$$\boldsymbol{e} = (\Delta\Delta_1 e_0, \Delta_1 e_1, \Delta e_2, e_3) = (\Delta\Delta_1, \Delta_1\varepsilon_0, \Delta\varepsilon_1, \varepsilon_0\varepsilon_1). \tag{7}$$

We aim to show that $\boldsymbol{e}$ is a small vector in the lattice $\mathcal{L}$. We have:

$$e_0 = 1, \qquad |e_1| \le \Delta \qquad |e_2| \le \Delta_1, \qquad |e_3| \le \Delta\Delta_1.$$

Using the bounds given in Equation (3), the Euclidean norm of $\boldsymbol{e}$ satisfies the inequality

$$\|\boldsymbol{e}\| \le \sqrt{\Delta^2\Delta_1^2 + \Delta^2\Delta_1^2 + \Delta^2\Delta_1^2 + \Delta^2\Delta_1^2} = 2\Delta\Delta_1. \tag{8}$$

Assume there is another vector $\boldsymbol{f} = (\Delta\Delta_1 f_0, \Delta_1 f_1, \Delta f_2, f_3) \in \mathcal{L}$ with

$$\|\boldsymbol{f}\| \le \|\boldsymbol{e}\| \le 2\Delta\Delta_1.$$

which is not parallel to $\boldsymbol{e}$, in particular:

$$|f_0| \le 2, \qquad |f_1| \le 2\Delta \qquad |f_2| \le 2\Delta_1, \qquad |f_3| \le 2\Delta\Delta_1. \tag{9}$$

We define the vector $\boldsymbol{d} = f_0\boldsymbol{e} - e_0\boldsymbol{f}$. The first component of the vector $\boldsymbol{d}$ is zero, and $\boldsymbol{d}$ lies in the lattice $\mathcal{L}$. Then, the first congruence in Equation (6) implies that

$$B\Delta_1 d_1 + C\Delta d_2 + Dd_3 = 0.$$

Simplifying the above equation:

$$Q_0 d_1 + P_0 d_2 + d_3 = 0, \tag{10}$$

where $d_i = e_i f_0 - f_i$. Note that $|d_i| \le 2|e_i| + |f_i|$ for $i = 1, 2, 3$ and so our bounds on e_i and f_i imply

$$|d_1| < 4\Delta, \quad |d_2| < 4\Delta_1, \quad |d_3| < 4\Delta\Delta_1. \tag{11}$$

From Equation (10), if $d_1 = d_2 = 0$ then $d_3 = 0$. But this implies $\boldsymbol{d} = \boldsymbol{0}$ and so $f_0\boldsymbol{e} = e_0\boldsymbol{f}$. This contradicts the fact that $\boldsymbol{f}$ and $\boldsymbol{e}$ are not parallel.

The case $d_1 = 0 \ne d_2$ is easily avoided because the assumption $4\Delta^2\Delta_1 < p$ implies $p - \Delta > 4\Delta\Delta_1$, and so, the resulting equation $P_0 d_2 + d_3 = 0$ cannot be satisfied.

Making the substitutions $P_0 = P - \varepsilon_0$ and $Q_0 = Q - \varepsilon_1$ in Equation (10) and reducing modulo P, we find that

$$Qd_1 + d_3 - \varepsilon_0 d_2 - \varepsilon_1 d_1 \equiv 0 \mod P. \tag{12}$$

Writing:

$$E = d_3 - \varepsilon_0 d_2 - \varepsilon_1 d_1,$$

we obtain:

$$Qd_1 + E \equiv 0 \bmod P. \tag{13}$$

The bounds (11) imply $|d_1| \leq 4\mathit{\Delta}$ and $|E| \leq 8\mathit{\Delta}\mathit{\Delta}_1$. But then (13) implies that $Q \in \mathcal{V}(\mathit{\Delta}, \mathit{\Delta}_1)$, and so we have a contradiction. This contradiction shows that there exists no small vector $\boldsymbol{f}$ in $\mathcal{L}$ other than vectors parallel to $\boldsymbol{e}$.

We remark that $\mathcal{L}$ is defined using information we are given, and recall that the shortest vector problem can be solved in deterministic polynomial (in the bit size of a given basis of the lattice) time in any fixed dimension, see [15]. This certainly applies to the lattice $\mathcal{L}$. Once we have found a short vector $\boldsymbol{f}$ in $\mathcal{L}$, we know that $\boldsymbol{e} = \boldsymbol{f}/f_0$ since $\boldsymbol{f}$ is parallel to $\boldsymbol{e}$ and since $e_0 = 1$. Obviously, given the second component $\mathit{\Delta}_1\varepsilon_0$ of $\boldsymbol{e}$ we can find P.

To finish the proof, remember that we have assumed to know the number $\lfloor \log P \rfloor$. If we do not have this information we can apply the above algorithm from 1 to $\lfloor \log N \rfloor$. Obviously the time complexity keep polynomial on $\log N$. This completes the proof. □

For practical application of this bound, we note that in most of the cases the values P and Q are taken randomly, then the probability that Q lies in $\mathcal{V}(\mathit{\Delta}, \mathit{\Delta}_1)$ is:

$$\frac{\mathit{\Delta}^2\mathit{\Delta}_1}{P} = \frac{\mathit{\Delta}^3 Q}{P^2}. \tag{14}$$

And this is less than one if:

$$\mathit{\Delta}^3 < \frac{P^2}{Q}. \tag{15}$$

In other words:

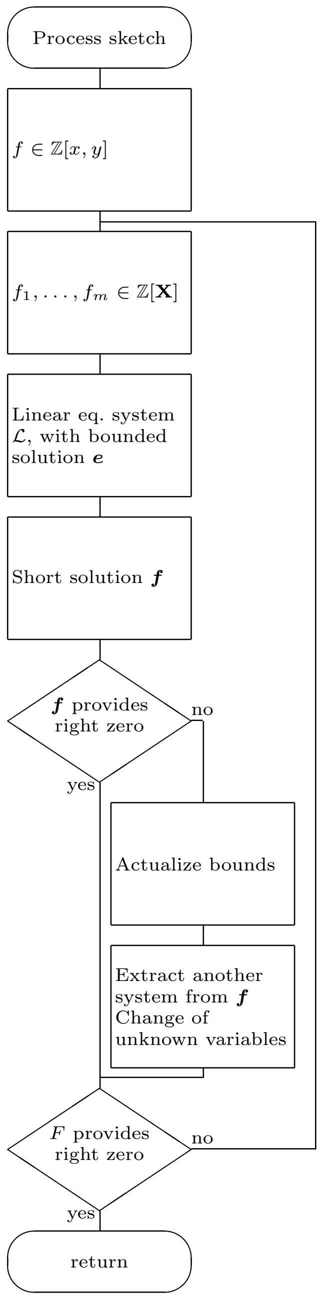
$$\frac{\log N}{3} = \frac{\log P + \log Q}{3} \leq h. \tag{16}$$

If this inequality holds, then we can probably find the complete factorization by this method. In fact this was the first bound obtained by Rivest and Shamir [22]. However, the present algorithm compare favorably computationally speaking with Coppersmith method, because we are working in a lattice of dimension 4 instead of 41, see Remark 1.

The previous theorem can be generalized to two rounds lattice reductions obtaining a better result, see [6]. The details are complicated; moreover, in this case we have to use the Closest Vector Problem CVP (see Section 2) instead of SVP used in the Theorem 3.

4 Numerical results and implementation

The process explained in the previous section, including the generalization proposed, can be sumarized in the following chart:


Process sketch
$f \in \mathbb{Z}[x, y]$
$f_1, \ldots, f_m \in \mathbb{Z}[\mathbf{X}]$
Linear eq. system $\mathcal{L}$, with bounded solution $\boldsymbol{e}$
Short solution $\boldsymbol{f}$
$\boldsymbol{f}$ provides right zero
no
yes
Actualize bounds
Extract another system from $\boldsymbol{f}$ Change of unknown variables
F provides right zero
no
yes
return

We want to provide some experimental results. We have implemented the algorithm introduced in the previous section on a Debian-PC computer with two processors [1Ghz and 1Gb RAM]. Our implementation use the free library **NTL** (Number Theory Library)[21] together with the library **GMP** (GNU Multi Precision Library).

We present the result of some tests in the following table:

Bits of P	Bits of Q	Bits known	Iterations	Time
100	100	66	1	3.675 sec
100	100	60	2	10.271 sec
512	512	306	2	11.392 sec
512	512	300	3	14.025 sec
512	512	292	3	15.339 sec
1024	1024	624	2	7.240 sec
1024	1024	600	3	29. 357 sec
1024	1024	580	3	1 m. 35 sec

In every case choosen, the algorithm has returned the factorization. The first two cases are toy examples. The next three ones are not trivial: factoring a 1024 bits integer is not easy at all. On the negative side, the results are quite distant from Coppersmith result, the reason is that we need to perform more than one round.

5 Conclusion

We have presented an heuristic algorithm for integer factoring with high-order bits known and its implementation in C++ program. Unfortunately, we do not know how to provide a rigorous proof of this method. It is no clear how to evaluate the volume of the associated lattice in each iteration. Partial results appear in [6] and we believe that the approach of this work can be completed.

Our algorithm is asymptotically less efficient than Coppersmith's algorithm, but experiments show that it works well in practice. We would like to compare these experiments with the algorithms in [11, 13].

Finally, we also would like to improve our C++ program. Certainly this question deserves further work.

Acknowledgment. This work is partially supported by Spanish Ministry of Science grant MTM2004-07086.

References

1. M. Ajtai, R. Kumar and D. Sivakumar, 'A sieve algorithm for the shortest lattice vector problem', *Proc. 33rd ACM Symp. on Theory of Comput. (STOC 2001)*, Association for Computing Machinery, 2001, 601–610.

2. L. Babai, ‘On Lovasz Lattice Reduction and the Nearest Lattice Point Problem’, *Combinatorica.*, **6**, 1986, 1–6.
3. S. R. Blackburn, D. Gomez-Perez, J. Gutierrez and I. E. Shparlinski, ‘Predicting nonlinear pseudorandom number generators’, *Math. Computation*, **74** (2005), 1471-1494.
4. J.W.S. Cassels, ‘An Introduction to the Geometry of Numbers’. Springer-Verlag, New York, 1971.
5. D. Gomez-Perez, J. Gutierrez and A. Ibeas, ‘Cryptanalysis of the Quadratic generator’, Proceedings in Cryptology-INDOCRYPT 2005, *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **3797** (2005), 118–129.
6. D. Gomez-Perez, J. Gutierrez and A. Ibeas, ‘An algorithm for integer factoring with high bits known based on lattice basis reduction techniques’, Preprint, *Depart. Mathematics and Computing.*, University of Cantabria, Spain, 2006.
7. D. Coppersmith, ‘Small solutions of small degree polynomials’, *Proc. Intern. Conf. on Cryptography and Lattices*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **2146**, 2001, 20–31.
8. D. Coppersmith: “Small solutions to polynomial equations, and low exponent RSA vulnerabilities”. J. Cryptology **10 (4)**: 233-260, 1997.
9. D. Coppersmith: “Finding a Small Root of a Bivariate Integer Equations; Factoring with High Bits Known”. U. Maurer (Ed), *Proc. EUROCRYPT-96, Lect. Notes in Comp. Sci. Vol. 1070*, Springer-Verlag, Berlin, 1996, 155-156.
10. D. Coppersmith: “Factoring with a hint”. *IBM Research Report RC. 19905*, January 16, 1995.
11. J-S Coron, “Finding small roots of Bivariate Integer Polynomial Equations Revisted”, *Proc. Advances in Cryptology- Eurocrypt’04*, LNCS **3027**, Springer Verlag, 2004, 492–505.
12. J.W.S. Gruber and C.G. Lekkerkerker, ‘Geometry of Numbers’. North-Holland, 1987.
13. N.A. Howgrave-Grahm, “Finding small roots of univariate revisted”, *Proc. Cryptography and Coding*, LNCS **1355**, Springer Verlag, 1997, 131–142.
14. A. Joux and J. Stern, ‘Lattice reduction: A toolbox for the cryptanalyst’, *J. Cryptology*, **11** (1998), 161–185.
15. R. Kannan, ‘Minkowski’s convex body theorem and integer programming’, *Math. Oper. Res.*, **12** (1987), 415–440.
16. A. May, ”Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring”, *In Advances in Cryptology (Crypto 2004), Lecture Notes in Computer Science Volume 3152,* pages 213-219, Springer Verlag, 2004.
17. Paul C. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. *Proc. CRYPTO-96, Lect. Notes in Comp. Sci. Vol. 1109*, Springer-Verlag, Berlin, 1996, 104-113.
18. A. K. Lenstra, H. W. Lenstra and L. Lovász, ‘Factoring polynomials with rational coefficients’, *Mathematische Annalen*, **261** (1982), 515–534.
19. D. Micciancio and S. Goldwasser, *Complexity of lattice problems*, Kluwer Acad. Publ., 2002.
20. P. Q. Nguyen and J. Stern, ‘Lattice reduction in cryptology: An update’, in: W. Bosma (Ed), *Proc. ANTS-IV, Lect. Notes in Comp. Sci. Vol. 1838*, Springer-Verlag, Berlin, 2000, 85–112.
21. V. Shoup, ‘Number theory **C++** library (NTL)’, version 5.3.1, available at `http://www.shoup.net/ntl/`.

22. R. L. Rivest and A. Shamir: “Efficient factoring based on partial information”. Advances in Cryptology, *Proc. EUROCRYPT-85, Lect. Notes in Comp. Sci. Vol. 219*, Springer-Verlag, Berlin, 1986, 31-34.
23. S.A. Vanstone and R. J. Zuccherato: “ Short RSA Keys and their Generation ”. J. Cryptology **8 (2)**: 101-114, 1995.