Received 8 April 2024, accepted 4 May 2024, date of publication 27 May 2024, date of current version 11 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3405609

RESEARCH ARTICLE

An Efficient Implementation Scheme for Lattice Reduction in the List-Decoding Algorithm for the Binary Goppa Codes

KI-SOON YU AND DAE-WOON LIM
Department of Information and Communication Engineering, Dongguk University, Seoul 04620, Republic of Korea

Corresponding author: Dae-Woon Lim (daewoonlim@gmail.com)

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by Korean Government [Ministry of Science and ICT (MSIT)] under Grant NRF-2021R1F1A1052578, and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by Korean Government (MSIT) (Development of Highly Efficient Post-Quantum Cryptography (PQC) Security and Performance Verification for Constrained Devices) under Grant 2021-0-00400.

ABSTRACT This paper presents a scheme that is designed for the effective implementation of lattice reduction for polynomial matrices within the list-decoding algorithm that is applied to the binary Goppa codes. The reduced form of a polynomial matrix is obtained by transforming the given polynomial matrix into a matrix in the weak Popov form. To achieve efficient lattice reduction within the list-decoding algorithm, the proposed scheme reorganizes the polynomial matrix by leveraging its inherent properties and converts it into the weak Popov form. When using the proposed implementation technique to convert the reorganized polynomial matrix into the weak Popov form, the number of simple transformations of the first kind that had to be performed was reduced by about 15% compared to the technique used to convert the original matrix to the weak Popov form. As a result, the execution time of lattice reduction was also decreased.

INDEX TERMS Binary Goppa codes, McEliece cryptosystem, list-decoding algorithm, weak Popov form, polynomial matrix, lattice reduction.

I. INTRODUCTION

The McEliece cryptosystem was introduced in 1978 as a code-based public-key cryptosystem [1]. In 1986, the Niederreiter cryptosystem was developed as a variation of the McEliece cryptosystem. In contrast to the McEliece cryptosystem, the Niederreiter cryptosystem employs a parity matrix of a code as its public key rather than a generation matrix of the code, while still achieving security levels equivalent to those of the McEliece cryptosystem [2]. The Classical McEliece, which is a finalist in the Key Encapsulation Mechanism (KEM) category of the NIST Post-Quantum Cryptography Standardization project, currently also employs a parity matrix as its public key, like the Niederreiter cryptosystem [3].

The principle of the McEliece cryptosystem is to select a linear code of length n and dimension k that can correct t errors [4]. In the McEliece cryptosystem, errors are added to a codeword to generate the ciphertext. Then, during the decoding process, errors are found using the decoding algorithm of the linear code. Increasing the number of errors has been shown to enhance the security and resistance of the McEliece cryptosystem [5].

In [6], a McEliece-type cryptosystem was introduced, and it was shown that an elevated security level was achieved by reducing the key size by using large minimum distance error-correcting codes derived from self-dual codes. Several variants of McEliece cryptosystems have also been proposed with the aim of enhancing security levels while

The associate editor coordinating the review of this manuscript and approving it for publication was Zilong Liu.

minimizing key size through the application of diverse error correction codes [4], [7], [8], [9]. However, their security is no longer guaranteed, as effective attack techniques have been disclosed against most variants. On the other hand, the McEliece cryptosystem based on the binary Goppa codes is still considered secure, as efficient attack techniques against it have yet to be disclosed [10].

The decoding algorithms for the binary Goppa codes include the Patterson algorithm, the Berlekamp-Massey algorithm, the EEA algorithm, the list-decoding algorithm, and others [5], [11], [12]. While other decoding algorithms can correct up to t errors in the received ciphertext, the list-decoding algorithm has the ability to correct up to $t + u$ errors [5]. Therefore, when the list-decoding algorithm is used in the binary Goppa codes based on the McEliece cryptosystem, the probability of no errors in randomly selected k bits out of n bits with t errors becomes $({}_{n-t-u}C_k / {}_nC_k)$ [14], which results in an increased security level. However, this cryptosystem faces challenges such as increased decryption time due to the need for an increased number of operations like lattice reduction and factorization.

The current paper proposes an implementation scheme that can be used to reduce lattice reduction operation time, thus addressing the issue of increased execution time with the list-decoding algorithm in the McEliece cryptosystem.

II. PRELIMINARIES

A. THE LIST-DECODING OF THE BINARY GOPPA CODES

The binary Goppa codes Γ is defined by the Goppa polynomial $g(x) \in F_{2^m}[x]$ which is a monic degree-t irreducible polynomial and a subset $\{\alpha_1, \alpha_2, \cdots, \alpha_n\} \subseteq F_{2^m}$ [5].

$$\begin{aligned}\Gamma &= \Gamma(\alpha_1, \alpha_2, \cdots, \alpha_n, g(x)) \\ &= \left\{ c \in F_2^n : \sum_i c_i \frac{h}{x - \alpha_i} \; (mod\ g(x)) \equiv 0 \right\} \qquad (1)\end{aligned}$$

where a codeword $c = (c_1, c_2, \ldots, c_n)$ and a polynomial $h = \prod_i (x - \alpha_i) \in F_{2^m}[x]$. Γ has a minimum distance of at least $2t+1$, hence the binary Goppa codes can correct t errors [23]. Thus, the maximum degree of the error location polynomial $\epsilon(x) = \prod_{i:r_i \neq c_i} (x - \alpha_i) \in F_{2^m}[x]$ calculated by the Patterson algorithm to decode the original codeword c of the input word r is t.

On the other hand, the list-decoding algorithm is an extension of the Patterson algorithm [13], and it can correct u errors in addition to t errors [5]. Here, the parameter u takes a value in the range from $0 \leq u \leq n - t - \sqrt{n(n - 2t - 2)}$ [5]. Algorithm 1 shows how the list-decoding algorithm computes the error location polynomial [5]. The maximum degree of the calculated error location polynomial is $t + u$.

Algorithm 1 The List-Decoding Algorihtm

```
Input  : A word $r = (r_1, r_2, \ldots, r_n)$
Output: The error location polynomial $\epsilon(x)$
 1: Compute $s(x) = \sqrt{(1/\sum_i r_i/(x - \alpha_i))} - x \in F_{2^m}[x]$.
 2: Find $(a_0(x), b_0(x))$ and $(a_1(x), b_1(x))$.
      1) Construct a lattice $\Xi \subseteq F_{2^m}[x]^2$ with basis $(s(x), 1)$
         and $(g(x), 0)$.
      2) Perform a lattice basis reduction on $\Xi$ and obtain the
         minimum-length nonzero vector $(a_0(x), b_0(x))$ and the
         minimum-length independent vector $(a_1(x), b_1(x))$.
 3: Set $\epsilon_0(x) = a_0^2(x) + xb_0^2(x)$ and
    $\epsilon_1(x) = a_1^2(x) + xb_1^2(x) + \gamma\epsilon_0(x)$.
      1) Set $\gamma \in F_{2^m}$.
      2) Set $\epsilon_1(x) = a_1^2(x) + xb_1^2(x) + \gamma\epsilon_0(x)$.
      3) If $gcd\{\epsilon_1(x), h(x)\} = 1$, set
         $\epsilon_1(x) = a_1^2(x) + xb_1^2(x) + \gamma\epsilon_0(x)$ otherwise
         recalculate.
 4: Compute the polynomial
    $\delta(x) = \epsilon_0(x)/\epsilon_1(x) \pmod{h(x)} \in F_{2^m}[x]$.
 5: Define the $l$-dimensional lattice $\Xi \subset F_{2^m}(x)[z]$.
      1) Calculate the integer $\theta$:
           • $t_0 = \mathrm{deg}\epsilon_0(x)$;
           • $g_0 = 2\lfloor(u + t - t_0)/2\rfloor$;
           • $g_1 = 2\lfloor(u + t_0 - t - 1)/2\rfloor$;
           • $\theta = g_1 - g_0$.
      2) Select the integers $l$ and $d$ that satisfy the following
         conditions:
           • $(g_0 + g_1)(l - 1)/2d + n(d + 1)/2l < (t + u)$;
           • $l > d > 0$.
      3) Define the $l$-dimensional lattice $\Xi \subset F_{2^m}(x)[z]$
         generated with the following polynomials:
         $$1, \frac{x^\theta z + \delta(x)}{h(x)}, \left(\frac{x^\theta z + \delta(x)}{h(x)}\right)^2, \cdots, \left(\frac{x^\theta z + \delta(x)}{h(x)}\right)^d,$$
         $$x^\theta z\left(\frac{x^\theta z + \delta(x)}{h(x)}\right)^d, (x^\theta z)^2\left(\frac{x^\theta z + \delta(x)}{h(x)}\right)^d,$$
         $$\cdots, (x^\theta z)^{l-d-1}\left(\frac{x^\theta z + \delta(x)}{h(x)}\right)^d.$$
 6: Generate a polynomial matrix $M$ using the coefficients of
    $1, z, z^2, \cdots, z^{l-1}$ in $\Xi$.
 7: Convert the polynomial matrix to the weak Popov form.
 8: Find the minimum-length nonzero vector
    $\phi(z) = \phi_0(x) + \phi_1(x)z + \ldots \in \Xi$.
      1) The norm of a polynomial $\phi_i(x) \in F_{2^m}[x]$ is
         defined as $|\phi_i(x)| = 2^{\mathrm{deg}\phi_i(x)}$.
      2) The length of a vector is defined as
         $|\phi(z)| = \max\{|\phi_0(x)|, |\phi_1(x)|, \cdots\}$.
 9: Find the root of $\phi(z)$ that satisfies the form $q_1^2(x)/x^\theta q_0^2(x)$.
10: Set the error locator polynomial
    $\epsilon(x) = q_0^2(x)\epsilon_0(x) + q_1^2(x)\epsilon_1(x)$. The error vector
      $e = (e_1, e_2, \ldots, e_n)$ is defined as follows:
         $$e_i = \begin{cases} 0, & c_i = r_i \\ 1, & c_i \neq r_i. \end{cases}$$
    The roots of $\epsilon(x)$ indicate the error location with $e_i = 1$.
```

B. LATTICE REDUCTION FOR POLYNOMIAL MATRICES

As illustrated in Algorithm 1, the list-decoding algorithm computes the minimum-length nonzero vector $\phi(z)$ of the

l-dimensional lattice to find $q_0(x)$ and $q_1(x)$ [5]. The l-dimensional lattice denoted as $\Xi \subset F_{2^m}(x)[z]$ is shown as follows:

$$\begin{bmatrix} \Xi_0(z) \\ \Xi_1(z) \\ \Xi_2(z) \\ \vdots \\ \Xi_{l-1}(z) \end{bmatrix} = \begin{bmatrix} M_{0,0}(x) & M_{0,1}(x) & \cdots & M_{0,l-1}(x) \\ M_{1,0}(x) & M_{1,1}(x) & \cdots & M_{1,l-1}(x) \\ M_{2,0}(x) & M_{2,1}(x) & \cdots & M_{2,l-1}(x) \\ \vdots & \vdots & \ddots & \vdots \\ M_{l-1,0}(x) & M_{l-1,1}(x) & \cdots & M_{l-1,l-1}(x) \end{bmatrix} \times \begin{bmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{l-1} \end{bmatrix} \tag{2}$$

Here, the generators $\Xi_i(z)$ are as follows:

$$\Xi_i(z) = M_{i,0} + M_{i,1} + \ldots + M_{i,l-1}(x)z^{l-1} = \begin{cases} \left(\dfrac{x^\theta z + \delta(x)}{h(x)}\right)^i, & 0 \le i \le d, \\ \left(x^\theta z\right)^{i-d}\left(\dfrac{x^\theta z + \delta(x)}{h(x)}\right)^d, & d+1 \le i \le l-1. \end{cases} \tag{3}$$

$\phi(z)$ can be obtained through lattice reduction to a polynomial matrix $M = (M_{i,j}(x)) \in F_{2^m}[x]^{l \times l}$ composed of coefficients of $\Xi_i(z)$. Lattice reduction for a polynomial matrix M over $F_{2^m}[x]$ comprises the task of finding the basis with the smallest row degree for a lattice generated by a linear combination of the rows of M [15]. The given polynomial matrix can be transformed into either the weak Popov form or the Popov form to obtain a reduced matrix [16], [17]. The matrix transformed into the weak Popov form or the Popov form generates the same subspace as the given polynomial matrix. Algorithms that can be used to convert a polynomial matrix into the weak Popov form include the Lee-O'Sullivan algorithm [18], the Alekhnovich algorithm [19], and the Puchinger-Nielsen-Li-Sidorenko algorithm [21], among others [20], [22].

C. THE WEAK POPOV FORM

Mulders and Storjohann described an algorithm achieving the lattice reduction of polynomial matrices in [16]. In that algorithm, a polynomial matrix $M = (M_{i,j}(x)) \in F[x]^{v \times w}$ is considered to be in the weak Popov form if a pivot index I_i^M of i-th row vector $M_i = (M_{i,0}(x), M_{i,1}(x), \ldots, M_{i,w-1}(x))$ satisfies $I_i^M \neq I_j^M$ whenever $i \neq j$, and when both M_i and M_j are nonzero. Here, the pivot index I_i^M of M_i is defined as follows [16]:

$$I_i^M = \begin{cases} \max\{j | \max\{\deg M_{i,j}(x)\}, 0 \le j < w\} & \text{for} \quad M_i \neq 0 \\ -1 & \text{for} \quad M_i = 0. \end{cases}$$

A given polynomial matrix can be transformed into the weak Popov form by applying unimodular row-transformations [16]. For $i \neq j$, if $I_i^M \neq -1$, $I_j^M \neq -1$, and $\deg M_{i,I_j^M}(x) \ge \deg M_{j,I_j^M}(x)$, and there exists unique $\zeta \in F$ and $\pi \in \mathbb{N}$ such that

$$\deg\left(M_{i,I_j^M}(x) - \zeta x^\pi M_{j,I_j^M}(x)\right) < \deg M_{i,I_j^M}(x).$$

In [16], this operation—where ζx^π times row j is subtracted from row i—is termed as the simple transformation of row j on row i. If $I_i^M = I_j^M$, the operation is termed the simple transformation of the first kind; otherwise, it is termed the simple transformation of the second kind [16]. In this paper, the simple transformation of the first kind is hereafter denoted as ST.

Algorithm 2 The Weak Popov Form

```
Input  : Polynomial matrix $M \in F_{2^m}[x]^{l \times l}$
Output: Reduced polynomial matrix $N$ in the weak
        Popov form
1  begin
2      $N = copy(M)$;
3      for $i$ to $l$ do
4          $I_i^N = find\_pivot\_index(N_i)$;
5          if $I_i^N = 0$ then
6              continue
7          end
8          for $j$ to $i$-$1$ do
9              if $I_j^N = I_i^N$ then
10                 $\pi = \deg N_{j,I_j^N}(x) - \deg N_{i,I_i^N}(x)$;
11                 if $\pi > 0$ then
12                     $swap(N_j, N_i)$;
13                 end
                   /* $LC()$ returns the
                      leading coefficient of
                      a polynomial.                */
14                 $\zeta = LC(N_{i,I_i^N}(x)) \times LC(N_{j,I_j^N}(x))^{-1}$
                   /* $abs()$ returns the
                      absolute value of a
                      number.                      */
15                 $N_i = N_i - \zeta x^{abs(\pi)} N_j$;
16                 $i = i - 1$;
17                 break
18             end
19         end
20     end
21     return $N$;
22 end
```

Algorithm 2 is the pseudocode proposed by [16] to transform a polynomial matrix into the weak Popov form using the ST operation. Algorithm 2 sequentially compares vectors, and if there are vectors with the same pivot index, it performs an ST operation. This process of comparing vectors and performing ST operations is repeated until obtaining a reduced matrix N that conforms to the weak Popov form.

D. THE POPOV FORM

In [17], a matrix $M = (M_{i,j}(x)) \in F[x]^{v \times w}$ is said to be in the Popov form if it satisfies the following conditions:

1) M is in the weak Popov form.
2) $M_{i,I_i^M}(x)$ is monic for all $I_i^M \neq -1$.
3) $\deg M_{i,I_i^M}(x) < \deg M_{j,I_j^M}(x)$ for all $i < j$.

Reference [22] proposed a method to reduce computational complexity by utilizing the saturation of M to efficiently find the pivot support of M and then use this information to calculate the Popov form of M. When this method is used, the complexity bound is reduced $\tilde{O}(vwRank(M)(degM)^2)$ to $\tilde{O}(v^{\omega-1}w(degM))$. where the value of ω is typically between 2 and 3 and $(degM)$ is defined as the maximum of the degree of M entries.

We can transform a matrix that is in the weak Popov form into the Popov form through operations such as swapping the positions of the rows, the simple transformation of the second kind, and multiplying rows by scalars [16].

In this paper, we propose an implementation scheme that efficiently transforms a polynomial matrix into the weak Popov form by leveraging the features of the polynomial matrix that are generated during the list-decoding algorithm.

III. AN EFFICIENT IMPLEMENTATION SCHEME OF LATTICE REDUCTION IN THE LIST-DECODING ALGORITHM

This paper proposes an efficient scheme for transforming a polynomial matrix that generated in the list-decoding algorithm—specifically when the integer parameter d of the l-dimensional lattice $\Xi \subset F_{2^m}(x)[z]$ is 2^β—into the weak Popov form.

In the finite field F_{2^m}, raising a polynomial to the power of 2^β is equivalent to raising each term of the polynomial to the power of 2^β. Therefore, when d is 2^β, among the polynomials that generate Ξ, $((x^\theta z + \delta(x))/h(x))^d$ is a polynomial consisting of two terms $((x^\theta z)/h(x))$ and $(\delta(x)/h(x))$ raised to the power of 2^β, respectively. Similarly, the polynomials resulting from multiplying $((x^\theta z+\delta(x))/h(x))^d$ by $(x^\theta z)^i$, $1 \leq i \leq l-d-1$ also consist of two terms. Therefore, the polynomial matrix generated based on the coefficients of the above polynomials contains a significant number of zero elements. This allows us to predict which other row vectors are likely to share the same pivot index as the index of the modified pivot when the pivot of a row vector in a polynomial matrix changes. The matrix T shows the polynomial matrix $M \in F_{2^m}[x]^{l \times l}$, where the coefficients of $1, z, z^2, \ldots, z^{l-1}$ are represented as row vectors. Here, the element $T_{i,j}$ is displayed as follows:

$$T_{i,j} = \begin{cases} 0, & M_{i,j}(x) = 0 \\ 1, & M_{i,j}(x) \neq 0. \end{cases}$$

The value of u lies within the range of $0 \leq u \leq n - t - \sqrt{n(n-2t-2)}$. However, in the case where $u = 0$, other decoding algorithms can also decrypt ciphertexts efficiently. Hence, this paper assumes that u is greater than zero.

Since $t_0 \leq t$, in cases where $u > 0$ and $t_0 = t$, then $g_0 = u - u\%2$ and $g_1 = (u-1) - (u-1)\%2$. Consequently,

$$\begin{aligned} &1, \frac{x^\theta z + \delta(x)}{h(x)}, \left(\frac{x^{2\theta}z^2 + \delta^2(x)}{h^2(x)}\right), \left(\frac{x^{3\theta}z^3 + \delta(x)x^{2\theta}z^2 + \delta^2(x)x^\theta z + \delta^3(x)}{h^3}\right), \ldots, \left(\frac{x^{d\theta}z^d + \delta^d(x)}{h^d(x)}\right), \\ &\left(\frac{x^{(d+1)\theta}z^{d+1} + \delta^d(x)x^\theta z}{h^d(x)}\right), \left(\frac{x^{(d+2)\theta}z^{d+2} + \delta^d(x)x^{2\theta}z^2}{h^d(x)}\right), \ldots, \left(\frac{x^{(l-1)\theta}z^{l-1} + \delta^d(x)x^{(l-d-1)\theta}z^{l-d-1}}{h^d(x)}\right). \end{aligned} \tag{4}$$

$$T = \left[\begin{array}{cccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & & 0 & & 0 & & 0 \\
1 & 1 & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & & 0 & & 0 & & 0 \\
1 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
1 & 1 & 1 & 1 & 0 & & 0 & 0 & 0 & & 0 & 0 & & 0 & & 0 & & 0 \\
1 & 0 & 0 & 0 & 1 & & 0 & 0 & 0 & & 0 & 0 & & 0 & & 0 & & 0 \\
 & & \vdots & & & \ddots & & \vdots & & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
1 & 0 & 0 & 0 & 0 & & 1 & 0 & 0 & & 0 & 0 & & 0 & & 0 & & 0 \\
0 & 1 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & 1 & 0 & 0 & & 0 & 0 & 1 & & 0 & 0 & & 0 & & 0 & & 0 \\
 & & \vdots & & & \ddots & & \vdots & & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
 & & \vdots & & & \ddots & & \vdots & & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & \cdots & 1 & \cdots & 0 \\
 & & \vdots & & & \ddots & & \vdots & & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 1 & \cdots & 1
\end{array}\right]
\begin{array}{c}
0 \\ 1 \\ 2 \\ 3 \\ 4 \\ \vdots \\ d \\ d+1 \\ d+2 \\ \vdots \\ 2d \\ \vdots \\ l-d-1 \\ \vdots \\ l-1
\end{array} \tag{5}$$

when u is even, $g_0 > g_1$, while when u is odd, $g_0 = g_1$. If $u > 0$, $t_0 < t$, then $t_0 - t - 1 < t - t_0$, which always means that $g_1 \leq g_0$. Consequently, $\theta = g_1 - g_0 \leq 0$. Therefore, M is scaled by multiplying $x^{(-\theta)(l-1)}h^d(x)$. At this point, by converting the scaled M to the weak Popov form, we can avoid the complexities associated with fractional arithmetic.

As before scaling, the elements of M corresponding to the constant terms in the polynomials generating Ξ become the pivot elements $M_{i,I_i^M}(x)$ of the i-th row vector M_i. Therefore, the pivot indices from M_{d+1} to M_{l-1} of M are different whereas the pivot indices from M_0 to M_d are the same. Hence, the conditions for the weak Popov form cannot be satisfied. However, we can transform M into the weak Popov form, denoted as N, by applying ST operation to its row vectors.

According to [16], the maximum number of ST operations that can be applied during the process of converting M into N is given by $S^{M,N} = S^M - S^N$. Here, S^M is defined as $S^M = \sum_{i \in C^M}(\text{deg}M_{i,I_i^M}(x)\ rank(M) + I_i^M)$, where $C^M = \{i | I_i^M \neq -1, 0 \leq i < l\}$. The number of ST operations performed is affected by the particularly sequence in which the row vectors are utilized in ST operations.

Definition 1: The degree difference of a polynomial $p(x) = p_0 + p_1x + p_2x^2 + \ldots + p_\tau x^\tau$ is defined as:

$$D_{poly}^{p(x)} = min(i - j)$$

where $i, j \in \{i, j | p_i \neq 0, p_j \neq 0, 0 \leq j < i \leq \tau\}$. If a polynomial is a monomial, its degree difference is considered to be infinite.

Definition 2: The degree difference of a row vector $P = (p_0(x), p_1(x), p_2(x), \ldots)$ with polynomials as components is defined as:

$$D_{vec}^{P} = min(D_{poly}^{p_i(x)})$$

where $P \neq 0$ and $p_i(x) \neq 0$.

In a scenario where multiple row vectors are provided, let us assume that their pivot indices are identical. To modify the pivot of a specific row vector, ST operation is employed by selecting a row vector from various other row vectors with matching pivot indices. The number of ST operation executions needed to modify the pivot depends on the chosen row vector.

In the example of (6), as shown at the bottom of the page, the row vectors share the same pivot index. The degree difference of the 0th-row vector is considered to be infinite, the degree difference of the 1st-row vector is 1, and the degree difference of the 2nd-row vector is 4. To change the pivot index of the 0th-row vector by using the 1st-row vector, 4 ST operations are required. Conversely, if the 2nd-row vector is chosen, the pivot index of the 0th-vector can be altered by performing only 1 ST operation. The degree difference among row vectors significantly influences the number of ST operations required, as illustrated in the example.

M is formed based on the coefficients of $\Xi_i(z)$; in this paper, it is specified that $d = 2^\beta$. Consequently, the elements of M consist of polynomials with degree differences of approximately $D_{poly}^{\delta(x)}, 2D_{poly}^{\delta(x)}, 2^2D_{poly}^{\delta(x)}, \cdots, 2^\beta D_{poly}^{\delta(x)}$. Therefore, the row vectors of M can be divided according to these degree differences.

During pivot alterations, consideration is given to the fact that it is advantageous to use row vectors with large degree differences for pivot changes. So, we initially distinguish between the row vectors whose pivot indices overlap and the row vectors whose pivot indices do not overlap. The row vectors with overlapping pivot indices are organized into sets according to their degree differences. Subsequently, when the pivot of row vectors within a set undergoes modification due to ST operation, row vectors with the potential to share the same pivot index as the pivot index of the modified pivot are discerned from the non-overlapping row vectors and included as elements. The degree difference of the added row vectors is guaranteed to be greater than or equal to the degree differences of the row vectors forming the set. By separating row vectors in this manner, it is feasible to initially execute the ST operation on the row vectors with large degree differences. As a result, the ST operation can be sequentially applied to row vectors with small degree differences.

The row vectors from the 0th to the d-th positions in M share the same pivot index, while the pivot indices for the row vectors from the $(d + 1)$-th to the $(l - 1)$-th positions are distinct. Further, the degree difference for the row vectors from the $(d+1)$-th to the $(l-1)$-th positions is greater than or equal to $2^\beta D_{vec}^{\delta(x)}$. To generate sets of row indices based on the degree differences of the row vectors in M, the row vectors from 0th to $(d - 1)$-th are classified as follows:

- $\Psi_\beta = \{i \mid 2^\beta D_{poly}^{\delta(x)} \leq D_{vec}^{M_i}, 0 \leq i \leq d - 1\}$,
- $\Psi_{\beta-1} = \{i \mid 2^{\beta-1} D_{poly}^{\delta(x)} \leq D_{vec}^{M_i} < 2^\beta D_{poly}^{\delta(x)}, 0 \leq i \leq d - 1\}$,
- $\cdots$,
- $\Psi_0 = \{i \mid D_{poly}^{\delta(x)} \leq D_{vec}^{M_i} < 2D_{poly}^{\delta(x)}, 0 \leq i \leq d - 1\}$.

If a set contains more than two elements, they are listed in ascending order.

Following this, when the pivots of the previously classified row vectors change, a search is conducted to identify row vectors that may share overlapping pivot indices with the pivot indices of the changed pivots. Here, the row

$$\begin{bmatrix} \alpha^{39}x^{16} & 0 & 0 \\ \alpha^{31}+\alpha^{11}x+\alpha^{24}x^2+\alpha^{40}x^3+\alpha^{12}x^4+\alpha^{55}x^5+\alpha^{34}x^6+\alpha^{19}x^7+\alpha^{36}x^8 & \alpha^{36}+\alpha^{52}x+\alpha^{16}x^2+\alpha^{31}x^3+\alpha^{30}x^4 & 0 \\ \alpha^{36}+\alpha^{17}x^4+\alpha^{50}x^8 & 0 & \alpha^8+\alpha^{16}x^4 \end{bmatrix} \tag{6}$$

Algorithm 3 The Proposed Row-Reordering Method

```
Input  : $d, l, M, D_{poly}^{\delta(x)}$
Output: Reordered sequence $IDX$
1  begin
2      for $i$ to $d$-1 do
3          for $j = \beta$ to 0 do
4              if $D_{vec}^{M_i} \geq 2^j D_{poly}^{\delta(x)}$ then
5                  $\Psi_j \cup = \{i\}$;
6                  break
7              end
8          end
9      end
10     $\rho = 0$;
11     for $j = \beta$ to 0 do
12         for $i$ to $|\Psi_j|$ do
13             $\lambda = 0$;
14             repeat
15                 $idx_\rho = \Psi_j^i + d \times \lambda$ ; //  $\Psi_j^i \in \Psi_j$
16                 $\lambda = \lambda + 1$;
17                 $\rho = \rho + 1$;
18             until $\Psi_j^i + d \times \lambda < l$;
19         end
20     end
21     return $IDX$;
22 end
```

vectors are not redundantly selected, as they are selected as follows:

- $\Psi_\beta \cup = \{i + d \times \lambda \mid i \in \Psi_\beta, 1 \leq \lambda, (i + d \times \lambda) < l\}$,
- $\Psi_{\beta-1} \cup = \{i+d \times \lambda \mid i \in \Psi_{\beta-1}, 1 \leq \lambda, (i+d \times \lambda) < l\}$,
- $\cdots$,
- $\Psi_0 \cup = \{i + d \times \lambda \mid i \in \Psi_0, 1 \leq \lambda, (i + d \times \lambda) < l\}$.

The row vectors of M are rearranged using the $\beta + 1$ sets generated in this way. When a row vector with a small degree difference is used to change the pivots of other row vectors, it affects the degree difference of the other row vectors. Therefore, the row vectors of M are rearranged by referring to the order of $\Psi_\beta, \Psi_{\beta-1}, \cdots, \Psi_0$. The polynomial matrix whose rows are rearranged is denoted as M'. Algorithm 3 generates sets as described previously and returns the information IDX that can then be used to rearrange the row vectors of M. In the process of generating the lattice, if the parameters (d, l) and $D_{poly}^{\delta(x)}$ remain consistent with the previously computed values, IDX will also be the same as before. Therefore, when (d, l) and $D_{poly}^{\delta(x)}$ are identical to the previous values, precomputed IDX can be utilized.

Algorithm 4 demonstrates the procedure for transforming the converted matrix M' into the weak Popov form by reordering the rows of M according to IDX. By transforming M' into the weak Popov form, the number of ST operations can be reduced.

The number of ST operations executed during the transformation of M into the weak Popov form is expressed as follows:

$$ST^M = \sum_{i=1}^{l-1} (\eta_i - \eta_{i-1}) \tag{7}$$

where η_i denotes the number of ST operations performed until the row vectors from the 0th to the i-th row vector possess distinct pivot indices. Therefore, $\eta_0 = 0$.

Meanwhile, Algorithm 2 sequentially compares the row vectors and applies ST operation. As a consequence, when transforming the given matrix M into the weak Popov form, the row vectors with small degree differences are the first ones used in ST operation. This in turn affects the degree differences of the other row vectors, ultimately leading to an increase in the number of ST operations. On the other hand, when converting M', the ST operation is applied in the order from M_i, where $i \in \Psi_\beta$, to M_i, where $i \in \Psi_0$. Consequently, during ST operations on M_i, where $i \in \Psi_j$, the degree differences are maintained at 2^j. Thus, during the process of transforming M' into the weak Popov form, the number of ST operations between the row vectors corresponding to $\Psi_\beta, \Psi_{\beta-1}, \cdots, \Psi_0$ is reduced by approximately $1/2^\beta, 1/2^{\beta-1}, \cdots, 1$.

The number of ST operations performed in the process of converting M' to weak Popov form is defined as follows:

$$ST^{M'} = ST_\beta^M + ST_{\beta-1}^M + \cdots + ST_0^M. \tag{8}$$

Algorithm 4 The Proposed scheme

```
Input  : $d, l, M, D_{poly}^{\delta(x)}$
Output: Reduced polynomial matrix $N$ in the weak
        Popov form
1  begin
       /* Use Algorithm 3 to obtain $IDX$.
          */
2      $IDX = row\_reordering(d, l, M, D_{poly}^{\delta(x)})$;
       /* Rearrange the rows of $M$
          according to $IDX$ to convert it
          to $M'$.                          */
3      for $i$ to $l$ do
4          tmp = $M_i$;
5          $M_i = M_{IDX_i}$;
6          $M_{IDX_i}$ = tmp;
7      end
8      $N = weak\_Popov\_form(M')$;
9      return $N$;
10 end
```

TABLE 1. The test environment.

Project	Description
OS	Ubuntu 18.04.6 LTS(amd64)
CPU	11-th Gen Intel(R) Core(TM) i7-11700 @ 2.50GH
Memory	64GB
Language	C
Compiler	gcc version 7.5.0

TABLE 2. The number of ST operations.

(n,k,t,d,l)	Conventional: Algorithm 2 [16]	Proposed: Algorithm 4	Operation Reduction (%)
(64,4,10,8,42)	401724.0	339989.4	15.4
(128,16,16,4,30)	208800.3	179321.9	14.1
(256,80,22,8,87)	7360447.6	6246631.2	15.1

TABLE 3. The execution time for conversion.

(n,k,t,d,l)	Conventional: Algorithm 2 [16] (s)	Proposed: Algorithm 4 (s)	Operation Reduction (%)
(64,4,10,8,42)	15.7	12.0	23.4
(128,16,16,4,30)	5.1	4.3	16.6
(256,80,22,8,87)	1770.0	1444.9	18.4

TABLE 4. Theoretical and experimental values of ST operation numbers.

(n,k,t,d,l)	Theoretical Values	Experimental Values	Percent Error (%)
(64,4,10,8,42)	358489.5	339989.4	5.2
(128,16,16,4,30)	185248.2	179321.9	3.2
(256,80,22,8,87)	6423463.7	6246631.2	2.8

Here, ST_i^M can be expressed as follows:

$$ST_i^M = \left(\eta_{\sum_{j=i}^{\beta}|\Psi_j|-1} - \eta_{\sum_{j=i+1}^{\beta}|\Psi_j|-1}\right) \times \frac{1}{2^i}. \tag{9}$$

In this section, we have proposed a method that can be used to rearrange the row vectors based on the degree differences of the row vectors to efficiently reduce the polynomial matrix in the list-decoding algorithm. We have also demonstrated that the number of ST operations decreases when transforming the modified polynomial matrix into the weak Popov form according to the proposed technique.

IV. PERFORMANCE EVALUATION

We have introduced an efficient technique for transforming polynomial matrices generated in the list-decoding algorithm to the weak Popov form when the parameter d is 2^β. The proposed technique involves reordering row vectors within a specified polynomial matrix while considering their degree differences. The process for evaluating the performance of the proposed technique is as follows:

1) During the encoding process of the binary Goppa codes, an error vector with a Hamming weight of $t+u$ is randomly generated and added to a codeword.
2) The matrix M generated in the decoding process based on the list-decoding algorithm as well as the modified matrix M' using the proposed method are separately transformed into the weak Popov form.
3) In line 9 of Algorithm 2, the given row vector is compared with previous row vectors, and the number of ST operations is recorded if there is a row vector with the same pivot index.
4) To validate whether M and M' have each been correctly converted to the weak Popov form, we find the roots of $\phi(z)$ of M and $\phi(z)$ of M' obtained from the reduction outcomes. Subsequently, we verify that both possess an equal root of the desired form $q_1^2(x)/x^\theta q_0^2(x)$.

The test environment is described in Table 1.

Table 2 provides a comparison of the number of ST operations performed for each parameter. In line 9 of Algorithm 2, the count was recorded whenever row vectors with the same pivot index were encountered. After the polynomial matrix was converted using the proposed scheme, a reduction of around 15% in the count of ST operations could be observed during the conversion process to the weak Popov form.

Table 3 presents the time taken to convert M and M' to the weak Popov form by each parameter. The time was measured at lines 3 and 20 of Algorithm 2 to calculate the execution time.

If the parameters (d, l) and $D_{poly}^{\delta(x)}$ remain consistent with their previously computed values, we can utilize the previously calculated IDX. Hence, in Table 3, the computation time for IDX is omitted from the execution time of transforming M' into the weak Popov form, and only the time to reference IDX and transform M into M' is additionally included. The execution time of Algorithm 2 decreased when transforming M' into the weak Popov form compared to the corresponding time when transforming M into the weak Popov form.

Table 4 presents a comparison between theoretical and experimental values of the count of ST operations during the process of converting M' to the weak Popov form. While there are slight differences for each parameter, the maximum error rate observed in the experiments is 5.2%.

Upon converting M' into the weak Popov form, the number of ST operations decreases, which leads to a reduction in the overall execution time required for polynomial matrix reduction.

It is not necessary to execute Algorithm 3 every time during decryption. However, to store the precomputed *IDX*, it is necessary to have additional memory of $l \times 4$ bytes. Further, as pointers are employed to generate M', extra memory of another $l \times 4$ bytes is also required.

V. CONCLUSION

The fact that the McEliece cryptosystem utilizes the list-decoding algorithm enhances security by using u additional errors alongside t errors to create ciphertexts. However, an issue arises that decoding time increases, as lattice reduction and factorization operations become necessary due to these additional errors.

In this paper, to address the issue of increasing execution time caused by lattice reduction, we proposed a scheme that efficiently reduces the polynomial matrix of the l-dimensional lattice generated from the list-decoding algorithm. The proposed scheme is designed to decrease the execution time of lattice reduction by reducing the number of ST operations required when transforming a given polynomial matrix into the weak Popov form.

In cases where $d = 2^\beta$, we reorganize the row vectors within the given polynomial matrix in accordance with its properties, thereby reducing the number of ST operations. Although this process necessitates additional memory of $2l \times 4$ bytes to rearrange the row vectors, through this experimentation, we have verified a reduction of approximately 15% in the number of ST operations when transforming the reordered polynomial matrix into the weak Popov form.

While there are variations among the parameters, compared to the transformation of the original polynomial matrix into the weak Popov form, the execution time of transforming the polynomial matrix with rearranged row vectors into the weak Popov form decreased by approximately 16% to 23% when using the proposed scheme.

When the proposed scheme is applied to a polynomial matrix containing numerous zero elements and row vectors with large degree differences, the frequency of ST operations is significantly diminished. Conversely, when this scheme is applied to a polynomial matrix lacking these properties, the reduction in the number of ST operations is not substantial. However, even if d is not a power of 2, q_0 and q_1 are obtained by expanding d to a power of 2, creating a lattice, and converting it to the weak Popov form. Accordingly, if necessary, the proposed algorithm can be applied by converting d to a power of 2.

APPENDIX A
EXAMPLES OF THE LIST-DECODING ALGORITHM

Let $\Gamma(L, g(x))$ be the binary Goppa codes over a finite field F_{2^6}. The code has a monic degree-10 irreducible polynomial $g(x) = \alpha^{56} + \alpha^{29}x + \alpha^{55}x^2 + \alpha^{23}x^3 + \alpha^{49}x^4 + \alpha^{11}x^5 + \alpha^{59}x^6 + \alpha^{55}x^7 + \alpha^{12}x^8 + \alpha^{22}x^9 + x^{10}$ and a subset $L = \{\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{64}\} = \{0, 1, \alpha, \cdots, \alpha^{62}\}$. Suppose we have $r = (1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1)$ with 11 errors. We will use Algorithm 1 to find the error location polynomial $\epsilon(x)$.

First, $h(x)$ is defined as follows:

$$h(x) = \prod_i (x - \alpha_i) = x + x^{64}$$

1) First we compute the polynomial $s(x)$.

$$\begin{aligned} s(x) &= \sqrt{(1/\sum_i r_i/(x-\alpha_i)) - x} \\ &= \alpha^{50} + \alpha^{34}x^2 + \alpha^{36}x^3 + \alpha^{13}x^4 + \alpha^{59}x^5 + \alpha^3x^6 \\ &\quad + \alpha^{23}x^7 + \alpha^{37}x^8 + \alpha x^9. \end{aligned}$$

2) Then we find the minimum-length nonzero vector $(a_0(x), b_0(x))$ and the minimum-length independent vector $(a_1(x), b_1(x))$.

$$\begin{bmatrix} g(x) & 0 \\ s(x) & 1 \end{bmatrix} \rightarrow \begin{bmatrix} \alpha^{29} + \ldots + \alpha^{29}x^5 & \alpha^{13} + \ldots + \alpha^{20}x^4 \\ \alpha^{51}x + \ldots + \alpha^{34}x^4 & \alpha^{27} + \ldots + \alpha^{34}x^5 \end{bmatrix}$$

Thus we have

$$\begin{aligned} a_0(x) &= \alpha^{29} + \alpha^{35}x + \alpha^{47}x^2 + \alpha^{49}x^3 + \alpha^{17}x^4 + \alpha^{29}x^5 \\ b_0(x) &= \alpha^{13} + \alpha^{34}x + \alpha^{30}x^2 + \alpha^{47}x^3 + \alpha^{20}x^4 \\ a_1(x) &= \alpha^{51}x + \alpha^{26}x^2 + \alpha^{33}x^3 + \alpha^{34}x^4 \\ b_1(x) &= \alpha^{27} + \alpha^{48}x + \alpha^{46}x^2 + \alpha^{53}x^3 + \alpha^{37}x^4 + \alpha^{34}x^5. \end{aligned}$$

3) We calculate $\epsilon_0(x)$ such that

$$\begin{aligned} \epsilon_0(x) &= a_0^2(x) + xb_0^2(x) \\ &= a^{58} + a^{26}x + a^7x^2 + a^5x^3 + a^{31}x^4 + a^{60}x^5 \\ &\quad + a^{35}x^6 + a^{31}x^7 + a^{34}x^8 + a^{40}x^9 + a^{58}x^{10}. \end{aligned}$$

So we can find ϵ_1 by following the next three steps:

- We choose $\gamma = 1$.
- We compute $gcd\{a_1^2(x) + xb_1^2(x) + \gamma\epsilon_0(x), h(x)\}$.
- Since the result of the previous step is 1, we thus put

$$\begin{aligned} \epsilon_1(x) &= a_1^2(x) + xb_1^2(x) + \gamma\epsilon_0(x) \\ &= a^{58} + a^{58}x + a^{35}x^2 + \ldots + a^{58}x^{10} + a^5x^{11}. \end{aligned}$$

4) We compute the polynomial $\delta(x)$ such that

$$\begin{aligned} \delta(x) &= \frac{\epsilon_0(x)}{\epsilon_1(x)} (\text{mod } h(x)) \\ &= 1 + \alpha^{35}x + \alpha^{10}x^2 + \ldots + \alpha^{44}x^{62} + \alpha^{21}x^{63}. \end{aligned}$$

5) We define an l-deimensional lattice Ξ by following the next three steps:

- Calculate the integer θ.
  - $t_0 = \deg \epsilon_0(x) = 10$

$$
M = \left[\begin{array}{ccccccccccccccc|c}
h^4(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
h^3(x)\delta(x) & h^3(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
h^2(x)\delta^2(x) & 0 & h^2(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
h(x)\delta^3(x) & h(x)\delta^2(x) & h(x)\delta(x) & h(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\
\delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\
0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \\
0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \\
0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\
0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 9 \\
0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 10 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 11 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 12 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 13 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 14
\end{array}\right. \rightarrow
$$

$$
\begin{bmatrix}
\alpha^{7}+\ldots+\alpha^{51}x^{35} & \alpha^{7}+\ldots+\alpha^{53}x^{35} & \alpha^{47}+\ldots+\alpha^{8}x^{37} & \alpha^{47}+\ldots+\alpha^{10}x^{37} & \alpha^{38}+\ldots+\alpha x^{36} & \cdots & \alpha^{25}+\ldots+\alpha^{19}x^{36} & \alpha^{38}+\ldots+\alpha^{13}x^{26} \\
\alpha^{38}+\ldots+\alpha^{23}x^{34} & \alpha^{41}+\ldots+\alpha^{3}x^{34} & \alpha^{38}+\ldots+\alpha^{43}x^{36} & \alpha^{21}+\ldots+\alpha^{23}x^{36} & \alpha^{23}+\ldots+\alpha^{5}x^{36} & \cdots & \alpha^{23}+\ldots+\alpha^{56}x^{36} & \alpha^{59}+\ldots+x^{36} \\
\alpha^{46}+\ldots+\alpha^{56}x^{34} & \alpha^{44}+\ldots+\alpha^{10}x^{34} & \alpha^{41x}+\ldots+\alpha^{13}x^{36} & \alpha^{3}+\ldots+\alpha^{30}x^{36} & \alpha^{59}+\ldots+\alpha^{18}x^{36} & \cdots & \alpha^{27}+\ldots+\alpha^{38}x^{35} & \alpha^{45}+\ldots+\alpha^{33}x^{32} \\
\alpha^{17}+\ldots+\alpha^{20}x^{34} & \alpha^{46}+\ldots+\alpha^{61}x^{34} & \alpha^{22}+\ldots+\alpha^{40}x^{36} & \alpha^{19}+\ldots+\alpha^{18}x^{36} & \alpha^{12}+\ldots+\alpha^{34}x^{36} & \cdots & \alpha^{35}+\ldots+\alpha^{50}x^{35} & \alpha^{14}+\ldots+\alpha^{51}x^{30} \\
\alpha^{37x}+\ldots+\alpha^{26}x^{34} & \alpha^{53}+\ldots+\alpha^{28}x^{34} & \alpha^{7}+\ldots+\alpha^{46}x^{36} & \alpha^{32}+\ldots+\alpha^{48}x^{36} & \alpha^{14}+\ldots+\alpha^{21}x^{36} & \cdots & \alpha^{54}+\ldots+\alpha^{37}x^{36} & \alpha^{32}+\ldots+\alpha^{44}x^{27} \\
\alpha^{19}+\ldots+\alpha^{59}x^{34} & \alpha^{52}+\ldots+\alpha^{39}x^{34} & \alpha^{48}+\ldots+\alpha^{16}x^{36} & \alpha^{33}+\ldots+\alpha^{59}x^{36} & \alpha^{28}+\ldots+\alpha^{34}x^{36} & \cdots & \alpha^{13}+\ldots+\alpha^{12}x^{35} & \alpha^{29}+\ldots+\alpha^{22}x^{33} \\
\alpha^{31}+\ldots+\alpha^{8}x^{34} & \alpha^{42}+\ldots+\alpha^{46}x^{34} & \alpha^{61}+\ldots+\alpha^{28}x^{36} & \alpha^{57}+\ldots+\alpha^{3}x^{36} & \alpha+\ldots+\alpha^{42}x^{36} & \cdots & \alpha^{10}+\ldots+\alpha^{23}x^{35} & \alpha^{54}+\ldots+\alpha^{18}x^{31} \\
\alpha^{14}+\ldots+\alpha^{6}x^{35} & \alpha^{9}+\ldots+\alpha^{4}x^{35} & \alpha^{58}+\ldots+\alpha^{26}x^{37} & \alpha^{35}+\ldots+\alpha^{24}x^{37} & \alpha^{62}+\ldots+\alpha^{61}x^{37} & \cdots & \alpha^{46}+\ldots+\alpha^{49}x^{36} & \alpha^{21}+\ldots+\alpha^{4}x^{25} \\
\alpha^{59}+\ldots+\alpha^{14}x^{34} & \alpha^{40x}+\ldots+\alpha^{11}x^{34} & \alpha^{9}+\ldots+\alpha^{34}x^{36} & \alpha^{12}+\ldots+\alpha^{31}x^{36} & \alpha^{13}+\ldots+\alpha^{27}x^{36} & \cdots & \alpha^{32}+\ldots+\alpha^{14}x^{35} & \alpha^{48}+\ldots+\alpha x^{28} \\
\alpha^{48}+\ldots+\alpha^{27}x^{42} & \alpha^{29}+\ldots+\alpha^{52}x^{41} & \alpha^{22}+\ldots+\alpha^{47}x^{44} & \alpha^{2}+\ldots+\alpha^{9}x^{43} & \alpha^{53}+\ldots+\alpha^{32}x^{43} & \cdots & 0 & 0 \\
\alpha^{46}+\ldots+\alpha^{32}x^{34} & \alpha^{40}+\ldots+\alpha^{33}x^{34} & \alpha^{28}+\ldots+\alpha^{52}x^{36} & \alpha^{26}+\ldots+\alpha^{53}x^{36} & \alpha^{25}+\ldots+\alpha^{7}x^{36} & \cdots & 1+\ldots+\alpha^{13}x^{35} & \alpha^{47}+\ldots+\alpha^{44}x^{35} \\
\alpha^{25}+\ldots+\alpha^{47}x^{34} & \alpha^{3}+\ldots+\alpha^{57}x^{34} & \alpha^{33}+\ldots+\alpha^{4}x^{36} & \alpha^{26}+\ldots+\alpha^{14}x^{36} & \alpha^{55}+\ldots+\alpha^{59}x^{36} & \cdots & \alpha^{14}+\ldots+\alpha^{33}x^{35} & 1+\ldots+\alpha^{36}x^{34} \\
\alpha^{35}+\ldots+\alpha^{36}x^{34} & \alpha^{6}+\ldots+\alpha^{36}x^{34} & \alpha^{60}+\ldots+\alpha^{56}x^{36} & \alpha^{55}+\ldots+\alpha^{56}x^{36} & \alpha^{6}+\ldots+\alpha^{10}x^{36} & \cdots & \alpha^{19}+\ldots+\alpha^{37}x^{35} & \alpha^{13}+\ldots+x^{29} \\
\alpha^{28}+\ldots+\alpha^{62}x^{59} & \alpha^{25}+\ldots+\alpha^{9}x^{60} & \alpha^{3}+\ldots+\alpha^{51}x^{59} & \alpha^{15}+\ldots+\alpha^{13}x^{59} & \alpha+\ldots+\alpha^{17}x^{59} & \cdots & \alpha^{24}+\ldots+\alpha^{3}x^{59} & \alpha^{47}+\ldots+\alpha^{29}x^{21} \\
\alpha^{60}+\ldots+\alpha^{18}x^{102} & \alpha^{57}+\ldots+\alpha^{49}x^{101} & \alpha^{35}+\ldots+x^{101} & \alpha^{41}+\ldots+\alpha^{46}x^{101} & \alpha^{33}+\ldots+\alpha^{24}x^{101} & \cdots & \alpha^{42}+\ldots+\alpha^{56}x^{101} & \alpha^{8}+\ldots+\alpha^{5}x^{10}
\end{bmatrix}
$$

FIGURE 1. Convert *M* to the weak Popov form.

$$
M' = \begin{array}{c}\Psi_2 \\ \\ \Psi_1 \\ \\ \Psi_0\end{array}
\left[\begin{array}{ccccccccccccccc|c}
h^4(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 12 \\
\hline
h^2(x)\delta^2(x) & 0 & h^2(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 10 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 14 \\
\hline
h^3(x)\delta(x) & h^3(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 13 \\
h(x)\delta^3(x) & h(x)\delta^2(x) & h(x)\delta(x) & h(x) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\
0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta^4(x) & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 11
\end{array}\right. \rightarrow
$$

$$
\begin{bmatrix}
\alpha^{14}+\ldots+\alpha^{18}x^{34} & \alpha^{2}+\ldots+\alpha^{52}x^{34} & \alpha^{27}+\ldots+\alpha^{38}x^{36} & \alpha^{34}+\ldots+\alpha^{9}x^{36} & \alpha^{4}+\ldots+\alpha^{38}x^{36} & \cdots & \alpha^{50}+\ldots+\alpha^{32}x^{35} & \alpha^{56}+\ldots+\alpha^{43}x^{35} \\
\alpha^{41}+\ldots+\alpha^{40}x^{35} & \alpha^{25}+\ldots+\alpha^{59}x^{35} & \alpha^{36}+\ldots+\alpha^{60}x^{37} & \alpha^{30}+\ldots+\alpha^{16}x^{37} & \alpha^{55}+\ldots+\alpha^{40}x^{37} & \cdots & \alpha^{41}+\ldots+\alpha^{31}x^{36} & \alpha^{26}+\ldots+\alpha^{39}x^{36} \\
\alpha^{61}+\ldots+\alpha^{59}x^{42} & 0 & \alpha^{49}+\ldots+\alpha^{16}x^{44} & 0 & \alpha+\ldots+\alpha^{34}x^{42} & \cdots & 0 & 0 \\
\alpha^{16}+\ldots+\alpha^{54}x^{34} & \alpha^{10}+\ldots+\alpha^{21}x^{34} & \alpha^{20}+\ldots+\alpha^{11}x^{36} & \alpha^{4}+\ldots+\alpha^{41}x^{36} & \alpha^{62}+\ldots+\alpha^{18}x^{36} & \cdots & \alpha^{18}+\ldots+\alpha^{45}x^{35} & \alpha^{47}+\ldots+\alpha^{34}x^{35} \\
\alpha^{36}+\ldots+\alpha^{53}x^{34} & \alpha^{42}+\ldots+\alpha^{22}x^{34} & \alpha^{41}+\ldots+\alpha^{52}x^{35} & \alpha^{58}+\ldots+\alpha x^{35} & \alpha^{7}+\ldots+\alpha^{16}x^{35} & \cdots & \alpha^{52}+\ldots+\alpha^{57}x^{35} & \alpha^{54}+\ldots+\alpha^{55}x^{35} \\
\alpha^{11}+\ldots+\alpha^{17}x^{34} & \alpha^{33}+\ldots+\alpha^{24}x^{34} & \alpha^{48}+\ldots+\alpha^{37}x^{36} & \alpha+\ldots+\alpha^{44}x^{36} & \alpha^{56}+\ldots+\alpha^{29}x^{36} & \cdots & \alpha^{60}+\ldots+\alpha^{28}x^{35} & \alpha^{30}+\ldots+\alpha^{24}x^{35} \\
\alpha^{57}+\ldots+\alpha^{59}x^{34} & \alpha^{16}+\ldots+\alpha^{54}x^{34} & \alpha^{5}+\ldots+\alpha^{16}x^{36} & \alpha^{40}+\ldots+\alpha^{11}x^{36} & \alpha^{23}+\ldots+\alpha^{44}x^{36} & \cdots & \alpha^{24}+\ldots+\alpha^{8}x^{35} & \alpha^{37}+\ldots+\alpha^{47}x^{35} \\
\alpha^{44}+\ldots+\alpha^{38}x^{34} & \alpha^{37}+\ldots+\alpha^{26}x^{34} & \alpha^{16}+\ldots+\alpha^{58}x^{36} & \alpha^{53}+\ldots+\alpha^{46}x^{36} & \alpha^{19}+\ldots+\alpha^{49}x^{36} & \cdots & \alpha^{55}+\ldots+\alpha^{34}x^{35} & \alpha^{28}+\ldots+\alpha^{25}x^{35} \\
\alpha^{56}+\ldots+\alpha^{9}x^{35} & \alpha^{53}+\ldots+\alpha^{11}x^{35} & \alpha^{59}+\ldots+\alpha^{29}x^{37} & \alpha^{3}+\ldots+\alpha^{31}x^{37} & \alpha^{17}+\ldots+\alpha^{30}x^{36} & \cdots & \alpha^{62}+\ldots+\alpha^{20}x^{36} & \alpha^{36}+\ldots+\alpha^{53}x^{36} \\
1+\ldots+\alpha^{49}x^{34} & \alpha^{57}+\ldots+\alpha^{50}x^{34} & \alpha^{45}+\ldots+\alpha^{6}x^{36} & \alpha^{43}+\ldots+\alpha^{7}x^{36} & \alpha^{42}+\ldots+\alpha^{24}x^{36} & \cdots & \alpha^{17}+\ldots+\alpha^{30}x^{35} & a+\ldots+\alpha^{61}x^{35} \\
\alpha^{21}+\ldots+\alpha^{47}x^{34} & \alpha^{50}+\ldots+\alpha^{29}x^{34} & \alpha^{55}+\ldots+\alpha^{4}x^{36} & \alpha^{20}+\ldots+\alpha^{49}x^{36} & \alpha^{22}+\ldots+\alpha^{58}x^{36} & \cdots & \alpha^{35}+\ldots+\alpha^{56}x^{35} & \alpha^{24}+\ldots+\alpha^{45}x^{35} \\
\alpha^{58}+\ldots+x^{34} & \alpha^{12}+\ldots+\alpha^{27}x^{34} & \alpha^{46}+\ldots+\alpha^{20}x^{36} & 1+\ldots+\alpha^{47}x^{36} & \alpha^{2}+\ldots+\alpha^{10}x^{36} & \cdots & \alpha^{17}+\ldots+\alpha^{49}x^{36} & \alpha^{21}+\ldots+\alpha^{34}x^{35} \\
\alpha^{11}+\ldots+\alpha^{36}x^{102} & 0 & \alpha^{15}x^{58}+\ldots+\alpha^{2}x^{100} & 0 & \alpha^{26}+\ldots+\alpha^{15}x^{100} & \cdots & 0 & \alpha^{10}+\ldots+\alpha^{11}x^{20}
\end{bmatrix}
$$

FIGURE 2. Convert *M*′ to the weak Popov form.

- $g_0 = 2\lfloor (u+t-t_0)/2 \rfloor = 0$
- $g_1 = 2\lfloor (u+t_0-t-1)/2 \rfloor = 0$
- $\theta = g_1 - g_0 = 0$

- We choose $d = 4$, and $l = 15$.
- The polynomials that generate a 15-dimensional lattice are as follows:

$$
1, \frac{z+\delta(x)}{h(x)}, \frac{z^2+\delta^2(x)}{h^2(x)}, \frac{z^3+\delta(x)z^2+\delta^2(x)z+\delta^3(x)}{h^3(x)},
$$
$$
\frac{z^4+\delta^4(x)}{h^4(x)}, \frac{z^5+\delta^4(x)z}{h^4(x)}, \frac{z^6+\delta^4(x)z^2}{h^4(x)}, \ldots,
$$

$$\frac{z^{12}+\delta^4(x)z^8}{h^4(x)}, \frac{z^{13}+\delta^4(x)z^9}{h^4(x)}, \frac{z^{14}+\delta^4(x)z^{10}}{h^4(x)}.$$

6) We generate the polynomial matrix M using coefficients of $1, z, z^2, \cdots, z^{14}$ in the generators.
And then we scale the entire M such that

$$M = h^4(x) \times M.$$

7) Figure 1 shows the conversion of M to the weak Popov form.
8) We find the minimum-length nonzero vector $\phi(z) = \phi_0 + \phi_1 z + \ldots + \phi_{14}z^{14}$. There are several row vectors that can be $\phi(z)$. We chose the 2nd-row vector as $\phi(z)$, so

$$\phi(z) = (\alpha^{38} + \ldots + \alpha^{23}x^{34}) + (\alpha^{41} + \ldots + \alpha^3 x^{34})z + \ldots + (\alpha^{59} + \ldots + x^{36})z^{14}.$$

9) Since $\phi(\alpha^{61}) = 0$ and $\phi((\alpha^{53} + \ldots + \alpha^{53}x^{10})/(\alpha^{53} + \ldots + x^{11})) = 0$, the root that satisfies the $q_1^2(x)/x^\theta q_0^2(x)$ form is α^{61}. Therefore, it is

$$\begin{cases} q_0^2(x) = 1 \\ q_1^2(x) = \alpha^{61}. \end{cases}$$

10) Finally, we find the error locator polynomial $\epsilon(x)$ with

$$\begin{aligned}\epsilon(x) &= q_0^2(x)\epsilon_0(x) + q_1^2(x)\epsilon_1(x) \\ &= (\alpha^{58} + \alpha^{26}x + \ldots + \alpha^{58}x^{10}) \\ &\quad + \alpha^{61}(\alpha^{58} + \alpha^{58}x + \ldots + \alpha^5 x^{11}) \\ &= \alpha^{42} + \alpha^{31}x + \ldots + \alpha^{42}x^{10} + \alpha^3 x^{11}.\end{aligned}$$

The roots of $\epsilon(x)$ are

$$(\alpha^3, \alpha^8, \alpha^{11}, \alpha^{14}, \alpha^{21}, \alpha^{25}, \alpha^{28}, \alpha^{34}, \alpha^{38}, \alpha^{51}, \alpha^{58})$$
$$= (\alpha_4, \alpha_9, \alpha_{12}, \alpha_{15}, \alpha_{22}, \alpha_{26}, \alpha_{29}, \alpha_{35}, \alpha_{39}, \alpha_{52}, \alpha_{59}).$$

Thus, the error positions of r are

$$\{4, 9, 12, 15, 22, 26, 29, 35, 39, 52, 59\}.$$

APPENDIX B
EXAMPLE OF *IDX* GENERATION AND ROW VECTOR REARRANGEMENT

Let us new rearrange M in the example in Appendix A and convert it to the weak Popov form. First, we use Algorithm 3 to generate *IDX* to be used as a reference for row vector rearrangement.

1) We classify the row vectors from 0th to 3rd as follows:

$$\Psi_2 = \{0\},$$
$$\Psi_1 = \{2\},$$
$$\Psi_0 = \{1, 3\}.$$

2) We identify row vectors that could potentially share the same pivot indices as the pivot indices of the modified pivots of the previously classified row vectors.

$$\Psi_2 = \{0, 4, 8, 12\},$$
$$\Psi_1 = \{2, 6, 10, 14\},$$
$$\Psi_0 = \{1, 5, 9, 13, 3, 7, 11\}.$$

We generate *IDX* based on the created Ψ_i.

$$\begin{aligned} IDX &= (idx_0, idx_1, \cdots, idx_{l-1}) \\ &= (0, 4, 8, 12, 2, 6, 10, 14, 1, 5, 9, 13, 3, 7, 11). \end{aligned}$$

We rearrange the row vectors of M according to the generated *IDX*. The rearranged M is denoted as M'. We then convert M' to the weak Popov form.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \frac{\delta(x)}{h(x)} & \frac{1}{h(x)} & 0 & 0 & 0 & 0 & \cdots & 0 \\ \frac{\delta^2(x)}{h^2(x)} & 0 & \frac{1}{h^2(x)} & 0 & 0 & 0 & \cdots & 0 \\ \frac{\delta^3(x)}{h^3(x)} & \frac{\delta^2(x)}{h^3(x)} & \frac{\delta(x)}{h^3(x)} & \frac{1}{h^3(x)} & 0 & 0 & \cdots & 0 \\ \frac{\delta^4(x)}{h^4(x)} & 0 & 0 & 0 & \frac{1}{h^4(x)} & 0 & \cdots & 0 \\ 0 & \frac{\delta^4(x)}{h^4(x)} & 0 & 0 & 0 & \frac{1}{h^4(x)} & \cdots & 0 \\ 0 & 0 & \frac{\delta^4(x)}{h^4(x)} & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \frac{\delta^4(x)}{h^4(x)} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \frac{\delta^4(x)}{h^4(x)} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \frac{1}{h^4(x)} \end{bmatrix}$$

We calculate the degree difference of each row vector to find the minimum length non-zero vector $\phi(z)$. Among the several row vectors that can be $\phi(z)$, the first-row vector was selected as $\phi(z)$.

$$\phi(z) = (\alpha^{14} + \ldots + \alpha^{18}x^{34}) + (\alpha^2 + \ldots + \alpha^{52}x^{34})z + \ldots + (\alpha^{56} + \ldots + \alpha^{43}x^{35})z^{14}.$$

It can be observed that the roots of $\phi(z)$ coincide with the findings presented in Appendix A, as follows:

$$\begin{cases} \phi(\alpha^{61}) = 0 \\ \phi((\alpha^{53} + \ldots + \alpha^{53}x^{10})/(\alpha^{53} + \ldots + x^{11})) = 0. \end{cases}$$

Therefore, this also holds:

$$\begin{cases} q_0^2(x) = 1 \\ q_1^2(x) = \alpha^{61}. \end{cases}$$

REFERENCES

[1] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *DSN Prog. Rep.*, vol. 44, pp. 114–116, Jan. 1978.

[2] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems Control Inf. Theory*, vol. 15, no. 2, pp. 159–166, 1986.

[3] M. Albrecht, D. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang, "Classic McEliece: Conservative code-based cryptography," NIST's Post-Quantum Cryptogr. Standardization Project, USA, 2022. [Online]. Available: https://classic.mceliece.org/nist/mceliece-submission-20221023.pdf

[4] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani, "Reducing key length of the McEliece cryptosystem," in *Progress in Cryptology—AFRICACRYPT 2009*. Berlin, Germany: Springer, 2009, pp. 77–97.

[5] D. J. Bernstein, "List decoding for binary Goppa codes," in *Proc. Int. Workshop Coding Cryptol.*, vol. 6639, Qingdao, China, Y. M. Chee, Ed. Berlin, Germany: Springer, 2011, pp. 62–80.

[6] L. Mariot, S. Picek, and R. Yorgova, "On McEliece-type cryptosystems using self-dual codes with large minimum weight," *IEEE Access*, vol. 11, pp. 43511–43519, 2023.

[7] V. M. Sidelnikov, "A public-key cryptosystem based on binary Reed–Muller codes," *Discrete Math. Appl.*, vol. 4, no. 3, pp. 191–207, 1994.

[8] R. Misoczki and P. Barreto, "Compact McEliece keys from Goppa codes," in *Selected Areas in Cryptography 2009*, vol. 5867. Berlin, Germany: Springer, Aug. 2009, pp. 376–392.

[9] D. J. Bernstein, T. Lange, and C. Peters, "Wild McEliece," in *Selected Areas in Cryptography*, vol. 6544. Berlin, Germany: Springer, 2010, pp. 143–158.

[10] C. Balamurugan, K. Singh, G. Ganesan, and M. Rajarajan, "Post-quantum and code-based cryptography—Some prospective research directions," *Cryptography*, vol. 5, no. 4, p. 38, Dec. 2021.

[11] N. Patterson, "The algebraic decoding of Goppa codes," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 2, pp. 203–207, Mar. 1975.

[12] P. Beelen, T. Høholdt, J. S. R. Nielsen, and Y. Wu, "On rational interpolation-based list-decoding and list-decoding binary Goppa codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 6, pp. 3269–3281, Jun. 2013.

[13] D. Augot, M. Barbier, and A. Couvreur, "List-decoding of binary Goppa codes up to the binary Johnson bound," in *Proc. IEEE Inf. Theory Workshop*, Oct. 2011, pp. 229–233.

[14] P. J. Lee and E. F. Brickell, "An observation on the security of McEliece public-key cryptosystem," in *Proc. Workshop Theory Appl. Crypto. Tech.*, Davos, Switzerland, May 1988, pp. 275–280.

[15] S. Sarkar, "Computing Popov forms of polynomial matrices," Master's thesis, Dept. Comput. Sci., Univ. Waterloo, Waterloo, ON, Canada, 2011.

[16] T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices," *J. Symbolic Comput.*, vol. 35, no. 4, pp. 377–401, Apr. 2003.

[17] K. Thomas, *Linear Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1980, pp. 475–494.

[18] K. Lee and M. E. O'Sullivan, "List decoding of Hermitian codes using Gröbner bases," *J. Symbolic Comput.*, vol. 44, no. 12, pp. 1662–1675, Dec. 2009.

[19] M. Alekhnovich, "Linear diophantine equations over polynomials and soft decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2257–2265, Jul. 2005.

[20] V. Neiger and T. X. Xuan, "Computing canonical bases of modules of univariate relations," in *Proc. ISSAC*, 2017, pp. 357–364.

[21] S. Puchinger, J. Rosenkilde né Nielsen, W. Li, and V. Sidorenko, "Row reduction applied to decoding of rank-metric and subspace codes," *Des., Codes Cryptogr.*, vol. 82, nos. 1–2, pp. 389–409, Jan. 2017.

[22] V. Neiger, J. Rosenkilde, and G. Solomatov, "Computing Popov and Hermite forms of rectangular polynomial matrices," in *Proc. ISSAC*, 2018, pp. 295–302.

[23] E. Jochemsz, "Goppa codes & the McEliece cryptosystem," Master's thesis, Division Math. Comput. Sci., Vrije Universiteit, Amsterdam, The Netherlands, 2002.

KI-SOON YU received the M.S. degree from the Department of Information Security, Dongguk University, Seoul, Republic of Korea, in 2013. Her research interests include security, cryptography, and algebraic coding theory.

DAE-WOON LIM received the B.S. and M.S. degrees from the Department of Electrical Engineering, KAIST, Daejeon, South Korea, in 1994 and 1997, respectively, and the Ph.D. degree in electrical engineering and computer science from Seoul National University, in 2006. From 1997 to 2002, he was with LG Industrial Systems as a Senior Research Engineer. He is currently a Professor with the Department of Information and Communication Engineering, Dongguk University, Seoul, Republic of Korea. His research interests include signal processing, wireless communications, cryptography, and security.