A Simple and Improved Algorithm for Integer Factorization with Implicit Hints

Koji Nuida*† Naoto Itakura§ Kaoru Kurosawa§

* National Institute of Advanced Industrial Science and Technology (AIST), Japan (k.nuida@aist.go.jp)

† Japan Science and Technology Agency (JST), PRESTO Researcher

§ Ibaraki University, Japan

Abstract

Given two integers $N_1 = p_1q_1$ and $N_2 = p_2q_2$ with α-bit primes q_1, q_2, suppose that the t least significant bits of p_1 and p_2 are equal. May and Ritzenhofen (PKC 2009) developed a factoring algorithm for N_1, N_2 when $t \geq 2\alpha + 3$; Kurosawa and Ueda (IWSEC 2013) improved the bound to $t \geq 2\alpha + 1$. In this paper, we propose a polynomial-time algorithm in a parameter κ, with an improved bound $t = 2\alpha - O(\log \kappa)$; it is the first non-constant improvement of the bound. Both the construction and the proof of our algorithm are very simple; the worst-case complexity of our algorithm is evaluated by an easy argument, without any heuristic assumptions. We also give some computer experimental results showing the efficiency of our algorithm for concrete parameters, and discuss potential applications of our result to security evaluations of existing factoring-based primitives.

1 Introduction

For a large number of computationally secure cryptographic schemes in the literature, including the RSA cryptosystem [10], the (expected) computational hardness of integer factorization is a necessary (and sometimes sufficient) condition for their security. Consequently, the actual hardness of integer factorization has been intensively studied so far, e.g., [4, 5, 9].

Among these work, there exists a direction of studies on integer factorization with *hints*. One of the most remarkable results was given by Coppersmith [1]; the factorization of a composite integer $N = pq$ with primes p, q

becomes efficient when a half of the most significant bits of p are revealed. In the setting, a hint for the factorization is given *explicitly*.

On the other hand, there are also previous results where some *implicit* hints are supposed. May and Ritzenhofen [7] considered the following setting: Given two RSA moduli $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$, it is supposed that the t least significant bits of p_1 and of p_2 are equal. Here the precise values of their t common bits are *not* given; i.e., the hint is only implicit. They showed that, if q_1 and q_2 are α-bit primes and $t \geq 2\alpha + 3$, then N_1 and N_2 can be factorized efficiently. Recently, Kurosawa and Ueda [3] gave an improved algorithm providing a better bound $t \geq 2\alpha + 1$; they also slightly generalized the situation in such a way that $p_1 \equiv p_2 \pmod{T}$ for some parameter $T > {q_1}^2 + {q_2}^2$ (the original case corresponds to $T = 2^t$). In this paper, we improve these results further, yielding a better bound for T.

1.1 Our Contributions

In this paper, we study the integer factorization of composite integers $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ with implicit hint $p_1 \equiv p_2 \pmod{T}$. We aim at developing a polynomial-time algorithm with respect to a certain parameter κ; for example, κ can be the security parameter for some scheme, whose underlying assumption is the hardness of factorizing these composite integers. Then we propose an algorithm to factorize N_1 or N_2 with probability one in polynomial time with respect to the parameter κ under a condition[1]

$$\log T = 2 \log Q - O(\log \kappa) \qquad (1)$$

where Q is an upper bound for q_1, q_2. When $Q = 2^\alpha$ and $T = 2^t$ for integer parameters α and t, our condition above is equivalent to

$$2\alpha - t = O(\log \kappa) \ ,$$

significantly better than the best existing bound $2\alpha - t \leq -1$ in [3].[2] We emphasize that our result is the first result achieving *non-constant* improvement of the bound (in fact, it is even the first to cover the situation $t \leq 2\alpha$).

The essence of our remarkable improvement from the previous results [3, 7] can be explained as follows. In the previous results, a two-dimensional lattice L associated to the given composite integers N_1, N_2 is defined, and it is shown that its *shortest vector*, calculated by Gaussian reduction algorithm,

[1]In fact, some easy-to-satisfy conditions are also required for the sake of completeness.

[2]It was shown in [3] that their algorithm fails (rather than being inefficient) when the bound is not satisfied; hence our result is indeed an improvement of the previous work.

coincides with the vector (q_1, q_2) of the target factors under their condition for T and Q (or t and α, when $T = 2^t$ and $Q = 2^\alpha$). Now we point out that, the Gaussian reduction algorithm outputs not only the shortest vector, but also the *second shortest vector* of the lattice L. Our main idea is to utilize the second shortest vector (as well as the shortest vector) which was not previously used; this new ingredient enabled us to improve the algorithm.

Another noteworthy characteristic of our result is its simplicity; it relies solely on the basic fact that the vector $\vec{q} = (q_1, q_2)$, which lies in the lattice L, can be expressed by using the shortest vector $\vec{v}$ and the second shortest vector $\vec{u}$ of L as $\vec{q} = a\vec{v} + b\vec{u}$ for some integers a, b. Our algorithm finds the correct coefficients a, b by exhaustive search; now our improved condition (1) guarantees that there are only polynomially many (with respect to κ) candidates of (a, b). Our proof is also very simple and elementary; it does not use any typical facts for lattices such as Minkowski bound and Hadamard's inequality (which were used in the previous work [3, 7]).

We performed some computer experiments, which show that our proposed algorithm indeed works efficiently (e.g., the average running time on an ordinary PC was approximately 17 min. for $\alpha = 250$ and $t = 470$). We also discuss potential applications of our proposed algorithm to some existing schemes such as the Okamoto–Uchiyama cryptosystem [8] and Takagi's variant of the RSA cryptosystem [12]; we emphasize that our algorithm does *not* require the implicitly correlated factors p_1, p_2 to be primes.

1.2 Related Work

As mentioned above, for the case of factorization of two integers, our result improves the previous results by May and Ritzenhofen [7] and Kurosawa and Ueda [3]. On the other hand, May and Ritzenhofen also studied factorization of three or more integers which are implicitly correlated in a similar manner. Such an extension of our result is left as a future research topic.

Sarkar and Maitra [11] extended the result of May and Ritzenhofen [7] under a *heuristic* assumption (see Assumption 1 of [11, page 4003]). In a recent preprint [6], Lu et al. announced that they improved the result of Sarkar and Maitra. However, their result is also based on a *heuristic* assumption. In contrast, the evaluation of our algorithm in this paper needs *no such heuristic assumptions*; our algorithm is worst-case polynomial-time for the parameters specified in this paper.

1.3 Organization of the Paper

In Sect. 2, we summarize basic notations and terminology, as well as some properties of Gaussian reduction algorithm for two-dimensional lattice. In Sect. 3, we clarify our problem setting, describe our proposed factorization algorithm, and then show its correctness and computational complexity. In Sect. 4, we give the results of our computer experiments to show the efficiency of our proposed algorithm. Finally, in Sect. 5, we discuss potential applications to security evaluations of some existing cryptographic schemes.

2 Preliminaries

For two-dimensional vectors $\vec{v} = (v_1, v_2), \vec{u} = (u_1, u_2) \in \mathbb{R}^2$, let $||\vec{v}|| = \sqrt{v_1{}^2 + v_2{}^2}$ and $(\vec{v}, \vec{u}) = v_1u_1 + v_2u_2$ denote the Euclidean norm and the standard inner product. For a two-dimensional lattice $L \subset \mathbb{Z}^2$, let $\lambda_1 = \lambda_1(L)$ and $\lambda_2 = \lambda_2(L)$ denote the *successive minima* of L; i.e., λ_i is the minimal radius of a ball containing i linearly independent vectors of L.

We recall that, in a two-dimensional lattice L, a basis $(\vec{v_1}, \vec{v_2})$ of L satisfying $||\vec{v_1}|| = \lambda_1$ and $||\vec{v_2}|| = \lambda_2$ can be efficiently obtained by Gaussian reduction algorithm. Here we describe the algorithm:

Definition 1 (Gaussian reduction algorithm)**.** *Given any basis $(\vec{b_1}, \vec{b_2})$ of a lattice L, Gaussian reduction algorithm performs as follows:*

1. *First, order the vectors $\vec{b_1}, \vec{b_2}$ and rename those as $\vec{v_1}, \vec{v_2}$, in such a way that $||\vec{v_1}|| \le ||\vec{v_2}||$.*
2. *Set $\mu := \lfloor (\vec{v_1}, \vec{v_2})/||\vec{v_1}||^2 \rceil$, i.e., the integer closest to $(\vec{v_1}, \vec{v_2})/||\vec{v_1}||^2$ (if two integers have equal smallest distance from the value, then choose the one with smaller absolute value).*
3. *Repeat the following, until μ becomes 0:*
   - *(a) Update $\vec{v_2}$ by $\vec{v_2} \leftarrow \vec{v_2} - \mu \vec{v_1}$.*
   - *(b) If $||\vec{v_2}|| < ||\vec{v_1}||$, then swap $\vec{v_1}$ and $\vec{v_2}$.*
   - *(c) Set $\mu := \lfloor (\vec{v_1}, \vec{v_2})/||\vec{v_1}||^2 \rceil$.*
4. *Output the pair $(\vec{v_1}, \vec{v_2})$.*

The following property is well-known; see e.g., [2]:

Proposition 1. *The Gaussian reduction algorithm outputs a basis $(\vec{v_1}, \vec{v_2})$ of the lattice L satisfying that $||\vec{v_1}|| = \lambda_1$ and $||\vec{v_2}|| = \lambda_2$. Moreover, the computational complexity of the algorithm is $O(\log^2 \max\{||\vec{b_1}||, ||\vec{b_2}||\})$.*

We also use the following property of Gaussian reduction algorithm:

Lemma 1. *For any input $(\vec{b_1}, \vec{b_2})$ and the corresponding output $(\vec{v_1}, \vec{v_2})$ of Gaussian reduction algorithm, we have $|\det(\vec{b_1}, \vec{b_2})| = |\det(\vec{v_1}, \vec{v_2})|$, where we write* $\det((x_1, x_2), (y_1, y_2)) := \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1$.

Proof. The transformations for $(\vec{v_1}, \vec{v_2})$ performed at each step of Gaussian reduction algorithm are one of the followings:

- Subtract a scalar multiple of $\vec{v_1}$ from $\vec{v_2}$; it preserves the value $\det(\vec{v_1}, \vec{v_2})$.
- Swap $\vec{v_1}$ and $\vec{v_2}$; it changes the value $\det(\vec{v_1}, \vec{v_2})$ to $-\det(\vec{v_1}, \vec{v_2})$.

Hence, the absolute value of $\det(\vec{v_1}, \vec{v_2})$ is not changed, as desired. □

3 Our Proposed Algorithm

3.1 Problem Setting

Let $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ be given composite numbers. Let $T \geq 2$ be an integer parameter (for example, a power of two as in [7]) with $T < N_1$ and $T < N_2$. In this paper, we consider the following situation:

- We have $p_1 \equiv p_2 \equiv p \pmod{T}$ for some *unknown* integer p.
- Any two of N_1, N_2 and T are coprime to each other.

When $T = 2^t$ for an integer t, the first condition means that the t least significant bits of p_1 and p_2 are equal (the precise t bits are not known). We emphasize that *we do NOT assume that each of p_1, p_2, q_1 and q_2 is a prime.* The second condition implies that any two of q_1, q_2 and T are coprime to each other, and p is coprime to T (indeed, if p and T have a common divisor $a > 1$, then p_1 and p_2, hence N_1 and N_2, are multiples of a, a contradiction).

3.2 The Algorithm

In order to describe our proposed algorithm, first we define, for given composite numbers N_1 and N_2, the following two-dimensional lattice L:

$$L := \{(x_1, x_2) \in \mathbb{Z}^2 \mid N_2 x_1 - N_1 x_2 \equiv 0 \pmod{T}\} \ .$$

We have a basis of L consisting of two vectors $(1, N_2/N_1 \bmod T)$ and $(0, T)$, where $N_2/N_1 \bmod T$ signifies the unique integer a in $[0, T-1]$ with $N_1 a \equiv N_2 \pmod{T}$. It is indeed a basis of L, since N_1 and T are coprime; if $(0, x_2) \in L$, then we have $N_1 x_2 \equiv 0 \pmod{T}$, therefore x_2 must be a multiple of T.

Then we describe our proposed algorithm to find a non-trivial factor of at least one of the given composite numbers N_1 and N_2:

1. Compute, by Gaussian reduction algorithm with initial basis consisting of $(1, N_2/N_1 \bmod T)$ and $(0, T)$, a basis $(\vec{v} = (v_1, v_2), \vec{u} = (u_1, u_2))$ of the lattice L above with $||\vec{v}|| = \lambda_1 = \lambda_1(L)$ and $||\vec{u}|| = \lambda_2 = \lambda_2(L)$.
2. Compute $\gcd(v_1, N_1)$, $\gcd(v_2, N_2)$, $\gcd(u_1, N_1)$ and $\gcd(u_2, N_2)$, and if at least one of those is different from 1, then output it and halt.
3. If $v_1 u_2 - v_2 u_1 < 0$, then replace $\vec{u}$ with $-\vec{u}$.
4. For $A = 2, 3, \ldots$, execute the following:
   (a) For integers $a, b \neq 0$ satisfying $|a| + |b| = A$, execute the following: If $|au_1 - bv_1|$ is a non-trivial factor of N_1, then output it and halt.

3.3 Analysis of Our Algorithm

We analyze the correctness and the efficiency of our proposed algorithm. First, note that (since $T \geq 2$)

$$||(1, N_2/N_1 \bmod T)|| \leq \sqrt{1^2 + (T-1)^2} < T = ||(0, T)|| \ , \qquad (2)$$

therefore by Proposition 1, the complexity of Step 1 of our algorithm (consisting of Gaussian reduction algorithm) is $O(\log^2 T)$. Secondly, the lattice L contains the vector $\vec{q} := (q_1, q_2)$; indeed, we have

$$N_2 q_1 - N_1 q_2 = p_2 q_2 q_1 - p_1 q_1 q_2 \equiv p q_2 q_1 - p q_1 q_2 = 0 \pmod{T} \ .$$

Now we show the following property for Step 2 of our algorithm:

Lemma 2. *If our algorithm halts in Step 2, then the output of the algorithm is correctly a non-trivial factor of either N_1 or N_2. Moreover, if $||\vec{q}|| < \lambda_2$, then our algorithm always halts in Step 2.*

Proof. We have $\lambda_2 \leq T$ by (2), therefore $\lambda_2 < N_1$ and $\lambda_2 < N_2$ by the condition in Sect. 3.1. This implies that all of $|v_1|$, $|v_2|$, $|u_1|$ and $|u_2|$ are smaller than N_1 and N_2. Hence, $\gcd(v_1, N_1)$ will be a non-trivial factor of

N_1 if $\gcd(v_1, N_1) \neq 1$, and the same holds for $\gcd(v_2, N_2)$, $\gcd(u_1, N_1)$ and $\gcd(u_2, N_2)$. This deduces the first part of the claim.

For the second part, if $||\vec{q}|| < \lambda_2$, then $\vec{q}$ and $\vec{v}$ are linearly dependent by the definition of $\lambda_2 = \lambda_2(L)$; $c\vec{v} = c'\vec{q}$ for some coprime integers $c, c' \neq 0$. Since q_1 and q_2 are coprime, we have $|c| = 1$ and $\vec{v} = \pm c'\vec{q}$. Moreover, since $||\vec{q}|| \geq ||\vec{v}||$ by the choice of $\vec{v}$, we have $|c'| = 1$. Therefore, we have $|v_1| = q_1$ and $\gcd(v_1, N_1) = q_1 \neq 1$. This completes the proof of Lemma 2. $\square$

Note that the computation of gcd in Step 2 can be done in polynomial time with respect to $\max\{\log N_1, \log N_2\}$. By virtue of Lemma 2, to see the correctness of our algorithm, we may focus on the case that the algorithm does not halt at Step 2. Now we have $\lambda_2 \leq ||\vec{q}||$ by Lemma 2.

Since $p_1 \equiv p_2 \equiv p \pmod{T}$ and $\vec{v}, \vec{u} \in L$, we have

$$p(q_2 v_1 - q_1 v_2) \equiv p(q_2 u_1 - q_1 u_2) \equiv 0 \pmod{T} \ .$$

Moreover, since $\gcd(p, T) = 1$ as mentioned in Sect. 3.1, it follows that

$$q_2 v_1 - q_1 v_2 \equiv q_2 u_1 - q_1 u_2 \equiv 0 \pmod{T} \ .$$

Hence, there are integers $a_0, b_0 \in \mathbb{Z}$ satisfying

$$q_2 v_1 - q_1 v_2 = a_0 T \, , \; q_2 u_1 - q_1 u_2 = b_0 T \ , \tag{3}$$

or equivalently

$$\begin{pmatrix} -v_2 & v_1 \\ -u_2 & u_1 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} a_0 T \\ b_0 T \end{pmatrix} \ . \tag{4}$$

We have $a_0 \neq 0$ by (3), since q_1 is coprime to q_2 and v_1 (note that v_1 is coprime to $N_1 = p_1 q_1$, since our algorithm does not halt at Step 2 by the current assumption). Similarly, we have $b_0 \neq 0$. Now Lemma 1 implies that

$$\det\begin{pmatrix} -v_2 & v_1 \\ -u_2 & u_1 \end{pmatrix} = \det\begin{pmatrix} v_1 & v_2 \\ u_1 & u_2 \end{pmatrix} = \pm\det\begin{pmatrix} 1 & N_2/N_1 \bmod T \\ 0 & T \end{pmatrix} = \pm T \ ,$$

while $\det\begin{pmatrix} v_1 & v_2 \\ u_1 & u_2 \end{pmatrix} = v_1 u_2 - v_2 u_1 \geq 0$ by virtue of Step 3 of our algorithm, therefore we have $\det\begin{pmatrix} -v_2 & v_1 \\ -u_2 & u_1 \end{pmatrix} = T$. Hence the system of equations (4) can be solved as

$$\begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} -v_2 & v_1 \\ -u_2 & u_1 \end{pmatrix}^{-1} \begin{pmatrix} a_0 T \\ b_0 T \end{pmatrix} = \frac{1}{T} \begin{pmatrix} u_1 & -v_1 \\ u_2 & -v_2 \end{pmatrix} \begin{pmatrix} a_0 T \\ b_0 T \end{pmatrix} = \begin{pmatrix} a_0 u_1 - b_0 v_1 \\ a_0 u_2 - b_0 v_2 \end{pmatrix} \ .$$

Consequently, if the pair (a, b) in Step 4a of our algorithm becomes (a_0, b_0), then our algorithm halts with output q_1, which is indeed a non-trivial factor of N_1. This completes the proof of the property that our algorithm halts within a finite computational time and its output is always a non-trivial factor of either N_1 or N_2 (we note that $|a_0| + |b_0| \geq 2$, since $a_0, b_0 \neq 0$).

From now, we evaluate the number of iterations in Step 4, by evaluating the sizes of a_0 and b_0 above. For the purpose, we introduce the following additional assumption, where Q is an integer parameter:

- We have $q_1, q_2 \leq Q$ for any given N_1, N_2.

We emphasize that the parameter Q is used in the analysis of the algorithm only, and is not needed by our algorithm itself. By Lemma 2, we may focus on the case $\lambda_2 \leq ||\vec{q}||$; otherwise, our algorithm halts at Step 2. Note that $||\vec{q}|| = \sqrt{{q_1}^2 + {q_2}^2} \leq \sqrt{2} \cdot Q$, therefore $\lambda_2 \leq \sqrt{2} \cdot Q$. Now by (3), we have

$$|a_0| = \left| \frac{q_2 v_1 - q_1 v_2}{T} \right| \leq \frac{|q_2 v_1| + |q_1 v_2|}{T} \leq \frac{Q}{T}(|v_1| + |v_2|) \leq \frac{Q}{T}\sqrt{2} \cdot ||\vec{v}||$$

and similarly $|b_0| \leq (Q/T)\sqrt{2} \cdot ||\vec{u}||$. Since $||\vec{v}|| \leq ||\vec{u}|| = \lambda_2$ by the choice of $\vec{v}$ and $\vec{u}$, it follows that

$$|a_0|, |b_0| \leq \frac{Q}{T}\sqrt{2} \cdot \lambda_2 \leq \frac{2Q^2}{T} \enspace ,$$

therefore $|a_0| + |b_0| \leq 4Q^2/T$. Hence, the index A in Step 4 of our algorithm does not exceed $A_0 := \lfloor 4Q^2/T \rfloor$ during the execution. Since Step 4a of our algorithm is repeated at most $4A$ times for each choice of A, the total number of executions of Step 4a is at most $\sum_{A=2}^{A_0} 4A = 2A_0(A_0 + 1) - 4$. Moreover, for each $1 \leq A \leq A_0$, Step 4a for each choice of (a, b) can be done in polynomial time with respect to $\log A_0$, $\log Q$ and $\log N_1$ (note that $|a|, |b| \leq A_0$ and $|v_1|, |u_1| \leq \lambda_2 \leq \sqrt{2} \cdot Q$).

Summarizing the argument, our algorithm runs in polynomial time with respect to the maximum among $\log^2 T$, $\log N_1$, $\log N_2$, $\log(4Q^2/T)$, $\log Q$ and $4Q^2/T$. Here, the values $\log(4Q^2/T)$ ($\leq 4Q^2/T$) and $\log^2 T$ ($\leq \log N_1$, since $T < N_1, N_2$) are redundant. Moreover, we have $\max\{4Q^2/T, \log N_1\} \geq \log Q$; indeed, if $4Q^2/T < \log Q$, then we have $4Q^2/N_1 < \log Q$ (since $T < N_1$), $N_1 > 4Q^2/\log Q > 4Q$, and $\log N_1 > \log Q$. Therefore, the value $\log Q$ above is also redundant. Hence, we have the following result:

Theorem 1. *In the setting of Sect. 3.1, suppose that $q_1, q_2 \leq Q$. Then our proposed algorithm in Sect. 3.2 always outputs a non-trivial factor of either N_1 or N_2, and its computational complexity is polynomially bounded with respect to* $\max\{\log N_1, \log N_2, Q^2/T\}$.

By Theorem 1, if κ is another parameter (e.g., when the factorization problem we are discussing is the base of security of some cryptographic scheme, κ can be chosen as the security parameter for the scheme) and all of $\log N_1$, $\log N_2$ and Q^2/T are of polynomial order with respect to κ, then our proposed algorithm runs in polynomial time with respect to κ.

For example, we consider the case that q_1 and q_2 are α-bit integers and the t least significant bits of p_1 and p_2 coincide with each other (as in the previous work [3, 7]). Then Theorem 1 implies the following result:

Theorem 2. *Let κ be a parameter as mentioned above. Suppose that the bit lengths of N_1 and N_2 are polynomial in κ, and let $Q = 2^{\alpha}$ and $T = 2^{t}$. Then our proposed algorithm runs in polynomial time with respect to κ if*

$$t = 2\alpha - O(\log \kappa) \ .$$

This sufficient condition for t is significantly improved from the conditions $t \geq 2\alpha + 3$ in [7] and $t \geq 2\alpha + 1$ in [3]. In particular, this is the first result achieving that the difference $2\alpha - t$ can be beyond of constant order.

4 Computer Experiments

We performed a computer experiment to evaluate the running time of our proposed algorithm; see Figure 1. Here we set $Q = 2^{\alpha}$, $\alpha = 250$ (i.e., q_1 and q_2 are 250-bit primes), $T = 2^{t}$, and the bit length t of implicit hints is chosen as $t = 501, 500, \dots, 470$. The other factors p_1 and p_2 have 750-bit lengths. We used an ordinary machine environment, namely our algorithm is written in C++ with NTL for large-integer arithmetic, on CentOS 6.5 with 2.4GHz CPU and 32GB RAM. For each t, we calculated the average running time of our algorithm for 100 experiments (N_1 and N_2 are correctly factorized at every experiment). Our experimental result shows that our algorithm can successfully factorize the integers efficiently, even for a significantly better parameter $t = 470$ than the best bound $t \geq 2\alpha + 1 = 501$ in the previous results (now the average running time is approximately 1030 sec. $\approx$ 17 min.).

We also evaluated the sufficient number A of iterations for the main loop of our proposed algorithm by computer experiments. We used the same parameters and machine environment as above, except that the range of the bit length t of implicit hints is now $t = 499, 498, \dots, 475$. For each t, we calculated the maximum, average, and minimum of the numbers of iterations for 100 experiments; see Figure 2 (the factorization succeeded at every experiment again). We note that the upper bound of A given in our

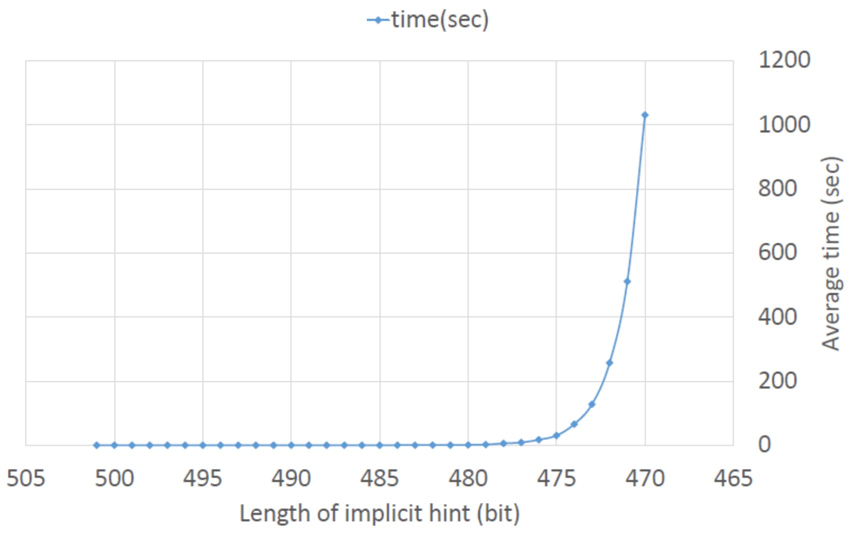

Figure 1: Running time of our proposed algorithm (here the bit lengths of q_1 and q_2 are $\alpha = 250$ bits, $T = 2^t$, and the range of t is $\{501, 500, \dots, 470\}$)

theoretical analysis in Sect. 3.3 is $\lfloor 4Q^2/T \rfloor = 2^{502-t}$; it is, for example, $2^{27} \approx 1.34 \times 10^8$ for $t = 475$. Our experimental result suggests that this theoretical bound of A would still be far from the precise value; further analyses to improve the bound of A are left as a future research topic.

5 Potential Applications

It is noteworthy that the implicitly correlated factors p_1, p_2 need not be primes in our proposed algorithm; see Sect. 3.1. This widens the potential applications of our method to security evaluations of existing schemes. In this section, we consider the cases of the Okamoto–Uchiyama cryptosystem [8] (Sect. 5.1) and Takagi's variant of the RSA cryptosystem [12] (Sect. 5.2).

5.1 Okamoto–Uchiyama Cryptosystem

In the Okamoto–Uchiyama cryptosystem [8], the public key involves a composite number of the form $n = (p')^2 \cdot q'$, where p' and q' are different large primes of the same bit length. Here p' and q' should be secret against the adversary; a necessary condition for the security of the scheme is the hardness of factorizing the integer n. Now we regard the integers $(p')^2$ and q' as

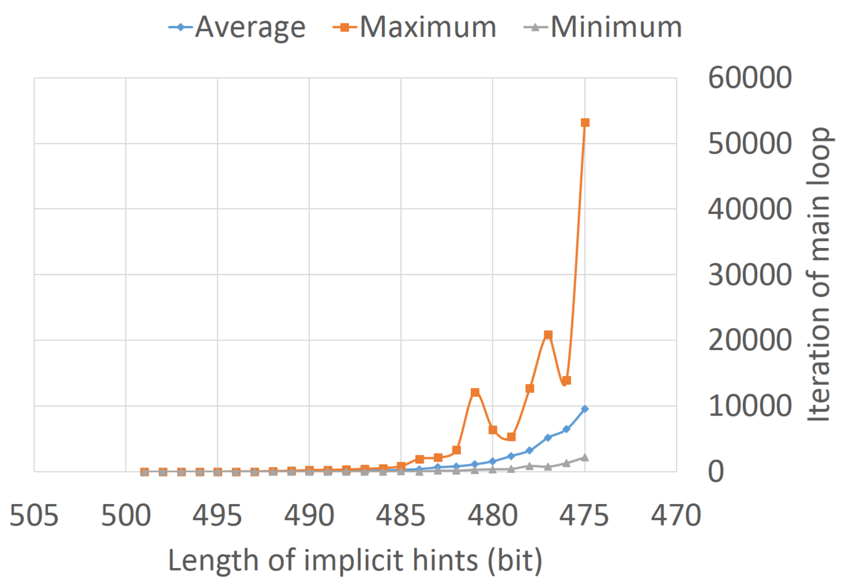

Figure 2: Number A of iterations for the main loop (here the bit lengths of q_1 and q_2 are $\alpha = 250$ bits, $T = 2^t$, and the range of t is $\{499, 498, \ldots, 475\}$)

p_i and q_i in our algorithm, respectively; we emphasize again that the factor p_i in our method is not necessarily a prime.

More precisely, given two public keys $n_1 = {p'_1}^2 \cdot q'_1$ and $n_2 = {p'_2}^2 \cdot q'_2$ of the Okamoto–Uchiyama cryptosystem, we consider the following situation: ${p'_1}^2 \equiv {p'_2}^2 \pmod{T}$ and $q'_1, q'_2 \leq Q$, where T and Q are parameters. To simplify the argument, we set $Q := 2^\alpha$ where α is the common bit length of p'_i and q'_i. Then our proposed algorithm factorizes at least one of n_1 and n_2 in polynomial time with respect to the security parameter κ, if Q^2/T is of polynomial order in κ, or equivalently, if $2\alpha - \log_2 T = O(\log \kappa)$.

From now, we discuss the frequency of the condition ${p'_1}^2 \equiv {p'_2}^2 \pmod{T}$ being satisfied, in the situation of the previous work [3, 7] and our situation. First, in the situation of [3, 7], T and Q should satisfy $\log_2 T \geq 2\alpha + 1$, therefore $T \geq 2{p'_1}^2$ and $T \geq 2{p'_2}^2$. Now the condition ${p'_1}^2 \equiv {p'_2}^2 \pmod{T}$ implies that ${p'_1}^2 = {p'_2}^2$ *as integers*, i.e., $p'_1 = p'_2$, which is a trivial case. This means that the algorithms in [3, 7] cannot be applied to the present case.

In contrast, in our method, the parameter $\log_2 T$ may be smaller than 2α, hence there *is* a (non-trivial) possibility of the case ${p'_1}^2 \equiv {p'_2}^2 \pmod{T}$. Going into detail, ${p'_1}^2 \equiv {p'_2}^2 \pmod{T}$ is equivalent to $p'_1 - p'_2 \equiv 0 \pmod{T_1}$

and $p'_1 + p'_2 \equiv 0 \pmod{T_2}$ for some factorization $T = T_1T_2$ of T. Hence, to increase the possibility of the case ${p'_1}^2 \equiv {p'_2}^2 \pmod{T}$, it would be better to use the parameter T with many possibilities of appropriate factorizations $T = T_1T_2$. Now if T_1 and T_2 have an odd common divisor $d > 1$, then $2p'_1$ and $2p'_2$, hence p'_1 and p'_2, are multiples of d. This is not desirable, since p'_1 and p'_2 are primes. By the observation, it seems better to use a smooth and square-free T; then the number of possible factorizations $T = T_1T_2$ with *coprime* factors T_1, T_2 is increased. For example, we may let T be the product of all primes smaller than a certain threshold. For such parameters T, further evaluations of how frequently given two composite numbers n_1, n_2 satisfy the condition above are left as a future research topic.

5.2 Takagi's Variant of RSA

A similar argument is also applicable to Takagi's variant of the RSA cryptosystem [12]. In the scheme, the public key involves a composite number of the form $N = (p')^r \cdot q'$, where p' and q' are different large primes of the same bit length and $r \geq 2$. We regard the integers $(p')^r$ and q' as p_i and q_i in our algorithm, respectively. Since the case $r = 2$ is essentially the same as the case of the Okamoto–Uchiyama cryptosystem (Sect. 5.1), here we focus on the other case $r \geq 3$. In the case, the bit length of the factor $(p')^r$ becomes much larger than that of the other factor q', which would make the condition ${p'_1}^r \equiv {p'_2}^r \pmod{T}$ easier to satisfy under the requirement $\log_2 T = 2\log_2 Q - O(\log \kappa)$ for our proposed algorithm. On the other hand, when $r \geq 3$, the analysis of the condition ${p'_1}^r \equiv {p'_2}^r \pmod{T}$ would be more difficult than the condition ${p'_1}^2 \equiv {p'_2}^2 \pmod{T}$ in the case of the Okamoto–Uchiyama cryptosystem. A detailed analysis of our method in relation to Takagi's RSA is left as a future reserach topic.

Acknowledgements. The authors thank the members of Shin-Akarui-Angou-Benkyo-Kai for their precious comments on this paper, especially Goichiro Hanaoka for his suggestion about the potential applications of our result mentioned in Sect. 5.

References

[1] D. Coppersmith, Finding a Small Root of a Bivariate Integer Equation, Factoring with High Bits Known, in: Proceedings of EUROCRYPT 1996, LNCS 1070, 178–189 (1996)

[2] S. D. Galbraith, Mathematics of Public Key Cryptography, Cambridge University Press (2012)

[3] K. Kurosawa and T. Ueda, How to Factor N_1 and N_2 When $p_1 = p_2 \bmod 2^t$, in: Proceedings of IWSEC 2013, LNCS 8231, 217–225 (2013)

[4] H. W. Lenstra Jr., Factoring Integers with Elliptic Curves, Ann. Math. 126, 649–673 (1987)

[5] A. K. Lenstra and H. W. Lenstra Jr., The Development of the Number Field Sieve, Springer, Heidelberg (1993)

[6] Y. Lu, L. Peng, R. Zhang and D. Lin, Towards Optimal Bounds for Implicit Factorization Problem, IACR Cryptology ePrint Archive 2014/825 (2014)

[7] A. May and M. Ritzenhofen, Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint, in: Proceedings of PKC 2009, LNCS 5443, 1–14 (2009)

[8] T. Okamoto and S. Uchiyama, A New Public-Key Cryptosystem as Secure as Factoring, in: Proceedings of EUROCRYPT 1998, LNCS 1403, 308–318 (1998)

[9] C. Pomerance, The Quadratic Sieve Factoring Algorithm, in: Proceedings of EUROCRYPT 1984, LNCS 209, 169–182 (1985)

[10] R. L. Rivest, A. Shamir and L. M. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Commun. ACM 21(2), 120–126 (1978)

[11] S. Sarkar and S. Maitra, Approximate Integer Common Divisor Problem Relates to Implicit Factorization, IEEE Transactions on Information Theory 57(6), 4002–4013 (2011)

[12] T. Takagi, Fast RSA-Type Cryptosystem Modulo $p^k q$, in: Proceedings of CRYPTO 1998, LNCS 1462, 318–326 (1998)