Extension of simultaneous Diophantine approximation algorithm for partial approximate common divisor variants

Wonhee Cho 1 | Jiseung Kim 2 | Changmin Lee 2

1 Department of Mathematical Sciences, Seoul National University, Seoul, South Korea
2 School of Computational Sciences, Korea Institute for Advanced Study, Seoul, Republic of Korea

Correspondence
Changmin Lee, School of Computational Sciences, Korea Institute for Advanced Study, 85 Hoegi-ro, Dongdaemun-gu, Seoul, Republic of Korea.
Email: changminlee@kias.re.kr

Funding information
Korea Institute for Advanced Study Individual Grant, Grant/Award Numbers: CG078201, CG080601; Institute for Information and Communications Technology Promotion, Grant/Award Number: No. 2016-6-00598

Abstract
A simultaneous Diophantine approximation (SDA) algorithm takes instances of the partial approximate common divisor (PACD) problem as input and outputs a solution. While several encryption schemes have been published the security of which depend on the presumed hardness of variants of the PACD problem, fewer studies have attempted to extend the SDA algorithm to be applicable to these variants. In this study, the SDA algorithm is extended to solve the general PACD problem. In order to proceed, first the variants of the PACD problem are classified and how to extend the SDA algorithm for each is suggested. Technically, the authors show that a short vector of some lattice used in the SDA algorithm gives an algebraic relation between secret parameters. Then, all the secret parameters can be recovered by finding this short vector. It is also confirmed experimentally that this algorithm works well.

1 | INTRODUCTION

Simultaneous Diophantine approximation (SDA) algorithms have been proposed to analyse the partial approximate common divisor (PACD) problem suggested by Howgrave-Graham [1]. Informally, PACD is the problem of recovering a secret prime $p$ when approximate multiples of the form $x_i = p \cdot q_i + r_i$ for a small integer term $r_i$ are given. Since the PACD problem was exploited to construct the fully homomorphic encryption over integers suggested by Dijk, Gentry, Halévi, and Vaikuntanathan (called the DGHV scheme) [2], both PACD and SDA are highlighted.

Shortly thereafter, several variants of the PACD problem were exploited. To encrypt multiple messages at once, using the DGHV scheme for its efficiency, Cheon et al. proposed a variant of the PACD problem and suggested batch fully homomorphic encryption over the integers [3]. In addition, Coron, Lepoint, and Tibouchi proposed a practical candidate for the multilinear map employing the variant of PACD [4]. In Cheon et al.’s paper, this new base problem is defined as $\ell$-DACD, while Galbraith, Gebregiyorgis, and Murphy referred to it as CRT-ACD [5]. To avoid confusion, throughout this paper, we define the variant problem as CCK-ACD. Similarly, we redefine the problem used in Coron et al.’s paper as scaled CRT-ACD; because these primitives offer improved functionality and efficiency, they have been used in a variety of applications, especially homomorphic encryption over the integers, and indistinguishability obfuscations [6–20].

Although these problems share common mathematical structures, both have identifying characteristics. Informally, CCK-ACD uses an extra-large prime factor, and scaled CRT-ACD is multiplied by a common secret constant. According to these two characteristics, we define scaled CCK-ACD and CRT-ACD and classify four distinct problems. The following four problems are defined as finding the $\eta$-bit integer factor of $N$ using given $\gamma$-bit instances and the $\gamma$-bit integer $N$. For each problem, an integer $N$ and instances $\{b_j\}_{1 \leq j \leq \tau}$ are of the following forms:

Wonhee Cho and Jiseung Kim are co-first authors.

This is an open access article published by John Wiley & Sons, Ltd on behalf of The Institution of Engineering and Technology.

© 2021 The Authors. *IET Information Security* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.
where the inequality $\rho \ll \eta \ll \gamma$ holds.

Because the SDA algorithm was only proposed for solving the P\-ACD problem, analysing variant problems emerged as follow-up research. Recently, Cheon et al. first suggest an extended SDA algorithm to solve the CCK-ACD problem $[21]$ under the condition $n \cdot \left(\frac{2 \log \beta}{\beta - 1} + 1\right) = O(\eta)$. However, to date, extending the SDA algorithm to solve the other variants is not clearly established.

### 1.1 | Our work

The original SDA algorithm for P\-ACD described by Howgrave-Graham (see Section 2 of [1]) is designed to find a short vector of the column lattice $L$ generated by the matrix

$$B = \begin{pmatrix}
1 & 0 & 0 & \cdots & 0 \\
b_1 & N & 0 & \cdots & 0 \\
b_2 & 0 & N & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
b_k & 0 & 0 & \cdots & N
\end{pmatrix},$$

where $N = p \cdot q_0$ and $b_i$'s are P\-ACD instances of the form $p \cdot q_0 + r_i$. We note that the lattice $L$ then contains a short vector $(q_0, q_0 \cdot r_1, \ldots, q_0 \cdot r_k)^T$. Therefore, computing this short vector is equivalent to recovering a non-trivial factor of $N$ in the case of P\-ACD. However, there is no indication that the short vector of the SDA algorithm on other variants of P\-ACD problem cases is related to the prime factors.

In this paper, we present SDA-variant algorithms for solving scaled CCK-ACD as well as CRT-ACD and its scaled version. In other words, SDA is also applicable to problems beyond P\-ACD and CCK-ACD. Thus, we can obtain all the secret factors of this class of problems.

Our algorithms employ lattice reduction algorithms such as the BKZ algorithm to obtain a short vector generated by multiple instances of the target problems, and the short vector obtained is used to reconstruct algebraic relations for recovering secret primes, similar to previous approaches $[21–23]$. Asymptotically, we can find all prime factors of CRT-ACD and scaled CRT-ACD under the condition $n \cdot \left(\frac{2 \log \beta}{\beta - 1} + 1\right) = O(\eta)$.

More precisely, we first suggest a new algorithm to solve the CRT-ACD problem under the condition

$$n \cdot \left(\frac{2 \log \beta}{\beta - 1} + 1\right) \leq \eta - 5 \rho - 4 \log n - 3 \log \beta - 5,$$

where $\beta$ is the block size of the BKZ algorithm, and $\eta$ and $\rho$ are the bit-sizes of primes $p_i$ and integers $r_i$ respectively. Under the condition, our algorithm shows that all secret factors can be recovered in time $\max\{\text{poly}(\eta, \beta), T(A_\beta)\}$, where $T(A_\beta)$ is the time complexity for running the BKZ algorithm with a block size $\beta$.

Second, we provide a similar algorithm to solve the scaled CRT-ACD problem. More precisely, the scaled problem provides instances of the form $c \cdot \text{CRT} (p_1, \ldots, p_n) (r_1, \ldots, r_n)$ and $N = \prod_{i=1}^n p_i$ for an unknown fixed constant $c \in \mathbb{Z}_N$, instead of the original instances $\text{CRT} (p_1, \ldots, p_n) (r_1, \ldots, r_n)$, and $N$. By exploiting a similar algorithm, we show that the problem is solved if

$$n \cdot \left(\frac{2 \log \beta}{\beta - 1} + 1\right) \leq \eta - 10 \rho - 4 \log n - 3 \log \beta - 5.$$

Similarly, the algorithm also takes time complexity $\max\{\text{poly}(\eta, \beta), T(A_\beta)\}$, where $T(A_\beta)$ is the time complexity for running the BKZ algorithm with a block size $\beta$.

Finally, we produce an algorithm for reducing the scaled CCK-ACD problem to the scaled CRT-ACD problem under the condition

$$2 \sqrt{\gamma \cdot \log \beta / \beta} \leq \eta - 2 \rho + \log \sqrt{2 \gamma - 3 \cdot \log \beta} - 2.$$

Thus, we can find all secret factors of scaled CCK-ACD through two steps: (1) reducing to the scaled CCK-ACD problem (2) solving the scaled CRT-ACD problem.

**Related work.** In addition to the SDA algorithm, orthogonal lattice attack (OLA) is another way to solve the P\-ACD problem. For example, Coron and Pereira recently proposed a result to solve the scaled CRT-ACD problem by extending the OLA $[23]$. Compared with our extended SDA algorithm, their approach has the same complexity for solving the scaled CRT-ACD problem.

The extended OLA and our SDA-type algorithm commonly aim at recovering a linear summation of $r_{i,j}$, where $r_{i,j}$ is an integer of each ACD variant problem. Once obtaining it, the next step for recovering the secret prime factor is the same. Both algorithms use distinct algorithms to get the linear summation.
As the name suggests, the extended OLA for ACD variants considers an orthogonal lattice to find the linear signature.

Let \( \{b_j\}_{1 \leq j \leq n} \) be scaled CCK-ACD instances with \( b_j \equiv p_j \cdot r_{ij} \cdot c \). Let’s denote \( \mathbf{b} \) and \( \mathbf{r} \) as \( t \)-dimensional vectors \((b_1, \ldots, b_t)\) and \((r_{i1}, \ldots, r_{it})\), respectively. Two lattices are then considered:

\[
\mathcal{L}_ \perp = \{ \mathbf{x} \in \mathbb{Z}^t | \mathbf{b} \cdot \mathbf{x} \equiv 0 \} \\
\mathcal{L}_{\perp i} = \{ \mathbf{x} \in \mathbb{Z}^t | r_{i} \cdot \mathbf{x} = 0 \text{ for } 1 \leq i \leq n \}
\]

According to the definition, it holds that \( \mathcal{L}_{\perp i} \subseteq \mathcal{L}_ \perp \) and \( \mathcal{L}_{\perp i} \) contain a relatively short basis. Intuitively, OLA expects that \( \mathbf{c} \), for a finite set \( \mathbf{A} \), Cheon et al. [24].

First, we introduce some notion regarding the lattice.

**Organisation.** In Section 2, we introduce preliminary information related to the lattice and previous works. In addition, we describe how to obtain an auxiliary input from CRT-ACD instances and variant samples in Sections 3 and 3.1, respectively. Next, we provide a heuristic analysis of scaled CCK-ACD in Section 3.2. Finally, we provide an experimental result in Section 4.

2 | PRELIMINARIES

In this section, we introduce notations and some background about CRT-ACD before presenting our theorems. First, we introduce some notion regarding the lattice. Then, we define the main problem CRT-ACD and a variant, the so-called CRT-ACDwAI. Lastly, we introduce a polynomial-time analysis of CRT-ACDwAI described in Cheon et al. [24].

**Notation.** Throughout this paper, we use \( a \leftarrow A \) to denote the uniform sampling operator that selects \( a \) from \( A \) for a finite set \( A \). Moreover, when the distribution \( D \) is given, we use \( a \leftarrow D \) to denote the selection operator from \( D \).

For integers \( t \) and \( p \), we denote by \([t]_p\) the integer in \([-p/2, p/2)\) satisfying \([t]_p \equiv t \mod p \). In general, we define \( \text{CRT}(p_1, p_2, \ldots, p_n)(r_1, r_2, \ldots, r_n) \) (alternatively abbreviated as \( \text{CRT}(p)(r) \)) for distinct primes \( p_1, p_2, \ldots, p_n \) as the integer defined in \((-\frac{1}{2}\prod_{i=1}^{n} p_i, \frac{1}{2}\prod_{i=1}^{n} p_i) \) satisfying \( b_i \equiv r_i \mod p_i \) for all \( i \in \{1, 2, \ldots, n\} \).

We use bold letters to denote vectors or matrices. For any matrix \( \mathbf{A} \), we denote by \( \mathbf{A} = (a_{ij}) \) that \( (a_{ij}) \) is the \((i, j)\)-entry of \( \mathbf{A} \), transposing \( \mathbf{A} \) as \( \mathbf{A}^T \), and the \( i \)-th row vector of \( \mathbf{A} \) as \( [\mathbf{A}]_i \). Moreover, we denote by \( \text{size}(\mathbf{A}) \) the logarithm of the maximal absolute values of all entries. In addition, we define the infinite norm \( \|\mathbf{A}\|_\infty \) as \( \max_{1 \leq i \leq m} \sum_{j=1}^{n} |a_{ij}| \) with \( \mathbf{A} = (a_{ij}) \). We also denote by \( \text{diag}(a_1, \ldots, a_n) \) the diagonal matrix with diagonal coefficients \( a_1, \ldots, a_n \). Finally, we denote by \( \mathbf{A} \mod N \) the matrix whose \((i, j)\)-entry is \( [a_{ij}]_N \) for an integer \( N \).

In case of a vector \( \mathbf{v} = (v_1, \ldots, v_n) \), we compute the 2-norm \( \|\mathbf{v}\| \) and 1-norm \( \|\mathbf{v}\|_1 \) as \( \sqrt{\sum_{i=1}^{n} v_i^2} \) and \( \sum_{i=1}^{n} |v_i| \), respectively. We also use the notation \( \mathbf{v} \mod N \) regarding \( \mathbf{v} \) as the matrix.

2.1 | Lattice

A lattice generated by a set of linearly independent vectors \( \mathbf{A} = \{a_1, a_2, \ldots, a_n\} \subset \mathbb{R}^m \), denoted by \( \mathcal{L}(\mathbf{A}) \), is the set of all integer linear combinations of \( \mathbf{A} \). The elements of \( \mathbf{A} \) are called a lattice basis of \( \mathcal{L}(\mathbf{A}) \), and the rank of \( \mathcal{L}(\mathbf{A}) \) is denoted as \( n \). If \( \mathbf{A} \) is a matrix, then \( \mathcal{L}(\mathbf{A}) \) is the lattice generated by the set of all column vectors of \( \mathbf{A} \), and we call \( \mathbf{A} \) a basis matrix of \( \mathcal{L}(\mathbf{A}) \). Generally, we define the determinant of matrix \( \mathbf{A} \) as \( \sqrt{\det(\mathbf{A})} \). Additionally, the determinant of the lattice with basis matrix \( \mathbf{A} \), denoted by \( \det \mathcal{L}(\mathbf{A}) \), is defined as the determinant of the basis matrix \( \det \mathbf{A} \).

Let \( \mathcal{L} \) be a lattice of rank \( n \). Then, the successive minima \( \lambda_1, \ldots, \lambda_n \in \mathbb{R}_+^+ \) of \( \mathcal{L} \) are defined as follows. For any \( 1 \leq i \leq n \), \( \lambda_i \) is a minimum value such that there exist \( i \) linearly independent vectors \( \mathbf{L} \in \mathcal{L} \) whose sizes do not exceed \( \lambda_i \). We use \( \lambda_i(\mathcal{L}) \) to denote the \( i \)-th successive minimum of the lattice \( \mathcal{L} \). In relation to the successive minima, there is a useful result to restrict them, which is called Minkowski’s Theorem [25].

**Theorem (Minkowski)** Let \( \mathcal{L} \subset \mathbb{R}^m \) be a \( n \)-rank lattice. Then, the following is valid:

\[
\lambda_1(\mathcal{L}) \leq \left( \prod_{i=1}^{n} \lambda_i(\mathcal{L}) \right)^\frac{1}{n} \leq \sqrt{n} \cdot \left( \det \mathcal{L} \right)^\frac{1}{n}.
\]

We also obtain an upper bound for \( \lambda_2(\mathcal{L}) \) from the above inequalities as follows:

\[
\left( \lambda_2(\mathcal{L}) \right)^\frac{1}{n-1} \leq \left( \prod_{i=2}^{n} \lambda_i(\mathcal{L}) \right)^\frac{1}{n-2} \leq \sqrt{n} \cdot \left( \det \mathcal{L} \right)^\frac{1}{n-2}.
\]

i.e.,

\[
\frac{\sqrt{n}}{\lambda_2(\mathcal{L})} \leq \left( \det \mathcal{L} \right)^\frac{1}{n-2} \cdot \lambda_1(\mathcal{L})^\frac{1}{n-2}.
\]

Finding a short vector of a lattice is essential in our attack. Fortunately, there are some algorithms to find a short vector of a lattice, called lattice reduction algorithms.

**Lattice Reduction Algorithm** The LLL algorithm and the BKZ algorithm, introduced in [25, 26], are lattice reduction algorithms. We mainly use these algorithms to find an approximately short vector of a lattice in finite time. According to [25], the LLL algorithm on the \( n \)-rank lattice \( \mathcal{L} \) with basis matrix \( \mathbf{B} \) gives a short vector \( \mathbf{v} \), which satisfies the following:

\[
\|\mathbf{v}\| \leq \min \left\{ 2^{\frac{2n}{m}} \cdot \left( \det \mathcal{L} \right)^\frac{1}{m}, 2^n \cdot \lambda_1(\mathcal{L}) \right\}.
\]
We denote the time complexity of the LLL algorithm by $T_L(n, \text{size}(B))$, which is a polynomial function on inputs.

For the BKZ algorithm, according to [26], the block size $\beta$ determines how short the output is. Applying the BKZ algorithm to the $n$-rank lattice $L$ with basis $B$, we can obtain a short vector $v$ in $\text{poly}(n, \text{size}(B)) \cdot C_{HKZ}(\beta)$ time, satisfying the following:

$$
\|v\| \leq \left\{ 2 \cdot \gamma_{\beta} \frac{n^2}{3} \cdot (\det L)^{1/2}, 4 \cdot \gamma_{\beta} \frac{n^2}{3} \cdot (\lambda_1(L)) \right\}
$$

where $\gamma_\beta$ is the Hermite constant of rank $\beta$, which does not exceed $\beta$, and $C_{HKZ}(\beta)$ denotes the time spent to obtain the shortest vector of the $\beta$-dimensional lattice. Because it takes $2^{O(\beta)}$ time to find it, we regard $C_{HKZ}(\beta)$ as $2^{O(\beta)}$ henceforth.

### 2.2 Cryptanalysis of the CRT-ACD with auxiliary input

The PACD, introduced by Howgrave-Graham [27], is a problem of finding a secret prime $p$ for a given many instances which are nearly multiples of $p$. With the Chinese remainder theorem (CRT) on multiple primes, we define a multiple prime version of PACD, called a CRT-ACD problem. We present a formal definition of CRT-ACD problems as follows.

**Definition 1 (CRT-ACD)** Let $n$, $\eta$, $q$ be positive integers and $\chi_\eta$ be a uniform distribution over $\mathbb{Z} \cap (-2^n, 2^n)$. For given $\eta$-bit primes $p_1$, ..., $p_n$, the sampleable CRT-ACD distribution $D_{X,\eta,n}(p_1, \ldots, p_n)$ is defined as

$$
D_{X,\eta,n}(p_1, \ldots, p_n) = \{ \text{CRT}(p_1)(r_1) \mid r_1 \sim \chi_\eta \}.
$$

The CRT-ACD problem is given as follows: for a given number of samples of $D_{X,\eta,n}(p_1, \ldots, p_n)$ with $N = \prod_{i=1}^{n} p_i$, find all $p_i$.

While the CRT-ACD problems are regarded to be too hard for proper parameters, Cheon et al. described an analysis of the CRT-ACD problem in polynomial time of $n$, $\eta$, $\rho$ when an auxiliary input CRT$_{ACD}$(N/p) is given. [24] In this Section, we define a CRT-ACD problem with an auxiliary input (CRT-ACDwAI) by importing Definition 1 and introduce a result of Cheon et al.’s research.

**Definition 2 (CRT-ACDwAI)** Let $n$, $\eta$, $q$ be positive integers. For given $\eta$-bit primes $p_1$, ..., $p_n$, define $N = \prod_{i=1}^{n} p_i$ and $\hat{p}_i = N/p_i$, for $1 \leq i \leq n$.

The CRT-ACDwAI problem is given as follows: for given many samples from $D_{X,\eta,n}(p_1, \ldots, p_n)$, $N$ and $\hat{p}_i$ is CRT$_{(p_i)}(\hat{p}_i)$, find $p_i$ for all $i$.

We state an useful lemma and a result described in [24] below.

**Lemma 3** ([22], Section 3.1) For a given $\hat{P} = \text{CRT}_{(p)}(\hat{p}_1)$ and $a = \text{CRT}_{(p)}(r_1)$, it satisfies

$$
[a \cdot \hat{P}]_N = \text{CRT}_{(p)}(r_1 \cdot \hat{p}_1) = \sum_{i=1}^{n} r_i \cdot \hat{p}_i,
$$

if $n < \eta - \rho - \log n - 1$.

**Proof:** The first equality is evident due to the definition of the Chinese remainder theorem. To show that the second equality is correct, consider the equation modulo $p_i$ for each $i$. Then the left-hand side is $r_i \cdot \hat{p}_i$ and the right-hand side is also $r_i \cdot \hat{p}_i$, because $\hat{p}_i \equiv 0 \mod p_i$ holds for $j \neq i$. Finally, $\sum_{i=1}^{n} r_i \cdot \hat{p}_i \leq \sum_{i=1}^{n} |r_i| \cdot \hat{p}_i \leq n \cdot 2^\eta \cdot 2^{(n-1)\eta}$, which is less than $N/2$. Hence, by the uniqueness of CRT, the second equality holds.

Lemma 3 implies that the product of a CRT-ACD instance and an auxiliary input is small compared with the integer $N$. Simultaneously, the product is an integer equation between secret elements.

**Theorem 4** ([22], Section 3.2) Let $\chi_\rho$ be the uniform distribution over $(-2^n, 2^n) \cap \mathbb{Z}$. Given O(n) CRT-ACD samples from $D_{X,\eta,n}(p_1, \ldots, p_n)$ with $N = \prod_{i=1}^{n} p_i$ and $\hat{P} = \text{CRT}_{(p)}(\hat{p}_1)$, one can recover every secret prime $p_1$, ..., $p_n$ in time $O(n^{2+\omega} \cdot \eta)$ with $\omega \leq 2.38$ and an overwhelming probability of $\rho$.

**Remark:** In general, even if the auxiliary input is $\hat{P} = \text{CRT}_{(p)}(a_i \cdot \hat{p}_i)$ such that $|a_i| \cdot \hat{p}_i \leq 2^{n-\eta-2\rho \cdot \log n - 1}$, Cheon et al.’s algorithm still can be applied. Therefore, we accordingly now generally regard the auxiliary input $\hat{P}$ in CRT-ACDwAI as the following:

$$
\hat{P} = \text{CRT}_{(p)}(a_i \cdot \hat{p}_i) \text{ with } |a_i| \leq 2^{n-\eta-2\rho \cdot \log n - 1}/\hat{p}_i
$$

### 3 Extension of the SDA Algorithm for ACD Variants

In this Section, we introduce an approach to solve the CRT-ACD problem by using the SDA algorithm. To achieve our goal, we recover two auxiliary inputs of the CRT-ACDwAI problem from the CRT-ACD instances. Then, by applying the result of Section 2.2, CRT-ACD is solved.
We note here that any elements in \( \mathbb{Z}_N \) can be written as 
\[
A = \sum_{j=1}^{n} a_j \cdot \hat{p}_j
\]
for some integers \( a_j \), because all \( \hat{p}_j = N/p_j \)'s are relative primes. Multiplying by the given CRT-ACD instances \( b_i = \text{CRT}(p_j)(r_{j,i}) \), we obtain elements of the form \( \sum_{j=1}^{n} r_{j,i} \cdot a_j \cdot \hat{p}_j \). The main idea to recover an auxiliary input is that if \( A \) is an auxiliary input, by Lemma 3, \( \langle A \cdot b_i \rangle_N \) is equal to \( \sum_{j=1}^{n} r_{j,i} \cdot a_j \cdot \hat{p}_j \); Compared with \( N \), it is relatively small for each \( i \).

To exploit this observation, we now consider a column lattice \( \mathcal{L} \) generated by the following basis matrix.

\[
B = \begin{pmatrix}
1 & 0 & 0 & \cdots & 0 \\
b_1 & N & 0 & \cdots & 0 \\
b_2 & 0 & N & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
b_k & 0 & 0 & \cdots & N
\end{pmatrix}.
\]

Our main observation is then that \( \langle A, \langle A \cdot b_1 \rangle_N, \ldots, \langle A \cdot b_k \rangle_N \rangle^T \) is a short vector of \( \mathcal{L} \) with an auxiliary input \( A \). Hence, by finding a short vector of the lattice \( \mathcal{L} \), one may recover an auxiliary input. More precisely, we are able to obtain the following result.

**Theorem 5** Let \( n, \eta, \rho \) be parameters of CRT-ACD. When \( k = O(n) \) CRT-ACD instances are given, CRT-ACD can be reduced to CRT-ACD with \( \eta \) under the condition \( 2n \leq \eta - 5\rho - 4 \log n - 3 \) in \( T_L(n, n \cdot \eta) \) times with the LLL algorithm. In other words, we can find an auxiliary input for given CRT-ACD instances.

**Proof:** Assuming here that \( k \) is greater than \( n \), let \( c = (c_0, c_1, \ldots, c_k)^T \) be a short vector in \( \mathcal{L} \). Then, for some integer \( d = \sum_{j=1}^{n} a_j \cdot \hat{p}_j \), \( c \) can be written as

\[
c = (\langle d \rangle_N, \langle d \cdot b_1 \rangle_N, \ldots, \langle d \cdot b_k \rangle_N)^T.
\]

Except for the first entry of \( c \), we define a vector \( \tilde{c} = (c_1, \ldots, c_k) \). Then, for all \( i \), we have \( c_i \equiv \sum_{j=1}^{n} r_{j,i} \cdot a_j \cdot \hat{p}_j \mod N \) with \( b_i = \text{CRT}(p_j)(r_{j,i}) \). Therefore, the vector \( \tilde{c} \) is decomposed as follows:

\[
\tilde{c} = a \cdot \hat{P} \cdot \mathbf{R} \mod N,
\]

where \( a = (a_1, \ldots, a_n) \), \( \hat{P} = \text{diag}(\hat{p}_1, \ldots, \hat{p}_n) \), and \( \mathbf{R} = (r_{j,i}) \in M_{n \times k}(\mathbb{Z}) \).

For a matrix \( \mathbf{R} \) and \( k \geq n \), there exists a right inverse \( \mathbf{R}^* \in M_{k \times n}(\mathbb{Z}) \) such that the following holds with overwhelming probability.

\[
\mathbf{R} \cdot \mathbf{R}^* = \mathbf{I}_n
\]

and

\[
[a \cdot \hat{P} \cdot R \cdot R^*]_N = ([a_1 \cdot \hat{p}_1]_N, \ldots, [a_n \cdot \hat{p}_n]_N),
\]

where \( \mathbf{I}_n \) is an \( n \times n \) identity matrix. Thus, we can restrict the size of \( [a \cdot \hat{p}_j]_N \)'s as follows.

\[
\|a \cdot \hat{P} \cdot \mathbf{R} \cdot \mathbf{R}^* \|_\infty \leq \| \tilde{c} \cdot \mathbf{R}^* \|_\infty \leq \| \tilde{c} \| \cdot \| \mathbf{R}^* \|_\infty.
\]

Our goal is to make the size of \( \| \tilde{c} \| \cdot \| \mathbf{R}^* \|_\infty \) less than \( 2^{\eta n - 2\rho \log n - 1} \), which implies the following:

\[
\|a \cdot \hat{P} \cdot \mathbf{R} \cdot \mathbf{R}^* \|_\infty \leq 2^{\eta n - 2\rho \log n - 1},
\]

i.e., \( \|a_j \cdot \hat{p}_j\|_N = \|a_j\|_p \cdot \hat{p}_j \leq 2^{\eta n - 2\rho \log n - 1} \) for each \( j \).

The above condition establishes that \( \|d\|_N = \sum_{j=1}^{n} a_j \cdot \hat{p}_j \|_N = \sum_{j=1}^{n} [a_j \cdot \hat{p}_j]_N \) holds, and would be the auxiliary input for the CRT-ACD. Therefore, \( c_0 = [d]_N \) is exactly \( \sum_{j=1}^{n} [a_j \cdot \hat{p}_j]_N \cdot \hat{p}_j \) with \( \|a_j\|_p \leq 2^{\eta n - 2\rho \log n - 1} \) for each \( j \).

We next define a lattice \( \mathcal{L} = \{ x_1 : \hat{R}, \cdot x_1 = 0 \} \). Its rank is \( k = (n - 1) \) and its determinant is less than \( (\sqrt{k} \cdot 2^\rho)^{(n-1)} \), by Hadamard's inequality. Then, by Gaussian Heuristics, there exist two vectors \( w_i, w'_i \in \mathcal{L} \) such that \( \|w_i\|_\infty, \|w'_i\|_\infty \leq (\sqrt{k} \cdot 2^\rho)^{(n-1)/(k-n+1)} \).

We now consider two integers \( (R_i, w_i) \) and \( (R_j, w'_j) \). We expect that the two integers are relatively prime because \( \mathcal{L} \) are independent of the vector \( R_i \). Because the size of each entry of \( R_i \) is less than \( 2^\rho \), it is evident that both sizes have an upper bound of \( n \cdot (\sqrt{k} \cdot 2^\rho)^{(n-1)/(k-n+1)} \cdot 2^\rho \). This size bound in particular implies that there exist two integers \( f_i \) and \( f'_i \) such that

\[
J_i \cdot \langle R_i, w_i \rangle + J'_i \cdot \langle R_i, w'_i \rangle = 1
\]

\[
\|J_i\|, \|J'_i\| \leq n \cdot (\sqrt{k} \cdot 2^\rho)^{(n-1)/(k-n+1)} \cdot 2^\rho.
\]

We now define \( v_i \) as \( f_i \cdot w_i + f'_i \cdot w'_i \). By the linear homomorphic property, the equation \( R \cdot v_i = \epsilon_i \) holds. It implies that there exist \( R_i^* \) satisfying \( \|R_i^*\|_\infty \leq n \cdot (\sqrt{k} \cdot 2^\rho)^{(2n-1)/(k-n+1)} \cdot 2^\rho \).

**Size of \( \|c\|_\infty \):** Because \( \langle \hat{p}_1, r_{1,1} \hat{p}_1, \ldots, r_{1,k} \hat{p}_1 \rangle^T \) is in \( \mathcal{L} \), the size of \( \lambda_1(\mathcal{L}) \) does not exceed the previous vector's size. Therefore, we at least establish that the following equation holds.

\[
\lambda_1(\mathcal{L}) \leq \sqrt{k} \cdot 2^{\rho(n-1)}
\]
Taking $c$ as the shortest vector of the LLL algorithm output on $L$, we can bound $\|c\|$, as described in Section 2.1. In other words, the size of vector $c$ is less then $2^{k/2} \cdot \lambda_1(L) \leq 2^{k/2} \cdot \sqrt{k} \cdot 2^{p(n-1)} \cdot n$.

Thus, the upper bound of $\|c\| \cdot \|R\|_\infty$ is computed by the following inequality:

$$\|c\| \cdot \|R\|_\infty \leq n \cdot (\sqrt{k} \cdot 2^p)^{(n-2)/2(n-1)} \cdot 2^p \cdot \|c\|\|.$$  

We note that the exponential term $k/2 + 2p(n-1)/(k-n+1)$ can be optimised as $(n-1)/2 + \sqrt{4p(n-1)}$ for a certain $k$ and is dominated by $(n-1)/2 + \sqrt{4p(n-1)}$. Thus, we can assume that $k = 2n$, the term is still bounded by $(n-1)/2 + \sqrt{4p(n-1)}$. Thus, considering optimised $k$ does not have a crucial impact. Therefore, for simplicity, we use $k = 2n$ so that the upper bound can be written as $4n^3 \cdot 2^{n-1(n)+n+3p}$.

To guarantee our goal, the following inequality is required:

$$\|c\| \cdot \|R\|_\infty \leq 4n^3 \cdot 2^{(n-1)n+n+3p} \cdot 2^{n-1-n+n+3p} \cdot 2^{n-1-n} = 2^{n-1-n+n+3p}.$$  

Then, this condition can be written as follows:

$$2n \leq n - 5p - 4 \log n - 3.$$  

Therefore, when this condition holds, we can regard $c_0$ as an auxiliary input of CRT-ACD. This completes the proof.

Remark: Formally, our algorithm cannot reduce the CRT-ACD to CRT-ACDwAI since we employ ‘two’ auxiliary inputs even though the original CRT-ACDwAI problem only provides a single auxiliary input. However, our algorithm is almost the same as the algorithm for solving CRT-ACDwAI except that our algorithm requires ‘two’ auxiliary inputs. Thus, in this paper, we simply say that CRT-ACD can be reduced to CRT-ACDwAI.

Remark: In the event that the BKZ algorithm with block size $\beta$ is exploited instead of the LLL algorithm, we can obtain a lattice point $c$ such that $\|c\| \leq 4 \cdot \beta^{2.5} \cdot \lambda_1$ in poly($n, \eta$) times. We then restrict $\|c\| \cdot \|R\|_\infty$ up to $4n^3 \cdot 2^{(n-1)n+n+3p} \cdot 4 \cdot \beta^{2.5}$. As a result, if it holds that

$$4n^3 \cdot 2^{(n-1)n+n+3p} \cdot 4 \cdot \beta^{2.5} \leq 2^{n-1-n+n+3p} \cdot 2^{n-1-n},$$

We can ensure that $c_0$ is an auxiliary input of CRT-ACD. This condition can be written as follows:

$$n \cdot \left(2 \log \beta \cdot \beta^{2.5} + 1\right) \leq n - 5p - 4 \log n - 3 \log \beta - 5.$$  

Now, we introduce our algorithm to solve a CRT-ACD problem using two auxiliary inputs. This process is almost the same to the algorithm in Cheon et al.’s algorithm [22]. First, by applying the above algorithm twice for different CRT-ACD instances, we assume that two auxiliary inputs $d = \sum_{i,j=1}^n a_{ij} \cdot \hat{p}_j$ and $d' = \sum_{i,j=1}^n a'_{ij} \cdot \hat{p}_j$ are given. By Lemma 3, the following equations hold:

$$[d \cdot b_u \cdot c_v]_N = \sum_{j=1}^{n} r_{j,u} \cdot (a_j \cdot \hat{p}_j) \cdot r_{j,v},$$

$$[d' \cdot b_u \cdot c_v]_N = \sum_{j=1}^{n} r_{j,u} \cdot (a'_j \cdot \hat{p}_j) \cdot r_{j,v},$$

where $b_i$'s are CRT-ACD instances. For $n$ CRT-ACD samples $\{b_i = \text{CRT}_{p_i}(r_{ij})\}_{i \in \mathbb{Z}^n}$, we denote by $\omega_{ij}$ and $\omega'_{ij}$ the expression $[b_i \cdot b_j \cdot d]_N$ and $[b_i \cdot b_j \cdot d']_N$, respectively. Then, we obtain the following matrix equations:

$$\omega_{ij} = \sum_{k=1}^{n} r_{kj} \cdot (a_k \cdot \hat{p}_k) \cdot r_{kj}$$  

$$= R_i \cdot \left( \begin{array}{c} a_1 \cdot \hat{p}_1 \\ 0 \end{array} \right) \cdot \left( \begin{array}{cc} R_i & 0 \\ 0 & \ldots \\ \vdots & \ddots & \ddots \\ 0 & 0 & \ldots & a_n \cdot \hat{p}_n \end{array} \right) \cdot R_i^T,$$

$$\omega'_{ij} = \sum_{k=1}^{n} r_{kj} \cdot (a'_k \cdot \hat{p}_k) \cdot r_{kj}$$  

$$= R_i \cdot \left( \begin{array}{c} a'_1 \cdot \hat{p}_1 \\ 0 \end{array} \right) \cdot \left( \begin{array}{cc} R_i & 0 \\ 0 & \ldots \\ \vdots & \ddots & \ddots \\ 0 & 0 & \ldots & a'_n \cdot \hat{p}_n \end{array} \right) \cdot R_i^T,$$

where the vectors $R_i$ are of the form $(r_{ij}, r_{i,j+1}, \ldots, r_{n,i})$.

By spanning $1 \leq i, j \leq n$, we can construct two matrices $W = (\omega_{ij})$ and $W' = (\omega'_{ij}) \in M_{n \times n}(\mathbb{Z})$. By computing all eigenvalues of a matrix $Y = W \cdot (W')^{-1} \in M_{n \times n}(\mathbb{Q})$, we can recover $a_i/a'_i$ for each $j$ in polynomial time $\eta$ and $n$. More precisely, the overall complexity for computing eigenvalues is $O(n^{2+\omega} \cdot \eta)$, where $\omega$ is a constant less than 2.38. Once we obtain the ratio, we can also get $a_i/g_i$ and $a'_i/g_i$, where $g_i$ is the greatest common divisor of $a_i$ and $a'_i$. Contrastingly, from the setting of $d$ and $d'$, we know that for all $j$,

$$d/d' \equiv a_i/a'_i \equiv (a_i/c)/(a'_i/c) \mod p_j.$$
This implies that
\[ d \cdot \alpha_j / c - d' \cdot \alpha_j' / c \equiv 0 \mod p_j. \]

Thus, by computing \( \gcd(N, d \cdot \alpha_j / g_j - d' \cdot \alpha_j' / g_j) \), we can find \( p_j \) for each \( j \).

We note that if there is an index \( k \) such that \( \alpha_j / \alpha_j' = \alpha_k / \alpha_k' \), then computing \( \gcd(N, d \cdot \alpha_j / g_j - d' \cdot \alpha_j' / g_j) \) outputs a multiple of \( p_j \cdot p_k \). Because \( d \) and \( d' \) are distinct integers, all eigenvalues cannot be the same. This means that if \( n = 2 \), computing \( \gcd(N, d \cdot \alpha_j / g_j - d' \cdot \alpha_j' / g_j) \) only outputs \( p_j \). In the general case, we can therefore obtain at least a non-trivial factor of \( N \). This allows us to reduce the CRT-ACD into the CRT-ACD with a small number of factors compared with \( n \). In other words, by repeating the above algorithm on the reduced problem, we can obtain all prime factors of \( N \). In summary, we have the following results.

**Corollary 6** Let \( n, \eta, \rho \) be parameters of CRT-ACD. When \( O(n) \) CRT-ACD instances are given, we can solve CRT-ACD under the condition \( 2n \leq \eta - 5\rho - 4 \log n - 3 \) in \( \max\{T_L(n, n \cdot \eta), O(n^{2+\omega} \cdot \eta)\} \) times with the LLL algorithm, where \( \omega \) is a constant less than 2.38.

### 3.1 Scaled CRT-ACD problem

As an extension of the previous result, we introduce a variant of the CRT-ACD problem and its analysis. First, we provide a precise definition of the scaled CRT-ACD problem (SCRT-ACD).

**Definition 7 (scaled CRT-ACD)** Let \( n, \eta, \rho \) be positive integers. For given \( \eta \)-bit primes \( p_1, \ldots, p_n \), \( k + 1 \) numbers of modified CRT-ACD instances for \( 0 \leq i \leq k \) are given in the following form:

\[
c \cdot b_i \text{ with } b_i = \text{CRT}_{(p_i)}(r_{j,i}) \leftarrow D_{x,n,n}(p_1, \ldots, p_n),
\]

where \( c \leftarrow \mathbb{Z}_N \). The scaled CRT-ACD problem is defined as follows: given such modified samples of CRT-ACD and \( N = \prod_{i=1}^{k} p_j \), find \( p_j \) for all \( i \).

Because the size of \( c \) is unknown, the algorithm described in Section 3 is not directly applicable to the given modified instances. Before applying the previous algorithm, we compute a ratio between scaled CRT-ACD instances. In other words, we obtain the new quantities

\[
b'_i = [(c \cdot b_i) \cdot (c \cdot b_0)^{-1}]_N.
\]

In this case, \( b'_i \equiv b_i \cdot b_0^{-1} \equiv r_{j,i} \cdot r_{j,0}^{-1} \mod p_j \) holds for each \( j \).

For the new samples \( b'_i \), we consider a new auxiliary input \( d \) of the form \( \sum_{j=1}^{n} \alpha_j \cdot r_{j,0}^{-1} \cdot \tilde{p}_j \). Suppose the size of \( r_{j,i} \) and \( \alpha_j \) is sufficiently small. Then, with the similar argument in the proof of Lemma 3, the following holds.

\[
[d \cdot b'_u \cdot b'_v]_N = \sum_{j=1}^{n} \alpha_j \cdot r_{j,u} \cdot r_{j,v} \cdot \tilde{p}_j,
\]

\[
[d \cdot b^2]_N = \sum_{j=1}^{n} \alpha_j \cdot r_{j,i}^2 \cdot \tilde{p}_j \ll N.
\]

More precisely, the above equation can hold under the condition

\[
[\alpha_j \cdot \tilde{p}_j]_N \leq [\alpha_j \cdot \tilde{p}_j]_N \leq 2^{2\eta - 2\rho \log n - 1} \text{ for each } j.
\]

From the observation, we consider a new lattice \( \mathcal{L}' \) generated by the following matrix.

\[
B' = \begin{pmatrix}
1 & 0 & 0 & \cdots & 0 \\
0 & \ddots & 0 & \cdots & 0 \\
0 & \cdots & \ddots & \cdots & 0 \\
0 & 0 & \cdots & \ddots & \cdots \\
\end{pmatrix}.
\]

Similar to the analysis in the Section 3, the first entry of the short vector of \( \mathcal{L}' \) becomes a new auxiliary input. As mentioned in Section 3, the process of optimising the number of samples \( k \) did not improve the results significantly. Therefore, here, for the convenience of computation, \( k \) is fixed to \( 2n \).

Let \( \mathbf{c}' = ([d]_N; [d \cdot b'^2_1]_N, \ldots, [d \cdot b'^2_k]_N)^T \) be the shortest output vector of the LLL algorithm on \( \mathcal{L}' \) with \( d = \sum_{j=1}^{n} \alpha_j \cdot r_{j,0}^{-2} \cdot \tilde{p}_j \) and let \( \mathbf{c} = ([d \cdot b^2]_N, \ldots, [d \cdot b^2_k]_N) \). Then, we can write the following:

\[
\mathbf{c}' \equiv \mathbf{a} \cdot \mathbf{P} \cdot \mathbf{R}' \mod N,
\]

where \( \mathbf{a} = (\alpha_1, \ldots, \alpha_n) \), \( \mathbf{P} = \text{diag}(\tilde{p}_1, \ldots, \tilde{p}_n) \), and \( \mathbf{R}' = (r_{j,i}^2) \in \mathbb{M}_{k \times n}(\mathbb{Z}) \).

Similar to Section 3, there exists a right inverse \( (\mathbf{R}')^* \in \mathbb{M}_{k \times n}(\mathbb{Z}) \) satisfying \( \mathbf{R}' \cdot (\mathbf{R}')^* = \mathbf{I}_n \). Then, the following holds.
\[ \| a \cdot \hat{P} \cdot R \cdot (R')^* \text{ mod } N \|_\infty = \| c' \cdot (R')^* \text{ mod } N \|_\infty, \]
\[ \leq \| c' \cdot (R')^* \|_\infty \leq \| c' \|_\infty \cdot \| (R')^* \|_\infty, \]
\[ \leq \| c' \| \cdot \| (R')^* \|_\infty. \]

In short, if the size of \( \| c' \| \cdot \| (R')^* \|_\infty \) is bounded by \( 2^{m_1-n-2p-\log n-1} \), the first entry \( d \) is a new auxiliary input.

For each entry of \( (r_j, i) \in R', |r_j|^2 \leq 2^{p} \) holds. Therefore, we can set the matrix \( (R')^* \) satisfying \( \| (R')^* \|_\infty \leq 2^{n^2} \cdot 2^{6p} \) for the same reason as in Section 3.

Conversely, we already know that \( (r_{i_0}, r_{i_1}, r_{i_2}, \ldots, r_{i_{2n}})^T \in L' \). Therefore, this implies that an equation \( \lambda_r(L') \leq \sqrt{2n} + 1 \cdot 2^{(n-1)\eta+5p} \) holds. As mentioned above, a vector \( c' \) is the shortest output vector of the LLL algorithm. Therefore, the size of \( c' \) is bounded by:

\[ \| c' \| \leq 2^n \cdot 2n \cdot 2^{(n-1)\eta+2p}. \]

Combining the above inequalities, we can obtain the following.

\[ \| a \cdot \hat{P} \cdot R \cdot (R')^* \text{ mod } N \|_\infty \leq \| c' \| \cdot \| (R')^* \|_\infty, \]
\[ \leq 4n^3 \cdot 2^{(n-1)\eta+n+8p} \leq 2^{m-n-2p-\log n-1}. \]

Therefore, the condition to find the new auxiliary input can be concisely written as follows:

\[ 2n \leq \eta - 10p - 4 \log n - 3. \]

Hence, we have the following result.

**Lemma 8** Let \( n, \eta, \rho \) be parameters of a scaled CRT-ACD. When \( O(n) \) scaled CRT-ACD instances are given, we can find an auxiliary input for instances of SCRT-ACD under the asymptotic condition \( 2n \leq \eta - 10p - 4 \log n - 3 \) in \( T_l(n, n \cdot \eta) \) time by using the LLL algorithm.

Suppose we now have two auxiliary inputs \( d = \sum_{j=1}^n \alpha_j \cdot r_j \cdot \hat{P}_j \) and \( d' = \sum_{j=1}^n \alpha'_j \cdot r_j \cdot \hat{P}_j \). We can check the following:

\[ [d \cdot b_j \cdot b_j']_N = \sum_{j=1}^n r_{j, a_j} \cdot (\alpha_j \cdot \hat{P}_j) \cdot r_{j, x}. \]
\[ [d' \cdot b_j \cdot b_j']_N = \sum_{j=1}^n r_{j, a_j'} \cdot (\alpha_j' \cdot \hat{P}_j) \cdot r_{j, x}. \]

With the same algorithm in Section 3, we can compute \( \alpha_j \) and \( \alpha_j' \) for all \( j \) in polynomial time of \( n, \eta, \) and \( \rho \). From the relation \( d \cdot (d')^{-1} \equiv \alpha_j \cdot (\alpha_j')^{-1} \text{ mod } \rho_j \), we can find \( \rho_j \) for each \( j \) by computing \( \gcd(N, d \cdot \alpha_j' - d' \cdot \alpha_j) \) and obtain the following theorem.

**Theorem 9** Let \( n, \eta, \rho \) be parameters of a scaled CRT-ACD problem. If the asymptotic condition holds for parameters \( 2n \leq \eta - 10p - 4 \log n - 3 \), we can solve the scaled CRT-ACD in \( \max \{ T_l(n, n \cdot \eta), O(n^{2+\omega}) \} \) time by using the LLL algorithm, where \( \omega \) is a constant less than 2.38.

**Remark:** If we use the BKZ algorithm with block size \( \beta \) instead of the LLL algorithm for the scaled CRT-ACD, the upper bound of a vector \( c' \) from the lattice \( L' \) is as follows:

\[ \| c' \| \leq 4 \cdot 2^{(n+3)} \cdot 2n \cdot 2^{(n-1)\eta+2p}. \]

Therefore, the required condition changes to the following:

\[ n \cdot \left( \frac{2 \log \beta}{\beta - 1} + 1 \right) \leq \eta - 10p - 4 \log n - 3 \log \beta - 5. \]

### 3.2 Scaled CCK-ACD problem

In the previous Section, we gave an algorithm to solve the scaled CRT-ACD. Similar to the scaled CRT-ACD, we can define a scaled CCK-ACD by multiplying an unknown constant by CCKACD instances. In this Section, we provide an algorithm to solve the scaled CCK-ACD. Before extending a method, we give a definition of the scaled CCK-ACD problem.

**Definition 10 (scaled CCK-ACD)** Let \( n, \eta, \theta, \eta' \) be positive integers, \( \gamma = n \cdot \eta + \eta' \). For given \( \eta' \)-bit primes \( p_0, \eta' \)-bit primes \( p_1, \ldots, p_n \), \( N \) is an integer of \( p_0 \cdot \prod_{i=1}^n p_i \). The sampleable distribution \( D_{x,n} \) is defined as

\[ \{ \text{CRT}(p) | r_0 \leftarrow X_N, r_i \leftarrow \chi_{p_i} \text{ for } i \in [n] \}. \]

Then, scaled CCK-ACD instances for \( 0 \leq i \leq k \) are defined to take the following form:

\[ [c \cdot b_i]_N \text{ with } b_i = \text{CRT}(p) | r_0 \leftarrow D_{x,n} \cdot (p_0, \ldots, p_n), \]

where \( c \leftarrow Z_N \). The scaled CCK-ACD problem is given as follows: given \( k \) scaled CCK-ACD samples and \( N \), find \( p_i \) for all \( i \).

Intuitively, except for the factor \( p_0 \), scaled CCK-ACD coincides with scaled CRT-ACD and \( [b_i]_{p_0} \) has an arbitrary large value in \( Z_{p_0} \).
Because the size of $c$ is unknown, the algorithm in [21] is not directly applicable to the given modified samples. Thus, we consider an analogous technique in Section 3.1. In other words, we use division with modulus $N$ and obtain the new instances

$$b'_i = [(c \cdot b_i) \cdot (c \cdot b_0)^{-1}]_N.$$  

It is evident that $b'_i \equiv b_i \cdot b_0^{-1} \equiv r_{ji} \cdot r_{j0}^{-1} \mod p_j$ holds for all $0 \leq j \leq n$.

To apply the method similarly to the algorithm in [21], we define a new auxiliary input $d$ of the form $\sum_{j=1}^n \alpha_j \cdot r_{ji} \cdot \hat{p}_j$ for scaled CCK-ACD. We here note that the auxiliary input has no term of $\hat{p}_0$. Similar to the analysis in Section 3.1, if the size of $r_{ji}$ and $\alpha_j$ is sufficiently small, we can guarantee the following holds:

$$[d \cdot b'_i^2]_N = \sum_{j=1}^n \alpha_j \cdot r_{ji}^2 \cdot \hat{p}_j.$$  

Compared with scaled CRT-ACD, the auxiliary input has no term of $\hat{p}_0$. Thus, the new auxiliary input is at least a multiple of $p_0$. This implies that $p_0$ can be recovered by computing the greatest common divisor of the auxiliary input and a modulus $N$. When we recover an integer $N/p_0$, scaled CCK-ACD can evidently be reduced to scaled CRT-ACD. Therefore, we can apply the algorithm in Section 3.1. In short, our main goal is to recover the auxiliary input to recover $p_0$.

Now we consider a lattice $L''$ generated by:

$$B'' = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ b_1^2 & N & 0 & \cdots & 0 \\ b_2^2 & 0 & N & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_k^2 & 0 & 0 & \cdots & N \end{pmatrix}.$$  

Because the vector

$$\left( \sum_{j=1}^n r_{j0} \cdot \hat{p}_j, \sum_{j=1}^n r_{j1}^2 \cdot \hat{p}_j, \ldots, \sum_{j=1}^n r_{jk}^2 \cdot \hat{p}_j \right)$$

is in a lattice $L$, $\lambda_1$ is smaller than $2^{n(n-1)\eta+\eta'+2\rho}$. However, we assume that $\lambda_2$ comes from the Gaussian heuristic assumption, i.e. $\lambda_2 \approx \sqrt{k+1} \cdot N^{\frac{1}{2n}}$. Hence, we can expect that the first component of the shortest output vector $e$ produced by the LLL algorithm on the above matrix $B$ is a multiple of $p_1$ as long as $2^{\frac{k+1}{2}} \lambda_1(L)$ is less than $\lambda_2(L)$. Therefore, we have the following inequalities.

$$2^{\frac{k+1}{2}} \cdot 2^{(n-1)\eta+\eta'+2\rho} \leq \sqrt{k+1} \cdot N^{\frac{1}{2n}}.$$  

Taking the logarithm of both sides of the inequality, it can be asymptotically rearranged as

$$\frac{k+1}{2} + \frac{1}{k+1} \cdot \gamma \leq \eta - 2\rho + \log k$$

$$\Rightarrow \sqrt{2} \gamma \leq \eta - 2\rho + \log \sqrt{2} \gamma.$$  

The second equality comes from the arithmetic-geometric mean inequality with $(k + 1)^2 = 2\gamma$. Hence, the following result holds.

**Theorem 11** Let $n$, $\eta$, $\eta'$, $\rho$ be parameters of scaled CCK-ACD. When $O(n)$ scaled CCK-ACD instances are given, scaled CCK-ACD can be reduced to scaled CRT-ACD under the asymptotic condition $\sqrt{2} \gamma \leq \eta - 2\rho + \log \sqrt{2} \gamma$, in $T_L(n, \gamma)$ times by using the LLL algorithm and heuristic assumption.

**Corollary 12** Let $n$, $\eta$, $\rho$, $\gamma$ be parameters of a scaled CCK-ACD problem. If the asymptotic condition on parameters $2n \leq \eta - 10\rho - 4 \log n - 3$ and $\sqrt{2} \gamma \leq \eta - 2\rho + \log \sqrt{2} \gamma$ holds, we can solve the scaled CCK-ACD in $\max\{T_L(n, n \cdot \eta), O(n^{2+\omega+\eta})\}$ time by using the LLL algorithm, where $\omega$ is a constant less than 2.38.

Remark: If we use the BKZ algorithm with block size $\beta$ instead of the LLL algorithm for the reduction from scaled CCK-ACD to scaled CRT-ACD, the size of the first output vector $c'$ from the lattice $B''$ is bounded by $4 \cdot \beta^{\frac{k+1}{2}} \cdot 2^{(n-1)\eta+\eta'+2\rho}$. Therefore, the required condition is changed to the following.

$$2 \sqrt{\gamma \cdot \log \beta / (\beta - 1)} \leq \eta - 2\rho + \log \sqrt{2} \gamma - 3 \cdot \log \beta - 2.$$  

### 4 | EXPERIMENTS

In this Section, we provide the experimental results of our algorithms for CRT-ACD, scaled CRT-ACD, and scaled CCK-ACD. All experiments were implemented on a single Intel Core i5 processor running at 2.1 GHz with 16 GB memory.

We implemented the algorithms described in Section 3 with toy parameters. We ran the LLL algorithm in the C++ library fplll [28] instead of the BKZ algorithm, and choose the number of samples $k = 2n$ in CRT-ACD, and scaled CRT-ACD, rather than optimised conditions, for simplicity. The time variable means the time taken to run the LLL algorithm to obtain an auxiliary input.
Table 1 shows the parameter settings we used. In each implementation, we successfully recovered solutions of each problem.

5 | CONCLUSION

This article attempts to extend the SDA algorithm for PACD variants. More precisely, we first classify variants of the PACD problem as CRT-ACD, scaled CCK-ACD, and scaled CRT-ACD problems. According to our results, these problems can be solved under certain conditions using the extended SDA algorithm. Therefore, these results show that the applicable range of the SDA algorithm can be expanded even further.

To date, many schemes have been designed based on PACD and variant problems. Existing schemes conservatively set the parameters to be secure against the SDA algorithm, even though the SDA algorithm does not work. Because the constraints of both our extended algorithm and the original SDA algorithm (and OLA algorithm) are asymptotically same, the parameters set in this way are still robust for our extended SDA algorithm. Nonetheless, it explains that the setup of existing schemes is not set redundantly, but is set well.

Although the present study offers an initial contribution to the literature concerning extending the SDA algorithm, we leave the following open problems for further research.

- Can we improve algorithms to mitigate parameter constraints?
- Our algorithms both require a public modulus $N$, while the original SDA for solving PACD does not need the integer $N$. Can we solve these problems without a public modulus $N$?

ACKNOWLEDGEMENT

Wonhee Cho was supported by the Institute for Information & Communication Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00,598, The mathematical structure of functional encryption and its analysis). Jiseung Kim was supported by a KIAS Individual Grant CG078201 at the Korea Institute for Advanced Study. Changmin Lee was supported by a KIAS Individual Grant CG080601 at the Korea Institute for Advanced Study.

REFERENCES

1. Howgrave-Graham, N.: Approximate integer common divisors. In: Proceedings of the Cryptography and Lattices, International Conference, CaLC 2001, pp. 51–66, Providence, 29–30 March 2001. https://doi.org/10.1007/3-540-44670-2_6
2. Van Dijk, M., et al.: Fully homomorphic encryption over the integers. In: Proceedings of the Advances in Cryptology—EUROCRYPT 2010, pp. 24–43. Monaco and Nice (2010)
3. Cheon, J.H., et al.: Batch fully homomorphic encryption over the integers. In: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 315–335. Springer, Berlin, Heidelberg (2013)
4. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Proceedings of the Advances in Cryptology—CRYPTO 2013, pp. 476–493. Berlin, Heidelberg (2013)
5. Galbraith, S.D., Gebregiyorgis, S.W., Murphy, S.: Algorithms for the approximate common divisor problem. LMS J. Comput. Math. 19(A), 58–72 (2016)
6. Cheon, J.H., Kim, M., Lauter, K.: Homomorphic computation of edit distance. In: Proceedings of the International Conference on Financial Cryptography and Data Security, pp. 194–212. Springer, Berlin, Heidelberg (2015)
7. Cheon, J.H., Kim, M., Kim, M.: Search-and-compute on encrypted data. In: Proceedings of the International Conference on Financial Cryptography and Data Security, pp. 142–159. Springer, Berlin, Heidelberg (2015)
8. Tang, J., et al.: Ensuring security and privacy preservation for cloud data services. ACM Comput. Surv. 49(1), 13 (2016)
9. Huang, D., et al.: Modelling and analysis in marine big data: advances and challenges. Math. Probl. Eng. 2015, 384742 (2015)
10. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: new constructions and applications. In: Proceedings of the EUROCRYPT (2), pp. 69–100. Springer, Berlin, Heidelberg (2015)
11. Attrapadung, N.: Fully secure and succinct attribute based encryption for circuits from multi-linear maps. IACR Cryptol. ePrint Arch., 833, 772 (2014)
12. Benhamouda, F., Pointcheval, D.: Verifier-based password-authenticated key exchange: new models and constructions. IACR Cryptol. ePrint Arch. 2013, 833 (2013)
13. Boneh, D., et al.: Key homomorphic prfs and their applications. In: Proceedings of the Advances in Cryptology—CRYPTO 2013, pp. 410–428. Springer, Berlin, Heidelberg (2013)
14. Garg, S., et al.: Fully secure functional encryption without obfuscation. IACR Cryptol. ePrint Arch. Available from http://eprint.iacr.org/2014/666 666 (2014)
15. Garg, S., et al.: Fully secure attribute based encryption from multilinear maps. IACR Cryptol. ePrint Arch. Available from http://eprint.iacr.org/2014/622 622 (2014)
16. Gentry, C., Lewko, A., Waters, B.: Witness encryption from instance independent assumptions. In: Proceedings of the Advances in

ORCID

Changmin Lee https://orcid.org/0000-0001-8191-1516

TABLE 1 Experimental results for ACD variants

| CRT-ACD | Scaled CRT-ACD | Scaled CCK-ACD |
|---------|----------------|----------------|
| $n$ | $k$ | $\eta$ | $\rho$ | Time (sec) | $n$ | $k$ | $\eta$ | $\rho$ | Time (sec) | $n$ | $k$ | $\gamma$ | $\eta$ | $\rho$ | Time (sec) |
| 30 | 60 | 300 | 43 | 40 | 135 | 30 | 126 | 8000 | 240 | 57 | 4846 |
| 40 | 80 | 400 | 59 | 360 | 1641 | 40 | 134 | 9000 | 200 | 33 | 6996 |
| 50 | 100 | 500 | 74 | 2365 | 10,890 | 50 | 141 | 10,000 | 180 | 20 | 10,116 |
| 60 | 120 | 400 | 50 | 4412 | 17,772 | 60 | 148 | 9000 | 170 | 11 | 9735 |
| 70 | 140 | 300 | 26 | 5975 | 21,213 | 70 | 154 | 11,000 | 165 | 6 | 10,795 |
17. Gentry, C., et al.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In: Proceedings of the Annual Symposium on Foundations of Computer Science, pp. 151–170. IEEE, New York, NY (2015)

18. Lewi, K., Montgomery, H., Raghunathan, A.: Improved constructions of PRFs secure against related-key attacks. In: Proceedings of International Conference on Applied Cryptography and Network Security, vol. 8479 of LNCS, pp. 44–61. Springer, Cham (2014)

19. Zhandry, M.: Adaptively secure broadcast encryption with small system parameters. IACR Cryptol. ePrint Arch. Available from http://eprint.iacr.org/2014/757 (2014)

20. Zimmerman, J.: How to obfuscate programs directly. In: Proceedings of the Advances in Cryptology–EUROCRYPT 2015, pp. 439–467. Springer, Berlin, Heidelberg (2015)

21. Cheon, J.H., et al.: Algorithms for crt-variant of approximate greatest common divisor problem. J. Math. Cryptol. 14(1), 397–413 (2020)

22. Cheon, J.H., et al.: Cryptanalysis of the multilinear map over the integers. In: Proceedings of the Advances in Cryptology–EUROCRYPT 2015, pp. 3–12. Springer, Berlin, Heidelberg (2015)

23. Coron, J.-S., Pereira, H.V.L.: On Kilian’s randomisation of multilinear map encodings. In: Proceedings of the Asiacrypt 2019, pp. 325–355. Springer, Cham (2019)

24. Cheon, J.H., et al.: Cryptanalysis of the CLT13 multilinear map. J. Cryptol. 32(2), 547–565 (2019). https://doi.org/10.1007/s001450189307y

25. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press (2012). https://www.amazon.co.uk/Mathematics/Public-Cryptography-Steven-Galbraith/dp/1107013925/ref=sr_1_16?ie=UTF8&qid=1331769976&sr=8-16

26. Hanrot, G., Pujol, X., Stehlé, D.: Terminating bkz. IACR Cryptol. ePrint Arch. 2011, 198 (2011)

27. Howgrave-Graham, N.: Approximate integer common divisors. International Cryptography and Lattices Conference. pp. 51–66. Springer, Berlin, Heidelberg (2001)

28. Development Team, T.F.: fplll, a lattice reduction library. Available from https://github.com/fplll/fplll (2016)