A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding

Hanxu Hou, *Member, IEEE,* Yunghsiang S. Han, *Fellow, IEEE,* Kenneth W. Shum, *Senior Member, IEEE* and Hui Li, *Member, IEEE*

***Abstract*— Array codes are used widely in data storage systems such as RAID (Redundant Array of Independent Disks). The row-diagonal parity (RDP) codes and EVENODD codes are two popular double-parity array codes. The increasing capacity of hard disks demands better fault tolerance by using array codes with three or more parity disks. Although many extensions of RDP and EVENODD codes have been proposed, their main drawback is high decoding complexity. In this paper, we propose a unified form of RDP and EVENODD codes under which RDP codes can be treated as shortened EVENODD codes. Moreover, an efficient decoding algorithm based on an LU factorization of a Vandermonde matrix is proposed. The LU decoding method is applicable to all the erasure patterns of RDP and EVENODD codes with three parity columns. It is also applicable to the erasure decoding of RDP and EVENODD codes with more than three parity columns when the number of continuous surviving parity columns is no less than the number of erased information columns and the first parity column has not failed. The proposed efficient decoding algorithm is also applicable to other Vandermonde array codes, with less decoding complexity than that of the existing method.**

***Index Terms*—RAID, array codes, EVENODD, RDP, efficient decoding, LU factorization.**

I. Introduction

Array codes are used widely in data storage systems such as RAID (Redundant Array of Independent Disks) [1], [2] to enhance data reliability. In the current RAID-6 system, two disks are dedicated to storing parity-check bits, meaning that the failure of any two disks can be tolerated. There have been many previous studies of designing array codes that can recover the failure of any two disks, such as the EVENODD codes [3] and the row-diagonal parity (RDP) codes [4].

Because the capacities of hard disks are increasing much faster than bit error rates are decreasing, the protection offered by double parities will soon be inadequate [5]. The issue of reliability is more pronounced in solid-state drives, which have significant wear-out rates when the frequencies of disk writes are high. To tolerate three or more disk failures, the EVENODD codes were extended in [6], [7] and the RDP codes were extended in [8], [9]. All of the above coding methods are binary array codes, whose codewords are $m \times n$ arrays for some positive integers m and n, with each entry belonging to the binary field $\mathbb{F}_2$. Binary array codes enjoy the advantage that encoding and decoding can be done by Exclusive OR (XOR) operations. The n disks are identified as n columns, and the m bits in each column are stored in the corresponding disk. A binary array code is said to be *systematic* if, for some positive integer r less than n, the right-most r columns are called *parity columns* that store the parity bits, while the left-most $k = n-r$ columns are called *information columns* that store the uncoded data bits. If the array code can tolerate r arbitrary erasures, then it is called a maximum-distance separable (MDS) array code. In other words, in an MDS array code, the information bits can be recovered from any k columns.

A. Related Work

There have been many follow-up studies on EVENODD codes [3] and RDP codes [4] in different directions, such as (i) extending their fault tolerances [6], [7], [8], (ii) improving the repair problem [10], [11], [12], [13], and (iii) efficient decoding methods [14], [15], [16], [17] for their extensions.

Huang and Xu [14] extended the EVENODD codes to be STAR codes with three parity columns. Blaum *et al.* [6], [7] extended the EVENODD codes for three or more parity columns. A sufficient condition for the extended EVENODD codes to be MDS with more than eight parity columns is given in [18]. Goel and Corbett [8] proposed the RTP codes that extend the RDP codes to tolerate three disk failures. Blaum [9] generalized the RDP codes to ones that can correct more than three column erasures and showed that the extended EVENODD codes and generalized RDP codes share the same condition of MDS property. Blaum and Roth [19] proposed the Blaum–Roth codes, which are constructed over a Vandermonde matrix. Some efficient systematic encoding methods for Blaum–Roth codes are given in [19], [20], [21]. We refer to the existing MDS array codes in [3], [4], [6], [7], [8], [9], [14], [15], [16], [19] as Vandermonde MDS array codes because their constructions are based on Vandermonde matrices.

Herein, we define *decoding complexity* as the number of XORs required to recover up to r erased columns (including

Hanxu Hou is with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology and the Shenzhen Key Lab of Information Theory & Future Internet Architecture, Future Network PKU Lab of National Major Research Infrastructure, Peking University Shenzhen Graduate School (E-mail: houhanxu@163.com), Yunghsiang S. Han is with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology (E-mail: yunghsiangh@gmail.com). Kenneth W. Shum is with the Institute of Network Coding, The Chinese University of Hong Kong (E-mail: wkshum@inc.cuhk.edu.hk). Hui Li is with the Shenzhen Key Lab of Information Theory & Future Internet Architecture, Future Network PKU Lab of National Major Research Infrastructure, Peking University Shenzhen Graduate School (E-mail: lih64@pkusz.edu.cn). This work was partially supported by the National Natural Science Foundation of China (No. 61701115, 61671007, 61671001), the Research Grants Council of the Hong Kong Special Administrative Region, China (No. 2150874), National Keystone R&D Program of China (No. 2017YFB0803204, 2016YFB0800101) and Shenzhen Research Program (No. ZDSYS201603311739428).

information erasure and parity erasure) from k surviving columns. There are many decoding methods for extended EVENODD codes [15] and generalized RDP codes; however, most of them focus on $r = 3$. Jiang *et al.* [15] proposed a decoding algorithm for extended EVENODD codes with $r = 3$. To further reduce the decoding complexity of the extended EVENODD codes with $r = 3$, Huang and Xu [14] invented STAR codes. One extension of RDP codes with three parity columns is RTP codes, whose decoding has been improved by Huang *et al.* [17]. Two efficient interpolation-based encoding algorithms for Blaum–Roth codes were proposed in [20], [21]. However, the efficient algorithms in [20], [21] are not applicable to the decoding of the extended EVENODD codes and generalized RDP codes. An efficient erasure decoding method that solves a Vandermonde linear system over a polynomial ring was given in [19] for Blaum–Roth codes, and the decoding method is also applicable to the erasure decoding of extended EVENODD codes if the number of information erasures does not exceed the number of continuous surviving parity columns. There is no efficient decoding method for arbitrary erasures; instead, one must use a traditional decoding method such as Cramer's rule to recover the erased columns.

B. Contributions

Herein, we present a unified form of EVENODD and RDP codes including the existing RDP codes and their extensions in [4], [9] and the existing EVENODD codes and their extensions in [6], [7]. Under this unified form, these two families of codes are shown to be closely related. Based on this unified form, we also propose a fast method for recovering failed columns. This method is based on factorizing a Vandermonde matrix into very sparse lower and upper triangular matrices. We then illustrate the methodology by applying it to EVENODD codes and RDP codes. The proposed fast decoding method can recover up to r erasures for EVENODD and RDP codes such that the number of information erasures does not exceed the number of continuous surviving parity columns and the first parity column has not failed. When $r = 3$, our decoding method can recover all erasure patterns for all existing Vandermonde MDS array codes. We compare the proposed decoding method with the decoding method given in [19] when both are applied to EVENODD and RDP codes. We show that the proposed method has lower decoding complexity than that of the decoding algorithm given in [19]. We also show that the decoding method requires less decoding complexity compared with the existing decoding methods given in [14], [15] for EVENODD codes with three parity columns. Moreover, our decoding method can recover all erasure patterns while decoding Blaum–Roth codes [19] and with less decoding complexity compared with the existing methods.

Note that EVENODD codes [6], [7] can be described over $\mathbb{F}_2[x]/(1 + x + \cdots + x^{p-1})$ with a parity-check matrix composed of an identity matrix and a Vandermonde matrix. If the number of information erasures does not exceed the number of continuous surviving parity columns, then we can formulate the erased information columns as a Vandermonde linear system. Our decoding method is a fast method for solving this linear system. However, to further reduce the decoding complexity, we show that it is more efficient to solve the Vandermonde linear system over $\mathbb{F}_2[x]/(1+x^p)$. To convert EVENODD codes into the codes over $\mathbb{F}_2[x]/(1+x^p)$ (see Section II.B), the first parity column should not fail. Therefore, we divide our decoding method into two cases: (i) over $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$ and (ii) over $\mathbb{F}_2[x]/(1+x^p)$. Since the operation over $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$ has been widely studied and the decoding method of (ii) requires less complexity, we will focus on the decoding method of case (ii), and we assume that the number of information erasures does not exceed the number of continuous surviving parity columns and that the first parity column has not failed when we consider the erasure decoding.

Note that both the decoding method in [19] and our decoding method cannot recover all erasure patterns for EVENODD and RDP codes when $r \geq 4$. However, our decoding method can recover all erasure patterns for EVENODD and RDP codes when $r = 3$. In addition, only a small proportion[1] of the erasures cannot be recovered by the proposed decoding method when $r = 4$. The erasure patterns for which our decoding method is not applicable may be recovered by a traditional decoding method such as Cramer's rule.

II. Unified Form of EVENODD and RDP Codes

In this section, we first review the definitions of EVENODD with $k \leq p$ and RDP codes with $k \leq p-1$ and then give them in a unified form. The array codes considered herein contain $p-1$ rows and $k+r$ columns, where p is a prime number, the first k columns are information columns, and the last r columns are parity columns. The i-th entries of column j are denoted as $a_{i,j}$ and $b_{i,j}$ for EVENODD codes and RDP codes, respectively. The subscripts are taken modulo p throughout unless otherwise specified.

A. Review of EVENODD and RDP Codes

In an EVENODD code, we have $p \geq \max\{k, r\}$ and information bits $a_{i,j}$ for $i = 0, 1, \ldots, p-2$ and $j = 0, 1, \ldots, k-1$. Letting $a_{p-1,0} = \cdots = a_{p-1,k-1} = 0$, the parity bits in column k are computed by

$$a_{i,k} = \sum_{j=0}^{k-1} a_{i,j} \text{ for } 0 \leq i \leq p-2, \tag{1}$$

and the parity bits stored in column $k+\ell$, $\ell = 1, 2, \ldots, r-1$, are computed by

$$a_{i,k+\ell} = a_{p-1,k+\ell} + \sum_{j=0}^{k-1} a_{i-\ell\cdot j,j} \text{ for } 0 \leq i \leq p-2, \tag{2}$$

[1]When $r = 4$, the only undecodable erasure patterns of our decoding method are those with the second or third parity column and any three information columns erased. In total, there are $\binom{k+4}{1} + \binom{k+4}{2} + \binom{k+4}{3} + \binom{k+4}{4}$ possible erasure patterns, and the number of undecodable erasure patterns is $2\binom{k}{3}$. Therefore, the proportion of erasures that cannot be recovered is $2\binom{k}{3}/(\binom{k+4}{1} + \binom{k+4}{2} + \binom{k+4}{3} + \binom{k+4}{4})$ when $r = 4$. The proportion is 10.4% when $k = 6$.

where

$$a_{p-1,k+\ell} = \sum_{j=0}^{k-1} a_{p-1-\ell\cdot j,j}. \quad (3)$$

We denote the EVENODD codes defined in the above equations as $\mathsf{EVENODD}(p,k,r)$. Under the above definition, the EVENODD code in [3] is $\mathsf{EVENODD}(p,p,2)$, and the extended EVENODD code in [6], [7] is $\mathsf{EVENODD}(p,p,r)$.

In an RDP code, the parameters k, r, and p satisfy $p \geq \max(k+1,r)$. Let $b_{p-1,0} = \cdots = b_{p-1,k-1} = 0$, as in $\mathsf{EVENODD}(p,k,r)$. The parity bits of $\mathsf{RDP}(p,k,r)$ are computed as follows:

$$b_{i,k} = \sum_{j=0}^{k-1} b_{i,j} \text{ for } 0 \leq i \leq p-2, \quad (4)$$

$$b_{i,k+\ell} = \sum_{j=0}^{k} b_{i-\ell\cdot j,j} \text{ for } 0 \leq i \leq p-2, 1 \leq \ell \leq r-1. \quad (5)$$

The RDP code in [4] is $\mathsf{RDP}(p,p-1,2)$, and $\mathsf{RDP}(p,p-1,r)$ is the extended RDP in [9].

Note that the parity bits in column $k+i$ are computed along the straight line with slope i in the array of $\mathsf{EVENODD}(p,k,r)$ and $\mathsf{RDP}(p,k,r)$. We generalize $\mathsf{EVENODD}(p,k,r)$ and $\mathsf{RDP}(p,k,r)$ by computing the parity bits along some polygonal lines in the array in our technical report [22].

B. Unified Form

There is a close relationship between $\mathsf{RDP}(p,k,r)$ and $\mathsf{EVENODD}(p,k,r)$. The relationship can be seen by augmenting the arrays as follows. For RDP codes, we define the corresponding augmented array as a $p \times (k+r)$ array with the top $p-1$ rows the same as in $\mathsf{RDP}(p,k,r)$, and the last row defined by $b_{p-1,j} = 0$ for $0 \leq j \leq k$ and

$$b_{p-1,k+\ell} = \sum_{j=0}^{k} b_{p-1-\ell\cdot j,j} \text{ for } 1 \leq \ell \leq r-1. \quad (6)$$

Note that (6) is the extension of (5) when $i = p-1$. The auxiliary row in the augmented array is defined such that the column sums of columns $k+1, k+2, \ldots, k+r-1$ are equal to zero. The above claim is proved as follows.

Lemma 1. *For $1 \leq \ell \leq r-1$, we have $\sum_{i=0}^{p-1} b_{i,k+\ell} = 0$.*

Proof. The summation of all bits in column $k+\ell$ of the augmented array is the summation of all bits in columns 0 to k. Because the summation of all bits in column k is the summation of all bits in columns 0 to $k-1$, we have that the summation of all bits in column $k+\ell$ is zero. □

By Lemma 1, we can compute $b_{p-1,k+\ell}$ for $\ell = 1,2,\ldots,r-1$ as $b_{p-1,k+\ell} = \sum_{i=0}^{p-2} b_{i,k+\ell}$. An example of the augmented array code of $\mathsf{RDP}(5,3,3)$ is given in Table I.

Similarly, for $\mathsf{EVENODD}(p,k,r)$, the augmented array is a $p \times (k+r)$ array $[a'_{i,j}]$ defined as follows. The first $k+1$ columns are the same as those of $\mathsf{EVENODD}(p,k,r)$. For $\ell = 1,2,\ldots,r-1$, we define the parity bits in column $k+\ell$ as

$$a'_{i,k+\ell} = \sum_{j=0}^{k-1} a_{i-\ell\cdot j,j} \text{ for } 0 \leq i \leq p-1. \quad (7)$$

We note that $a'_{p-1,k+\ell}$ is the same as $a_{p-1,k+\ell}$ defined in (3). According to (2), the parity bits in column $k+\ell$ of $\mathsf{EVENODD}(p,k,r)$ can be obtained from the augmented array by

$$a_{i,k+\ell} = a'_{i,k+\ell} + a'_{p-1,k+\ell}.$$

Lemma 2. *The bits in column $k+\ell$ for $\ell = 1,2,\ldots,r-1$ of the augmented array can be obtained from $\mathsf{EVENODD}(p,k,r)$ by*

$$a'_{p-1,k+\ell} = \sum_{i=0}^{p-2}(a_{i,k} + a_{i,k+\ell}), \text{ and} \quad (8)$$

$$a'_{i,k+\ell} = a_{i,k+\ell} + a'_{p-1,k+\ell} \text{ for } i = 0,1,\ldots,p-2. \quad (9)$$

Proof. Note that

$$\begin{aligned}
&\sum_{i=0}^{p-2}(a_{i,k} + a_{i,k+\ell}) && (10)\\
=\ &\sum_{i=0}^{p-1}(\sum_{j=0}^{k-1} a_{i,j}) + \sum_{i=0}^{p-2}(a_{p-1,k+\ell} + \sum_{j=0}^{k-1} a_{i-\ell\cdot j,j}) && (11)\\
=\ &\sum_{i=0}^{p-1} a_{i,0} + \cdots + \sum_{i=0}^{p-1} a_{i,k-1} + \sum_{i=0}^{p-2} a_{i,0} + \cdots + \\
&\sum_{i=0}^{p-2} a_{i-\ell(k-1),k-1} + \underbrace{a_{p-1,k+\ell} + \cdots + a_{p-1,k+\ell}}_{p-1} \\
=\ &\sum_{i=0}^{p-1} a_{i,0} + \cdots + \sum_{i=0}^{p-1} a_{i-\ell(k-1),k-1} + \\
&\sum_{i=0}^{p-2} a_{i,0} + \cdots + \sum_{i=0}^{p-2} a_{i-\ell(k-1),k-1} && (12)\\
=\ &(a_{p-1,0} + a_{p-1-\ell,1} + \cdots + a_{p-1-\ell(k-1),k-1}) \\
=\ &a'_{p-1,k+\ell},
\end{aligned}$$

where (11) comes from (1), (2), and $a_{p-1,j} = 0$ for $j = 0,1,\ldots,k-1$, and (12) comes from

$$\{-\ell\cdot j, 1-\ell\cdot j, \ldots, p-1-\ell\cdot j\} = \{0,1,\ldots,p-1\} \bmod p$$

for $0 \leq j \leq k-1$. Therefore, we can obtain the bit $a'_{p-1,k+\ell}$ by (8) and the other bits in parity column $k+\ell$ by (9). □

The augmented array of $\mathsf{EVENODD}(5,3,3)$ is given in Table II. We summarize the relationship between the augmented arrays of $\mathsf{EVENODD}(p,k+1,r)$ and $\mathsf{RDP}(p,k,r)$ in the following.

Proposition 3. *The augmented array of $\mathsf{RDP}(p,k,r)$ can be obtained by shortening the augmented array of*

TABLE I: Augmented array of $\mathsf{RDP}(5,3,3)$.

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}=b_{0,0}+b_{0,1}+b_{0,2}$	$b_{0,4}=b_{0,0}+b_{3,2}+b_{2,3}$	$b_{0,5}=b_{0,0}+b_{3,1}+b_{1,2}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}=b_{1,0}+b_{1,1}+b_{1,2}$	$b_{1,4}=b_{1,0}+b_{0,1}+b_{3,3}$	$b_{1,5}=b_{1,0}+b_{2,2}+b_{0,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}=b_{2,0}+b_{2,1}+b_{2,2}$	$b_{2,4}=b_{2,0}+b_{1,1}+b_{0,2}$	$b_{2,5}=b_{2,0}+b_{0,1}+b_{3,2}+b_{1,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}=b_{3,0}+b_{3,1}+b_{3,2}$	$b_{3,4}=b_{3,0}+b_{2,1}+b_{1,2}+b_{0,3}$	$b_{3,5}=b_{3,0}+b_{1,1}+b_{2,3}$
0	0	0	0	$b_{4,4}=b_{3,1}+b_{2,2}+b_{1,3}$	$b_{4,5}=b_{2,1}+b_{0,2}+b_{3,3}$

TABLE II: Augmented array of $\mathsf{EVENODD}(5,3,3)$.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}=a_{0,0}+a_{0,1}+a_{0,2}$	$a_{0,4}=a_{0,0}+a_{3,2}$	$a_{0,5}=a_{0,0}+a_{3,1}+a_{1,2}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}=a_{1,0}+a_{1,1}+a_{1,2}$	$a_{1,4}=a_{1,0}+a_{0,1}$	$a_{1,5}=a_{1,0}+a_{2,2}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}=a_{2,0}+a_{2,1}+a_{2,2}$	$a_{2,4}=a_{2,0}+a_{1,1}+a_{0,2}$	$a_{2,5}=a_{2,0}+a_{0,1}+a_{3,2}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}=a_{3,0}+a_{3,1}+a_{3,2}$	$a_{3,4}=a_{3,0}+a_{2,1}+a_{1,2}$	$a_{3,5}=a_{3,0}+a_{1,1}$
0	0	0	0	$a_{4,4}=a_{3,1}+a_{2,2}$	$a_{4,5}=a_{2,1}+a_{0,2}$

$\mathsf{EVENODD}(p,k+1,r)$ as follows: (i) imposing the following additional constraint on the information bits for $i=0,1,\ldots,p-1$, namely

$$a'_{i,k}=a'_{i,0}+a'_{i,1}+\cdots+a'_{i,k-1}; \tag{13}$$

(ii) removing column $k+1$ of the augmented array of $\mathsf{EVENODD}(p,k+1,r)$.

Proof. Consider the augmented array of $\mathsf{EVENODD}(p,k+1,r)$ and assume that the information bits of column k satisfy (13). By (1), the parity bits in column $k+1$ are all zeros. After deleting column $k+1$ from the augmented array of $\mathsf{EVENODD}(p,k+1,r)$ and reindexing the columns after this deleted column by reducing all indices by one, we have a new array with $k+r$ columns of a shortened $\mathsf{EVENODD}(p,k,r)$. Let the augmented array of $\mathsf{RDP}(p,k,r)$ with the k information columns being the same as the first k information columns of the augmented array of $\mathsf{EVENODD}(p,k+1,r)$ such that these columns are the same as those of the array of the shortened $\mathsf{EVENODD}(p,k,r)$. Then column k of the augmented array of $\mathsf{RDP}(p,k,r)$ is the same as column k of the array of the shortened $\mathsf{EVENODD}(p,k,r)$ according to (13) and (4). Recall that the bit $b_{i,k+\ell}$ in column $k+\ell$, $i=0,1,\ldots,p-1$ and $\ell=2,3,\ldots,r-1$, of the augmented array of $\mathsf{RDP}(p,k,r)$ is computed by (5) (or (6)). Because $a'_{i,j}=a_{i,j}=b_{i,j}$ for $i=0,1,\ldots,p-1$ and $j=0,1,\ldots,k$, $b_{i,k+\ell}$ is the same as $a'_{i,k+\ell}$ in the array of the shortened $\mathsf{EVENODD}(p,k,r)$ that is defined by (7). Therefore, we can obtain the augmented $\mathsf{RDP}(p,k,r)$ by shortening the augmented $\mathsf{EVENODD}(p,k+1,r)$ by imposing the condition (13) and removing column $k+1$. This completes the proof. □

By Proposition 3, the unified form of $\mathsf{RDP}(p,k,r)$ and $\mathsf{EVENODD}(p,k,r)$ is the augmented array of $\mathsf{EVENODD}(p,k+1,r)$. In the following, we focus on $\mathsf{EVENODD}(p,k,r)$.

III. Algebraic Representation

Let $\mathbb{F}_2[x]$ be the ring of polynomials over binary field $\mathbb{F}_2$, and let R_p be the quotient ring $\mathbb{F}_2[x]/(1+x^p)$. An element in R_p can be represented by a polynomial of degree strictly less than p with coefficients in $\mathbb{F}_2$. In the following, we refer to an element of R_p as a polynomial. Note that multiplying two polynomials in R_p is performed under modulo $1+x^p$.

The ring R_p has been discussed in [23], [24] and has been used to design regenerating codes with low computational complexity. Let

$$M_p(x)=1+x+\cdots+x^{p-1}.$$

R_p is isomorphic to a direct sum of two finite fields $\mathbb{F}_2[x]/(1+x)$ and $\mathbb{F}_2[x]/M_p(x)$[2] if and only if 2 is a primitive element in $\mathbb{F}_p$ [25]. In [26], $\mathbb{F}_2[x]/M_p(x)$ was used to perform computations in $\mathbb{F}_{2^{p-1}}$, where p is a prime such that 2 is a primitive element in $\mathbb{F}_p$. In addition, Blaum *et al.* [6], [7] discussed the rings $\mathbb{F}_2[x]/M_p(x)$ in detail.

We represent each column in an augmented array of $\mathsf{EVENODD}(p,k,r)$ by a polynomial in R_p, so that a $p\times n$ array is identified with an n-tuple

$$(a'_0(x),a'_1(x),\cdots,a'_{k+r-1}(x)) \tag{14}$$

in R_p^{k+r}, where $n=k+r$. In the $p\times(k+r)$ array, the p bits $a'_{0,j},a'_{1,j},\ldots,a'_{p-1,j}$ in column j for $j=0,1,\ldots,k+r-1$ can be represented as a polynomial

$$a'_j(x)=a'_{0,j}+a'_{1,j}x+\cdots+a'_{p-1,j}x^{p-1}.$$

The first k polynomials $a'_0(x),a'_1(x),\ldots,a'_{k-1}(x)$ are called *information polynomials*, and the last r polynomials $a'_k(x),a'_{k+1}(x),\ldots,a'_{k+r-1}(x)$ are *parity polynomials*. The parity bit of the augmented array of $\mathsf{EVENODD}(p,k,r)$ defined in (7) is equivalent to the following equation over R_p, namely

$$[a'_k(x)\ \cdots\ a'_{k+r-1}(x)]=[a'_0(x)\ \cdots\ a'_{k-1}(x)]\cdot\mathbf{V}_{k\times r}, \tag{15}$$

where $\mathbf{V}_{k\times r}$ is the $k\times r$ Vandermonde matrix

$$\mathbf{V}_{k\times r}=\begin{bmatrix}1 & 1 & \cdots & 1\\ 1 & x^1 & \cdots & x^{r-1}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & x^{k-1} & \cdots & x^{(r-1)(k-1)}\end{bmatrix} \tag{16}$$

and additions and multiplications in the above calculations are performed in R_p. Equation (15) can be verified as follows:

$$\begin{aligned}a'_{k+\ell}(x)&=\textstyle\sum_{i=0}^{p-1}a'_{i,k+\ell}x^i=\sum_{j=0}^{k-1}a'_j(x)x^{\ell\cdot j}\\ &=\textstyle\sum_{j=0}^{k-1}\sum_{i=0}^{p-1}a'_{i,j}x^{i+\ell\cdot j}=\sum_{i=0}^{p-1}\sum_{j=0}^{k-1}a'_{i,j}x^{i+\ell\cdot j}.\end{aligned} \tag{17}$$

[2]When 2 is a primitive element in $\mathbb{F}_p$, $\mathbb{F}_2[x]/M_p(x)$ is a finite field.

We have $a'_{i,k+\ell} = \sum_{j=0}^{k-1} a_{i-\ell\cdot j,j}$, which is the same as (7). In other words, each parity column in the augmented array of EVENODD codes is obtained by adding some cyclically shifted version of the information columns.

Recall that $a'_j(x)$ is a polynomial over R_p for $j = 0, 1, \ldots, k+r-1$. When we reduce a polynomial $a'_j(x)$ modulo $M_p(x)$, it means that we replace the coefficient $a'_{i,j}$ with $a'_{i,j}+a'_{p-1,j}$ for $i = 0, 1, \ldots, p-2$. When $j = 0, 1, \ldots, k$, we have that $a'_{p-1,j} = 0$. If we reduce $a'_j(x)$ modulo $M_p(x)$, we obtain $a'_j(x)$ itself, the coefficients of which are the bits of column j of $\mathsf{EVENODD}(p, k, r)$. Recall that the coefficients of $a'_{k+\ell}(x)$ for $\ell = 1, 2, \ldots, r-1$ are computed by (7). If we reduce $a'_{k+\ell}(x)$ modulo $M_p(x)$, we can obtain the $p-1$ bits in column $k+\ell$ of $\mathsf{EVENODD}(p, k, r)$. In fact, we have shown how to convert the augmented array of $\mathsf{EVENODD}(p, k, r)$ into the original array of $\mathsf{EVENODD}(p, k, r)$.

By Proposition 3, we can obtain the augmented array of $\mathsf{RDP}(p, k, r)$ by computing

$$(b_0(x), \cdots, b_{k-1}(x), \sum_{j=0}^{k-1} b_j(x)) \begin{bmatrix} \mathbf{I}_{k+1} & \mathbf{V}_{(k+1)\times r} \end{bmatrix},$$

and removing component $k+2$, which is always equal to zero, in the resultant product. If we arrange all coefficients in the polynomials with degree strictly less than $p-1$, we get the original $(p-1)\times(k+r)$ array of $\mathsf{RDP}(p, k, r)$.

The MDS property condition of $\mathsf{EVENODD}(p, k, r)$ is the same as that of the extended EVENODD codes [6]. The MDS property conditions of $r \leq 8$ and $r \geq 9$ were given in [6] and [18], respectively. Herein, we assume that the proposed $\mathsf{EVENODD}(p, k, r)$ and $\mathsf{RDP}(p, k, r)$ are MDS codes, and we focus on their erasure decoding.

In the remainder of the present paper, we present a fast decoding method to solve a Vandermonde linear system over R_p. The method can be used to recover the erasure patterns of $\mathsf{EVENODD}(p, k, r)$ and $\mathsf{RDP}(p, k, r)$. If the number of erased information columns does not exceed the number of continuous surviving parity columns and the first parity column has not failed, then we can represent the erased information polynomials by a Vandermonde linear system to find the erased columns efficiently. Therefore, we assume hereinafter that the number of erased information columns does not exceed the number of continuous surviving parity columns and the first parity column has not failed when applying the proposed decoding method to $\mathsf{EVENODD}(p, k, r)$. Note that one must recover the failed columns by downloading k surviving columns. First, we represent the k downloaded columns by some information polynomials and continuous parity polynomials. Then, we subtract all the downloaded information polynomials from the parity polynomials to obtain a Vandermonde linear system. Although $\mathsf{EVENODD}(p, k, r)$ can be described by the $k \times r$ Vandermonde matrix given in (16) over $\mathbb{F}_2[x]/M_p(x)$ and we can solve the Vandermonde linear system over $\mathbb{F}_2[x]/M_p(x)$ to recover the failure columns, it is more efficient to solve the Vandermonde linear system over R_p. We show in Section IV that we can calculate over R_p and then reduce the results modulo $M_p(x)$ in the decoding process. Then, in Section V, we propose an efficient decoding algorithm to solve a Vandermonde linear system over R_p.

IV. Vandermonde Matrix over R_p

Because the decoding algorithm proposed in Section V hinges on a quick method for solving a Vandermonde system of equations over R_p, we discuss some properties of the linear system of a Vandermonde matrix over R_p in this section.

Let $\mathbf{V}_{r\times r}(\mathbf{a})$ be an $r \times r$ Vandermonde matrix, namely

$$\mathbf{V}_{r\times r}(\mathbf{a}) = \begin{bmatrix} 1 & x^{a_1} & \cdots & x^{(r-1)a_1} \\ 1 & x^{a_2} & \cdots & x^{(r-1)a_2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{a_r} & \cdots & x^{(r-1)a_r} \end{bmatrix}, \tag{18}$$

where $a_1, \ldots, a_r$ are distinct integers and the difference of each pair of them is relatively prime to p. The entries of $\mathbf{V}_{r\times r}(\mathbf{a})$ are considered as polynomials in R_p. We investigate the action of multiplication over $\mathbf{V}_{r\times r}(\mathbf{a})$ by defining the function $F : R_p^r \to R_p^r$, namely

$$F(\mathbf{u}) = \mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a})$$

for $\mathbf{u} = (u_1(x), \ldots, u_r(x)) \in R_p^r$. Obviously, F is a homomorphism of abelian groups and we have $F(\mathbf{u}+\mathbf{u}') = F(\mathbf{u}) + F(\mathbf{u}')$ for $\mathbf{u}, \mathbf{u}' \in R_p^r$.

If a vector $\mathbf{v} = (v_1(x), \ldots, v_r(x))$ is equal to $F(\mathbf{u})$ for some $\mathbf{u} \in R_p^r$, it is necessary that

$$v_1(1) = v_2(1) = \cdots = v_r(1). \tag{19}$$

This is due to the fact that each polynomial $v_j(x)$ is obtained by adding certain cyclically shifted version of $u_i(x)$'s. In other words, if $\mathbf{v}$ is in the image of F, then either there is an even number of nonzero terms in all $v_i(x)$ or there is an odd number of nonzero terms in all $v_i(x)$ for $1 \leq i \leq r$.

We require the following lemma before discussing the properties of the Vandermonde linear system over R_p.

Lemma 4. *[6, Lemma 2.1] Suppose that p is an odd number and d is relatively prime to p, then $1+x^d$ is relatively prime to $M_p(x)$ in $\mathbb{F}_2[x]$, and x^i and $M_p(x)$ are relatively prime in $\mathbb{F}_2[x]$ for any positive integer i.*

If the vector $\mathbf{v}$ satisfies (19), in the next theorem, we show that there are many vectors $\mathbf{u}$ such that $F(\mathbf{u}) = \mathbf{v}$.

Theorem 5. *Let $a_1, a_2, \ldots, a_r$ be r integers such that the difference $a_{i_1} - a_{i_2}$ is relatively prime to p for all pairs of distinct indices $1 \leq i_1 < i_2 \leq r$. The image of F consists of all vectors $\mathbf{v} \in R_p^r$ that satisfy the condition* (19)*. All vectors $\mathbf{u}$ that satisfy*

$$\mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a}) = \mathbf{v} \mod 1+x^p \tag{20}$$

are congruent to each other modulo $M_p(x)$.

Proof. Suppose that $v_1(x), \ldots, v_r(x)$ are polynomials in R_p satisfying (19). We want to show that the vector $\mathbf{v} = (v_1(x), \ldots, v_r(x))$ is in the image of F. We first consider the case that $v_1(1) = v_2(1) = \cdots = v_r(1) = 0$. Because $1+x$

and $M_p(x)$ are relatively prime polynomials, by the Chinese remainder theorem, we have an isomorphism

$$\theta(f(x)) = (f(x) \bmod 1+x, f(x) \bmod M_p(x))$$

defined for $f(x) \in R_p$. The inverse of θ is given by

$$\theta^{-1}(a(x), b(x)) = M_p(x)a(x)+(1+M_p(x))b(x) \bmod 1+x^p,$$

where $a(x) \in \mathbb{F}_2[x]/(1+x)$ and $b(x) \in \mathbb{F}_2[x]/M_p(x)$. We thus have a decomposition of the ring R_p as a direct sum of $\mathbb{F}_2[x]/(1+x)$ and $\mathbb{F}_2[x]/M_p(x)$. It suffices to investigate the action of multiplication over $\mathbf{V}_{r\times r}(\mathbf{a})$ by considering

$$\mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a}) = \mathbf{v} \mod 1+x, \text{ and} \tag{21}$$

$$\mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a}) = \mathbf{v} \mod M_p(x). \tag{22}$$

Note that (21) is equivalent to

$$(\mathbf{u} \bmod (1+x)) \cdot (\mathbf{V}_{r\times r}(\mathbf{a}) \bmod (1+x)) = \mathbf{v} \bmod (1+x).$$

Also, $\mathbf{V}_{r\times r}(\mathbf{a}) \bmod (1+x)$ is an $r \times r$ all-one matrix and $\mathbf{v} \bmod (1+x) = \mathbf{0}$ because $v_i(1) = 0,\ 1 \le i \le r$. Therefore, there are many solutions $\mathbf{u}'$, each of which has an even number of ones among all its components after being reduced modulo $1+x$.

For (22), we need to show that the determinant of $\mathbf{V}_{r\times r}(\mathbf{a})$ is invertible modulo $M_p(x)$. Because the determinant of $\mathbf{V}_{r\times r}(\mathbf{a})$ is[3]

$$\det(\mathbf{V}_{r\times r}(\mathbf{a})) = \prod_{i_1<i_2} (x^{a_{i_1}} + x^{a_{i_2}}),$$

we need to show that $x^{a_{i_1}} + x^{a_{i_2}}$ and $M_p(x)$ are relatively prime polynomials in $\mathbb{F}_2[x]$, for all pairs of distinct (i_1, i_2). We first factorize $x^{a_{i_1}} + x^{a_{i_2}}$ in the form $x^{a_{i_1}} + x^{a_{i_2}} = x^{a_{i_1}}(1+x^d)$, and by assumption, d is an integer that is relatively prime to p. This problem now reduces to showing that (i) $1+x^d$ and $M_p(x)$ are relatively prime in $\mathbb{F}_2[x]$ whenever $\gcd(d,p) = 1$, and (ii) x^ℓ and $M_p(x)$ are relatively prime in $\mathbb{F}_2[x]$ for all integers ℓ. We can show this by Lemma 4. Therefore, we can solve the equation in (22) by Cramer's rule, say, to obtain the unique solution $\mathbf{u}''$. After obtaining the solutions $u_i'(x) \in \mathbb{F}_2[x]/(1+x)$ and $u_i''(x) \in \mathbb{F}_2[x]/M_p(x)$ to (21) and (22), respectively, for all i, we can calculate the solution via the isomorphism θ^{-1}, namely

$$\begin{aligned} & \theta^{-1}(u_i'(x), u_i''(x)) \\ = \ & M_p(x)u_i'(x) + (1+M_p(x))u_i''(x) \bmod 1+x^p. \end{aligned}$$

Therefore, the solutions of $\mathbf{u} \in R_p^r$ in (20) are

$$\begin{aligned} &((1+M_p(x))u_1''(x) + \epsilon_1 M_p(x), \cdots, \\ &\quad (1+M_p(x))u_r''(x) + \epsilon_r M_p(x)), \end{aligned} \tag{23}$$

where ϵ_i is equal to 0 or 1 for all i and the number of ones is an even number. That is to say, there are many solutions to (20), all of which are unique after being reduced modulo $M_p(x)$ and are solutions to (22).

When $v_1(1) = v_2(1) = \cdots = v_r(1) = 1$, a similar argument can be used to find the solution. The only difference between this solution and (23) is that there is an odd number of ones among all ϵ_i in the former. This completes the proof. □

From Theorem 5, whenever the vector $\mathbf{v}$ satisfies the condition (19), we can decode one solution of $\mathbf{u}$ in (20). Recall that for the augmented array of EVENODD codes, the coefficient of the term with degree $p-1$ of every component of $\mathbf{u}$ is zero. Therefore, each component of the real solution is of at most degree $p-2$. By Theorem 5, reducing $\mathbf{u}$ modulo $M_p(x)$ gives us the final solution. Therefore, to solve $\mathbf{u}$ in (22), we can first solve $\mathbf{u}$ over R_p and then take every component of $\mathbf{u}$ as modulo $M_p(x)$. We demonstrate this in Section V.

V. Efficient Decoding of Vandermonde System over R_p

In this section, we present an efficient decoding method for a Vandermonde system over R_p based on LU factorization of the Vandermonde matrix.

A. Efficient Division by $1+x^d$

We begin by presenting two decoding algorithms for dividing by $1+x^d$, which will be used in the decoding process, where d is a positive integer that is relatively prime to p and all operations are performed in the ring $R_p = \mathbb{F}_2[x]/(1+x^p)$. Consider the equation

$$(1+x^d)g(x) = f(x) \bmod 1+x^p, \tag{24}$$

where d is a positive integer such that $\gcd(d,p) = 1$ and $f(x)$ has an even number of nonzero terms. Lemma 6 in [27] is a way to compute $g(x)$ and is summarized below.

Lemma 6. *[27, Lemma 6] The coefficients of $g(x)$ in* (24) *can be computed by*

$$\begin{aligned} &g_{p-1} = 0, g_{p-d-1} = f_{p-1}, g_{d-1} = f_{d-1}, \\ &g_{(p-(i+1)d-1)} = f_{(p-id-1)} + g_{(p-id-1)} \text{ for } i = 1, \ldots, p-3, \end{aligned}$$

where $g(x) = \sum_{i=0}^{p-1} g_i x^i$, $f(x) = \sum_{i=0}^{p-1} f_i x^i$.

Although computing the division in (24) by Lemma 6 takes only $p-3$ XORs, we do not know whether the resulting polynomial $g(x)$ has an even number of nonzero terms. In solving the Vandermonde linear system in Section V-B, we must compute many divisions of the form in (24). If we do not require the solved polynomial $g(x)$ to have an even number of nonzero terms, then we can use Lemma 6 to solve the division. Otherwise, Lemma 6 is not applicable to such division. Therefore, we require Lemma 7 to compute the division when $g(x)$ is required to have an even number of nonzero terms.

Lemma 7. *[24, Lemma 13] Given* (24)*, we can compute the coefficient g_0 by*

$$g_0 = f_{2d} + f_{4d} + \cdots + f_{(p-1)d}, \tag{25}$$

and the other coefficients of $g(x)$ can be computed iteratively by

$$g_{d\ell} = f_{d\ell} + g_{d(\ell-1)} \text{ for } \ell = 1, 2, \ldots, p-1. \tag{26}$$

[3]Because -1 is the same as 1 in $\mathbb{F}_2$, we replace -1 with 1 herein, and addition is the same as subtraction.

Note that the subscripts in Lemma 7 are taken modulo p. Because $\gcd(d,p)=1$, we have that

$$\{0,d,2d,\cdots,(p-1)d\}=\{0,1,2,\cdots,p-1\} \bmod p.$$

Therefore, we can compute all the coefficients of $g(x)$ by Lemma 7. The result in Lemma 7 has been observed in [24], [28]. We can check that the computed polynomial $g(x)$ satisfies $g(1)=0$ by adding (25) and (26) for $\ell=2,4,\ldots,p-1$. The number of XORs required to compute the division by Lemma 7 is $(3p-5)/2$.

For the same parameters, $g(x)$ computed by Lemma 6 is equal to either $g(x)$ computed by Lemma 7 or the summation of $M_p(x)$ and $g(x)$ computed by Lemma 7, depending on whether $g(x)$ computed by Lemma 6 has an even number of nonzero terms. When computing the division in (24), we prefer using Lemma 6 if there is no requirement for the resulting polynomial $g(x)$ to have an even number of nonzero terms, the reason being that Lemma 6 involves fewer XORs.

B. LU Method for Vandermonde Systems over R_p

Theorem 8 is the core of the fast LU method for solving a Vandermonde system of equations $\mathbf{v}=\mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a})$ and is based on the LU decomposition of a Vandermonde matrix given in [29].

Theorem 8. *[29] For a positive integer r, the square Vandermonde matrix $\mathbf{V}_{r\times r}(\mathbf{a})$ defined in* (18) *can be factorized into*

$$\mathbf{V}_{r\times r}(\mathbf{a})=\mathbf{L}_r^{(1)}\mathbf{L}_r^{(2)}\cdots\mathbf{L}_r^{(r-1)}\mathbf{U}_r^{(r-1)}\mathbf{U}_r^{(r-2)}\cdots\mathbf{U}_r^{(1)},$$

where $\mathbf{U}_r^{(\ell)}$ is the upper triangular matrix

$$\mathbf{U}_r^{(\ell)}:=\left[\begin{array}{c|cccccc}\mathbf{I}_{r-\ell-1} & \multicolumn{6}{c}{\boldsymbol{0}}\\ \hline & 1 & x^{a_1} & 0 & \cdots & 0 & 0\\ & 0 & 1 & x^{a_2} & \cdots & 0 & 0\\ \boldsymbol{0} & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ & 0 & 0 & 0 & \cdots & 1 & x^{a_\ell}\\ & 0 & 0 & 0 & \cdots & 0 & 1\end{array}\right]$$

and $\mathbf{L}_r^{(\ell)}$ is the lower triangular matrix

$$\left[\begin{array}{c|cccc}\mathbf{I}_{t-1} & \multicolumn{4}{c}{\boldsymbol{0}}\\ \hline & 1 & 0 & \cdots & 0 & 0\\ & 1 & x^{a_{t+1}}+x^{a_t} & \cdots & 0 & 0\\ \boldsymbol{0} & \vdots & \vdots & \ddots & \vdots & \vdots\\ & 0 & 0 & \cdots & x^{a_{r-1}}+x^{a_t} & 0\\ & 0 & 0 & \cdots & 1 & x^{a_r}+x^{a_t}\end{array}\right]$$

for $\ell=1,2,\ldots,r-1$, where $t=r-\ell$.

For example, the Vandermonde matrix $\mathbf{V}_{3\times 3}(1,x,x^4)$ can be factorized as

$$\mathbf{L}_3^{(1)}\mathbf{L}_3^{(2)}\mathbf{U}_3^{(2)}\mathbf{U}_3^{(1)}=\begin{bmatrix}1&0&0\\0&1&0\\0&1&x^4+x\end{bmatrix}\cdot\begin{bmatrix}1&0&0\\1&x+1&0\\0&1&x^4+1\end{bmatrix}\begin{bmatrix}1&1&0\\0&1&x\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&1\\0&0&1\end{bmatrix}.$$

Based on the factorization in Theorem 8, we have a fast algorithm for solving a Vandermonde system of linear equations. Given the matrix $\mathbf{V}_{r\times r}(\mathbf{a})$ and a row vector $\mathbf{v}=(v_1(x),\ldots,v_r(x))$, we can solve the linear system $\mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a})=\mathbf{v}$ in (20) by solving

$$\mathbf{u}\mathbf{L}_r^{(1)}\mathbf{L}_r^{(2)}\cdots\mathbf{L}_r^{(r-1)}\mathbf{U}_r^{(r-1)}\mathbf{U}_r^{(r-2)}\cdots\mathbf{U}_r^{(1)}=\mathbf{v}. \quad (27)$$

Because each upper or lower triangular matrix can be inverted efficiently, we can solve for $\mathbf{u}$ by inverting $2(r-1)$ triangular matrices.

Algorithm 1 Solving a Vandermonde linear system.

Inputs: positive integer r, odd integer p, integers $a_1,a_2,\ldots,a_r$, and $\mathbf{v}=(v_1(x),v_2(x),\ldots,v_r(x))\in R_p^r$.
Output: $\mathbf{u}=(u_1(x),\ldots,u_r(x))$ that satisfies $\mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a})=\mathbf{v}$.
Require: $v_1(1)=v_2(1)=\cdots=v_r(1)$, and $\gcd(a_{i_1}-a_{i_2},p)=1$ for all $1\le i_1<i_2\le r$.

1: $\mathbf{u}\leftarrow\mathbf{v}$.
2: **for** i from 1 to $r-1$ **do**
3: **for** j from $r-i+1$ to r **do**
4: $u_j(x)\leftarrow u_j(x)+u_{j-1}(x)x^{a_{i+j-r}}$
5: **for** i from $r-1$ down to 1 **do**
6: Solve $g(x)$ from $(x^{a_r}+x^{a_{r-i}})g(x)=u_r(x)$ by Lemma 7 or 6 (only when $i=1$)
$u_r(x)\leftarrow g(x)$
7: **for** j from $r-1$ down to $r-i+1$ **do**
8: Solve $g(x)$ from $(x^{a_j}+x^{a_{r-i}})g(x)=(u_j(x)+u_{j+1}(x))$ by Lemma 7 or 6 (only when $i+j=r+1$)
$u_j(x)\leftarrow g(x)$
9: $u_{r-i}(x)\leftarrow u_{r-i}(x)+u_{r-i+1}(x)$
10: **return** $\mathbf{u}=(u_1(x),\ldots,u_r(x))$

The procedure for solving a Vandermonde system of linear equations is given in Algorithm 1, wherein steps 2–4 are forward additions that require $r(r-1)/2$ additions and $r(r-1)/2$ multiplications. Steps 5–9 are backward additions that require $r(r-1)/2$ additions and $r(r-1)/2$ divisions by factors of the form $x^{a_j}+x^{a_{r-i}}$. Division by $x^{a_j}+x^{a_{r-i}}$ is done by invoking the method in Lemma 6 or 7. We could compute all the divisions by Lemma 7, but some can be computed by Lemma 6, thereby reducing the computational complexity.

Theorem 9. *Algorithm 1 outputs $\mathbf{u}$, which is a solution to $\mathbf{u}\mathbf{V}_{r\times r}(\mathbf{a})=\mathbf{v}$ over R_p. Furthermore, $g(x)$ in (i) step 6 when $i=1$ and (ii) step 8 when $i+j=r+1$ and $i\in\{2,3,\ldots,r-1\}$ can be computed by Lemma 6 to reduce computation complexity.*

Proof. First, we want to show that Algorithm 1 implements precisely the matrix multiplications in (27). Consider the linear equations $\mathbf{u}\mathbf{U}_r^{(i)}=\mathbf{v}$ for $i=1,2,\ldots,r-1$. According to the upper triangular matrix $\mathbf{U}_r^{(i)}$ in Theorem 8, we can obtain the relationship between $\mathbf{u}$ and $\mathbf{v}$ as

$$u_j(x)=v_j(x) \text{ for } j=1,\ldots,r-i,$$
$$x^{a_{i+j-r}}u_{j-1}(x)+u_j(x)=v_j(x) \text{ for } j=r-i+1,\ldots,r.$$

We see from Algorithm 1 that steps 1–4 solve $\mathbf{u}$ from $\mathbf{u}\mathbf{U}_r^{(r-1)}\mathbf{U}_r^{(r-2)}\cdots\mathbf{U}_r^{(1)} = \mathbf{v}$, and we denote the solved r polynomials as $\mathbf{v}' = (v_1'(x),\ldots,v_r'(x))$. Consider the equations $\mathbf{u}\mathbf{L}_r^{(i)} = \mathbf{v}'$. According to the lower triangular matrix $\mathbf{L}_r^{(i)}$ in Theorem 8, the relationship between $\mathbf{u}$ and $\mathbf{v}'$ is as follows:

$$\begin{aligned}
&u_j(x) = v_j'(x) \text{ for } j = 1,\ldots,r-i-1,\\
&u_{r-i}(x) + u_{r-i+1}(x) = v_{r-i}'(x), \qquad (28)\\
&(x^{a_j} + x^{a_{r-i}})u_j(x) + u_{j+1}(x) = v_j'(x)\\
&\text{ for } j = r-i+1,\ldots,r-1, \qquad (29)\\
&(x^{a_r} + x^{a_{r-i}})u_r(x) = v_r'(x). \qquad (30)
\end{aligned}$$

It is easily seen that steps 6, 8, and 9 solve $u_r(x)$ from (30), $u_j(x)$ from (29), and $u_{r-i}(x)$ from (28), respectively. Thus we have that steps 5–9 solve $\mathbf{u}$ from $\mathbf{u}\mathbf{L}_r^{(1)}\mathbf{L}_r^{(2)}\cdots\mathbf{L}_r^{(r-1)} = \mathbf{v}'$. Therefore, Algorithm 1 computes $\mathbf{u}$ precisely from the matrix multiplication in (27).

Note that we need to compute $r(r-1)/2$ divisions (solving $g(x)$) in steps 6 and 8 in Algorithm 1, and all divisions can be solved by Lemma 6 or 7. To solve all divisions by Lemma 6 or 7, all polynomials $u_r(x)$ and $u_j(x) + u_{j+1}(x)$ in steps 6 and 8 must have even numbers of nonzero terms, which is a requirement in computing the divisions. That is, $u_r(1) = 0$ and $u_j(1) + u_{j+1}(1) = 0$ when we calculate $g(x)$ in these steps. Next, we show that this requirement is met during the process of Algorithm 1.

Consider two cases, namely (i) $v_i(1) = 0$ for all i and (ii) $v_i(1) = 1$ for all i. If $v_i(1) = 0$ for all i, then we have $u_i(1) = 0$ for all i after the double "for" loops between steps 2 and 4. Therefore, $u_r(1) = 0$ in step 6 when $i = r-1$. In step 6, we must ensure that all $u_r(x)$ produced satisfy $u_r(1) = 0$. Because all $g(x)$, and therefore $u_r(x)$, generated by Lemma 7 have such a property, we need consider only the case in which $g(x)$ is generated by Lemma 6, that is, when $i = 1$. When $i = 1$, the $u_r(x)$ assigned by $g(x)$ will not be invoked in computing the division in steps 7 and 8 because $r-i+1 = r$, which is larger than the initial value $r-1$ of the loop in step 7. Therefore, there is no need to run the "for" loop in steps 7 and 8.

Similarly, we must ensure that $u_j(1) + u_{j+1}(1) = 0$ for $j = r-1, r-2, \ldots, r-i-1$. It is sufficient to ensure that $u_j(1) = 0$ and $u_{j+1}(1) = 0$. Because all $g(x)$, and therefore $u_j(x)$, generated by Lemma 7 already have such a property, in step 8 we need consider only the case in which $g(x)$ is generated by Lemma 6, that is, when $i + j = r + 1$. Next, we prove that when $i + j = r + 1$, the $u_j(x)$ assigned by $g(x)$ in step 8 are not used again to solve the division in step 8. Let $i = t$ $(i = r-1,\ldots,2)$. Note that when $i + j = r + 1$, Algorithm 1 is in the final iteration of the "for" loop in step 7. Therefore, we must prove that $u_j(x) = u_{r-t+1}(x)$ will not be used in the iterations $i < t$. When $i < t$ in step 7, as the last iteration $j = r - i + 1 > r - t + 1$, so $u_{r-t+1}(x)$ will not be used in the calculation involving $u_j(x)$ and $u_{j+1}(x)$ in step 8.

If $v_i(1) = 1$ for all i, then after the double "for" loops between steps 2 and 4, we have

$$u_1(1) = 1, \text{ and } u_2(1) = u_3(1) = \cdots = u_r(1) = 0.$$

In this case, we need show only that $u_1(x)$ has never been used in the calculation of $g(x)$ in step 8. Note that in the last iteration of step 7, $j = r - i + 1$ has never gone down to 1 because $i \le r - 1$. Therefore, $u_1(x)$ has never been used in step 8. □

Theorem 10 specifies the computational complexity of Algorithm 1.

Theorem 10. *The computation complexity of Algorithm 1 is at most*

$$r(r-1)p + (r-1)(p-3) + (r-1)(r-2)(3p-5)/4. \quad (31)$$

Proof. In Algorithm 1, there are $r(r-1)$ additions that require $r(r-1)p$ XORs, $r(r-1)/2$ multiplications that require no XORs (only cyclic shifts are applied), and $r(r-1)/2$ divisions that require $(r-1)(p-3)+(r-1)(r-2)(3p-5)/4$ XORs ($r-1$ divisions are computed by Lemma 6 and the other divisions are computed by Lemma 7). Therefore, the total number of computations in Algorithm 1 is at most that given by (31). □

According to Theorem 5, we can solve the Vandermonde system over $\mathbb{F}_2[x]/M_p(x)$ by first solving the Vandermonde system over R_p and then reducing the r resulting polynomials by modulo $M_p(x)$. By Theorem 9, we can solve the Vandermonde system over R_p by using the factorization method in Theorem 8.

Remark. Because in Algorithm 1 the output $u_r(x)$ when $i = 1$ is computed from the division in step 6, which is solved by Lemma 6, we have that the last coefficient of $u_r(x)$, namely g_{p-1}, is zero. For $i = 1, 2, \ldots, r-1$, the output $u_{r-i}(x)$ is a summation of $u_{r-i}(x)$ and $u_{r-i+1}(x)$, where $u_{r-i+1}(x)$ is computed in the last iteration in step 7 for $i = r-1, r-2, \ldots, 2$, and $u_{r-i}(x)$ is computed in the last iteration in step 7 for $i+1 = r-1, r-2, \ldots, 2$. Note that the last iteration for each i is solved by Lemma 6. Therefore, the last coefficient of $u_i(x)$ is zero for $i = 2, 3, \ldots, r-1$. The output $u_1(x)$ is the summation of $u_1(x)$ and $u_2(x)$ by step 9, where $u_1(x) = v_1(x)$ and $u_2(x)$ is computed from the division in step 8 when $i = r-1$, which is solved by Lemma 6. Therefore, if the last coefficient of $v_1(x)$ is zero, then the last coefficient of the output $u_1(x)$ is zero. Otherwise, the last coefficient of the output $u_1(x)$ is one. Therefore, we have that the last coefficient of each of the last $r-1$ resulting polynomials is zero. Therefore, it is not necessary to reduce the last $r-1$ resulting polynomials by modulo $M_p(x)$; we need do so only for the first resulting polynomial. In the example of $\mathsf{EVENODD}(5,3,3)$, the three components of the returned $\mathbf{u}$ are exactly equal to the three information polynomials of $\mathsf{EVENODD}(5,3,3)$ because the last coefficient of $v_1(x)$ is zero.

Although the LU decoding method for an $r \times r$ Vandermonde linear system over R_p is discussed also in Theorem 14 of [24], the complexity of the algorithm provided there is $\frac{7}{4}r(r-1)p$, which is larger than (31) herein. The reason for reduced computation specified in (31) is that there are $r-1$ divisions in Algorithm 1 that are solved with $p-3$ XORs involved for each division, whereas in [24] all $r(r-1)$

divisions are solved with $(3p-5)/2$ XORs involved for each division.

VI. Erasure Decoding of EVENODD(p,k,r) and RDP(p,k,r)

The proposed LU decoding method given in Section V is applicable to the erasure decoding of EVENODD(p,k,r) and RDP(p,k,r) only if the number of erased information columns does not exceed the number of continuous surviving parity columns and the first parity column has not failed.

A. Erasures Decoding of EVENODD(p,k,r)

Suppose that γ information columns $e_1,\ldots,e_\gamma$ and δ parity columns $f_1,\ldots,f_\delta$ are erased with $0\le e_1<\ldots<e_\gamma\le k-1$ and $k+1\le f_1<\ldots<f_\delta\le k+r-1$, where $k\ge\gamma\ge 0$, $r-1\ge\delta\ge 0$, and $\gamma+\delta=\rho\le r$. Letting $f_0=k-1$ and $f_{\delta+1}=k+r-1$, we assume that there exists $\lambda\in\{0,1,\ldots,\delta\}$ such that $f_{\lambda+1}-f_\lambda\ge\gamma+1$. We have that the columns $f_\lambda+1,\ldots,f_\lambda+\gamma$ are not erased. Let

$$\mathcal{A}:=\{0,1,\ldots,k-1\}\setminus\{e_1,e_2,\ldots,e_\gamma\}$$

be a set of indices of the available information columns. We begin by recovering the lost information columns by reading $k-\gamma$ information columns with indices $i_1,i_2,\ldots,i_{k-\gamma}\in\mathcal{A}$, and γ parity columns with indices $f_\lambda+1,f_\lambda+2,\ldots,f_\lambda+\gamma$. We then recover the failure parity column by re-encoding the failure parity bits according to (2) for $\ell=f_1-k,\ldots,f_\delta-k$.

First, we compute the bits of the γ parity columns $f_\lambda+1,f_\lambda+2,\ldots,f_\lambda+\gamma$ of the augmented array according to (8) and (9) in Lemma 2. This can be done because column k has not failed. Then, we represent the $k-\gamma$ information columns and γ parity columns by $k-\gamma$ information polynomials $a'_i(x)$ as

$$a'_i(x):=a_{0,i}+a_{1,i}x+\cdots+a_{p-2,i}x^{p-2} \tag{32}$$

for $i\in\mathcal{A}$ and by γ parity polynomials $a'_{f_\lambda+j}(x)$ as

$$a'_{f_\lambda+j}(x):=a'_{0,f_\lambda+j}+a'_{1,f_\lambda+j}x+\cdots+a'_{p-1,f_\lambda+j}x^{p-1} \tag{33}$$

for $j=1,2,\ldots,\gamma$, respectively. Then, we subtract the $k-\gamma$ information polynomials $a'_i(x)$ in (32), $i\in\mathcal{A}$, from the γ parity polynomials $a'_{f_\lambda+1}(x),\ldots,a'_{f_\lambda+\gamma}(x)$ in (33) to obtain γ *syndrome polynomial* $\bar{a}_h(x)$ over R_p as

$$\bar{a}_h(x)=a'_{f_\lambda+h}(x)+\sum_{i\in\mathcal{A}}a'_i(x)x^{g(i)\cdot(f_\lambda+h-k)}, \tag{34}$$

for $h=1,2,\ldots,\gamma$. Thus, we can establish the relationship between the syndrome polynomials and the erased information polynomials as

$$\begin{bmatrix}\bar{a}_1(x) & \cdots & \bar{a}_\gamma(x)\end{bmatrix}=\begin{bmatrix}a'_{e_1}(x) & \cdots & a'_{e_\gamma}(x)\end{bmatrix}\cdot\begin{bmatrix}x^{e_1(f_\lambda+1-k)} & x^{e_1(f_\lambda+2-k)} & \ldots & x^{e_1(f_\lambda+\gamma-k)}\\ x^{e_2(f_\lambda+1-k)} & x^{e_2(f_\lambda+2-k)} & \ldots & x^{e_2(f_\lambda+\gamma-k)}\\ \vdots & \vdots & \ddots & \vdots\\ x^{e_\gamma(f_\lambda+1-k)} & x^{e_\gamma(f_\lambda+2-k)} & \ldots & x^{e_\gamma(f_\lambda+\gamma-k)}\end{bmatrix}.$$

The right-hand side of the above can be reformulated as

$$\begin{bmatrix}x^{e_1(f_\lambda+1-k)}a'_{e_1}(x) & \cdots & x^{e_\gamma(f_\lambda+1-k)}a'_{e_\gamma}(x)\end{bmatrix}\mathbf{V}_{\gamma\times\gamma}(\mathbf{e}),$$

where $\mathbf{V}_{\gamma\times\gamma}(\mathbf{e})$ is a Vandermonde matrix

$$\mathbf{V}_{\gamma\times\gamma}(\mathbf{e}):=\begin{bmatrix}1 & x^{e_1} & \cdots & x^{e_1(\gamma-1)}\\ 1 & x^{e_2} & \cdots & x^{e_2(\gamma-1)}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & x^{e_\gamma} & \cdots & x^{e_\gamma(\gamma-1)}\end{bmatrix}.$$

By (17), we have that $a'_{f_\lambda+h}(1)=\sum_{j=0}^{k-1}a'_j(1)$ and thus we have

$$\bar{a}_h(1)=a'_{f_\lambda+h}(1)+\sum_{i\in\mathcal{A}}a'_i(1)=\sum_{j=0}^{k-1}a'_j(1)+\sum_{i\in\mathcal{A}}a'_i(1),$$

which is independent of h. Thus, we find that $\bar{a}_1(1)=\cdots=\bar{a}_r(1)$. We can then obtain the erased information polynomials by first solving the Vandermonde linear systems over R_p by Algorithm 1, cyclic-left shifting the solved polynomials $x^{e_i(f_\lambda+1-k)}a'_{e_i}(x)$ by $e_i(f_\lambda+1-k)$ positions for $i=1,2,\ldots,\gamma$, and then reducing $a'_{e_1}(x)$ modulo $M_p(x)$ when $\lambda>0$ according to the remark at the end of Section V-B. If $\lambda=0$, we have $f_\lambda+1-k=0$ and the last coefficient of $\bar{a}_1(x)$ is zero, then we do not need to reduce $a'_{e_1}(x)$ modulo $M_p(x)$ according to the remark at the end of Section V-B. The parity bits in the δ erased parity columns can be recovered by (2). Note that column k is assumed not to fail because it is needed to compute the bits of the augmented array by (8) and (9).

B. Erasure Decoding of RDP(p,k,r)

Similar to the decoding for EVENODD(p,k,r), we assume that γ information columns indexed by $e_1,\ldots,e_\gamma$ and δ parity columns $f_1,\ldots,f_\delta$ of RDP(p,k,r) are erased with $0\le e_1<\ldots<e_\gamma\le k-1$ and $k+1\le f_1<\ldots<f_\delta\le k+r-1$, where $k\ge\gamma\ge 0$, $r-1\ge\delta\ge 0$, and $\gamma+\delta=\rho\le r$. Let $f_0=k-1$ and $f_{\delta+1}=k+r-1$, and assume that there exists $\lambda\in\{0,1,\ldots,\delta\}$ such that $f_{\lambda+1}-f_\lambda\ge\gamma+1$. The decoding procedure can be divided into two cases, namely (i) $\lambda\ge 1$ and (ii) $\lambda=0$.

(i) $\lambda\ge 1$. First, we formulate $k-\gamma$ surviving information polynomials $b_i(x)$ for $i\in\mathcal{A}$ as

$$b_i(x):=b_{0,i}+b_{1,i}x+\cdots+b_{p-2,i}x^{p-2},$$

and $\gamma+1$ parity polynomials as

$$b_k(x):=b_{0,k}+b_{1,k}x+\cdots+b_{p-2,k}x^{p-2},$$

$$b_{f_\lambda+j}(x):=b_{0,f_\lambda+j}+\cdots+b_{p-2,f_\lambda+j}x^{p-2}+(\sum_{i=0}^{p-2}b_{i,f_\lambda+j})x^{p-1},$$

where $j=1,2,\ldots,\gamma$. Then, we compute γ syndrome polynomials $\bar{b}_1(x),\bar{b}_2(x),\ldots,\bar{b}_\gamma(x)$ by

$$\bar{b}_h(x)=b_{f_\lambda+h}(x)+b_k(x)x^{k(f_\lambda+h-k)}+\sum_{i\in\mathcal{A}}b_i(x)x^{i(f_\lambda+h-k)},$$

for $h=1,2,\ldots,\gamma$. It is easy to check that $\bar{b}_1(1)=\cdots=\bar{b}_\gamma(1)$. By the remark at the end of Section V-B, the erased information polynomials can be computed by first solving the following Vandermonde system of linear equations, namely

$$\begin{bmatrix}\bar{b}_1(x) & \cdots & \bar{b}_\gamma(x)\end{bmatrix}=$$

$$\left[x^{e_1(f_\lambda+1-k)}b_{e_1}(x) \quad \cdots \quad x^{e_\gamma(f_\lambda+1-k)}b_{e_\gamma}(x)\right]\mathbf{V}_{r\times r}(\mathbf{e}),$$

over R_p by Algorithm 1, cyclic-left shifting the resultant $x^{e_i(f_\lambda+1-k)}b_{e_i}(x)$ by $e_i(f_\lambda+1-k)$ positions for $i=1,2,\ldots,\gamma$, and then reducing $b_{e_1}(x)$ modulo $M_p(x)$.

(ii) $\lambda=0$. We have that columns $k,k+1,\ldots,k+\gamma$ are not erased. We can obtain γ syndrome polynomials $\bar{b}_1(x),\bar{b}_2(x),\ldots,\bar{b}_\gamma(x)$ by

$$\bar{b}_1(x)=b_k(x)+\sum_{i\in\mathcal{A}}b_i(x)$$

and

$$\bar{b}_h(x)=b_{k+h}(x)+b_k(x)x^{k\cdot(h-1)}+\sum_{i\in\mathcal{A}}b_i(x)x^{i\cdot(h-1)}$$

for $h=2,3,\ldots,\gamma$. The erased information polynomials can be computed by solving the following Vandermonde linear system, namely

$$\left[\bar{b}_1(x) \quad \cdots \quad \bar{b}_\gamma(x)\right]=\left[b_{e_1}(x) \quad \cdots \quad b_{e_\gamma}(x)\right]\cdot\mathbf{V}_{r\times r}(\mathbf{e}).$$

We need not reduce $b_{e_1}(x)$ modulo $M_p(x)$ according to the remark at the end of Section V-B because the last coefficient of $\bar{b}_1(x)$ is zero. Lastly, we can recover the δ parity columns by (5) for $i=0,1,\ldots,p-2$ and $\ell=f_1-k,\ldots,f_\delta-k$. Note that we need column k to obtain the syndrome polynomials in the decoding procedure, so column k is assumed to be a non-failure column.

If column k has failed and the number of information erasures does not exceed the number of continuous surviving parity columns, then we can recover the erased information columns by using the LU decoding method over $\mathbb{F}_2[x]/M_p(x)$. It can be shown that the LU decoding method over $\mathbb{F}_2[x]/M_p(x)$ requires less complexity compared with the Blaum–Roth decoding method [19]. Because of space limitations, we omit the computational comparison here.

Remark. In the erasure decoding, the assumption that there exists $\lambda\in\{0,1,\ldots,\delta\}$ such that $f_{\lambda+1}-f_\lambda\geq\gamma+1$ is necessary, otherwise we could not obtain the Vandermonde linear system and Algorithm 1 would not be applicable. A traditional decoding method such as Cramer's rule could be used to recover the failures if the assumption was not satisfied. In Section VII, we consider the decoding complexity for two codes when the assumption is satisfied.

Note that the proposed LU decoding method can also recover all erasure patterns in Blaum–Roth codes [19] and in other Vandermonde array codes such as STAR codes [14] and RTP codes [8]. It can be shown that the decoding complexity of the proposed method is much less than that of the Blaum–Roth decoding method [19] and the two fast decoding methods given in [20], [21] for Blaum–Roth codes. However, because of space limitations, we omit the details of the comparison here.

VII. Decoding Complexity

In this section, we evaluate the decoding complexity for EVENODD(p,k,r) and RDP(p,k,r). We determine the *normalized decoding complexity* as the ratio of the decoding complexity to the number of information bits.

Theorem 11. *Suppose that γ information columns and δ parity columns $f_1,\ldots,f_\delta$ are erased. Let $f_0=k-1$ and $f_{\delta+1}=k+r-1$, we assume that there exists $\lambda\in\{0,1,\ldots,\delta\}$ such that $f_{\lambda+1}-f_\lambda\geq\gamma+1$. We use Algorithm 1 to recover the γ information erasures and recover the failure parity columns by re-encoding the parity bits. The decoding complexity of* EVENODD(p,k,r) *is*

$$p(\gamma k+\frac{3\gamma^2}{4}-\frac{\gamma}{4}+\frac{5}{2})-\gamma k-\frac{\gamma^2}{4}-\frac{5\gamma}{4}-\frac{5}{2}+\delta(kp-k-1) \text{ when } \lambda>0, \tag{35}$$

$$p(\gamma k+\frac{3\gamma^2}{4}-\frac{\gamma}{4}-\frac{1}{2})-\gamma k-\frac{\gamma^2}{4}-\frac{5\gamma}{4}+\frac{1}{2}+\delta(kp-k-1) \text{ when } \lambda=0. \tag{36}$$

The decoding complexity of RDP(p,k,r) *is*

$$p(\gamma k+\frac{3\gamma^2}{4}-\frac{\gamma}{4}+\frac{3}{2})-\gamma k-\frac{\gamma^2}{4}-\frac{9\gamma}{4}-\frac{1}{2}+\delta k(p-2) \text{ when } \lambda>0, \tag{37}$$

$$p(\gamma k+\frac{3\gamma^2}{4}-\frac{\gamma}{4}-\frac{3}{2})-\gamma k-\frac{\gamma^2}{4}-\frac{9\gamma}{4}+\frac{7}{2}+\delta k(p-2) \text{ when } \lambda=0. \tag{38}$$

Proof. Consider the decoding process of EVENODD(p,k,r). When $\lambda>0$, we compute the bits $a'_{i,k+\ell}$ of the augmented array from EVENODD(p,k,r) by (8) and (9) for $\ell=f_\lambda+1-k,\ldots,f_\lambda+\gamma-k$. We first compute $\sum_{i=0}^{p-2}a_{i,k}$, and then we compute $a'_{p-1,k+\ell}$ by (8) and $a'_{i,k+\ell}$ by (9). Thus, the total number of XORs involved in computing the bits $a'_{i,k+\ell}$ is $2(p-1)\gamma+(p-2)$. We now obtain $k-\gamma$ information polynomials in (32) and γ parity polynomials in (33). Next, we subtract $k-\gamma$ surviving information polynomials from the γ parity polynomials to obtain γ syndrome polynomials by (34), which takes $\gamma(k-\gamma)(p-1)$ XORs. The γ information polynomials are obtained by solving the Vandermonde system of equations by using Algorithm 1, the computational complexity of which is

$$\gamma(\gamma-1)p+(\gamma-1)(p-3)+(\gamma-1)(\gamma-2)(3p-5)/4$$

according to Theorem 10. Because the last $\gamma-1$ output polynomials of Algorithm 1 are exactly the last $\gamma-1$ information polynomials of EVENODD(p,k,r), we need reduce only the first polynomial modulo $M_p(x)$, which takes at most $p-1$ XORs. The erased δ parity columns can be recovered by (2) and the complexity is $\delta(kp-k-1)$. Therefore, the decoding complexity of EVENODD(p,k,r) is (35) when $\lambda>0$.

When $\lambda=0$, there are two differences compared with the case of $\lambda>0$. First, we need compute only the bits of the augmented array for $\gamma-1$ parity columns, with complexity $2(p-1)(\gamma-1)+(p-2)$, because the bits in the first parity column of the augmented array are the same as those in the first parity column of EVENODD(p,k,r). Second, we need not reduce the first polynomial modulo $M_p(x)$ after solving the Vandermonde system. Therefore, the decoding complexity of $\lambda=0$ has $3p-3$ reduction XORs and results in (36).

In RDP(p,k,r), computing r syndrome polynomials takes

$$\gamma(p-2)+\gamma(k-\gamma+1)(p-1)$$

XORs when $\lambda\geq1$ and

$$(\gamma-1)(p-2)+(k-\gamma)(p-1)+(\gamma-1)(k-\gamma+1)(p-1)$$

XORs when $\lambda = 0$. Similar to EVENODD(p,k,r), the Vandermonde linear system can be solved by Algorithm 1 with complexity

$$\gamma(\gamma-1)p+(\gamma-1)(p-3)+(\gamma-1)(\gamma-2)(3p-5)/4$$

XORs. Reducing one polynomial modulo $M_p(x)$ takes at most $p-1$ XORs when $\lambda > 0$. The δ parity columns are recovered by (5), the complexity of which is $\delta k(p-2)$. Therefore, the total number of XORs involved in the decoding process results in (37) for $\lambda \geq 1$ and (38) for $\lambda = 0$. □

The Blaum–Roth decoding method [19] proposed for decoding Blaum–Roth codes is also applicable to the decoding of EVENODD(p,k,r). Suppose that γ information columns and δ parity columns $f_1,\ldots,f_\delta$ are erased with the assumption that there exists $\lambda \in \{0,1,\ldots,\delta\}$ such that $f_{\lambda+1}-f_\lambda \geq \gamma+1$. If one uses the Blaum–Roth decoding method to recover the information erasures and recover the failure parity columns by re-encoding the parity bits, the decoding complexity of EVENODD(p,k,r) is [30]

$$\gamma(k+\gamma)p+(3\gamma^2+0.5\gamma)p+\gamma^2-0.5\gamma+\delta(kp-k-1).$$

The Blaum–Roth decoding method cannot be used directly for the erasure decoding for RDP(p,k,r). However, one can first transform the λ parity columns of RDP(p,k,r) into the form of EVENODD(p,k,r) and then recover the erased information columns by the decoding method of EVENODD(p,k,r). Let $a_{i,j} = b_{i,j}$ for $i = 0,1,\ldots,p-2$ and $j = 0,1,\ldots,k-1$. That is, the information bits of RDP(p,k,r) and EVENODD(p,k,r) are the same. We then have $a_{i,k} = b_{i,k}$ by (1) and (4), and

$$\sum_{i=0}^{p-2} b_{i,k+\ell} + b_{p-1-\ell\cdot k,k} = \sum_{i=0}^{p-2}\sum_{j=0}^{k} b_{i-\ell\cdot j,j} + b_{p-1-\ell\cdot k,k} \tag{39}$$

$$\begin{aligned}
&= \sum_{i=0}^{p-1}\sum_{j=0}^{k} b_{i-\ell\cdot j,j} + \sum_{j=0}^{k} b_{p-1-\ell\cdot j,j} + b_{p-1-\ell\cdot k,k} \\
&= \sum_{j=0}^{k} b_{p-1-\ell\cdot j,j} + b_{p-1-\ell\cdot k,k} \qquad (40)\\
&= \sum_{j=0}^{k-1} b_{p-1-\ell\cdot j,j} = \sum_{j=0}^{k-1} a_{p-1-\ell\cdot j,j} = a_{p-1,k+\ell},
\end{aligned}$$

where (39) comes from (5), (40) comes from (4), and

$$\{-\ell\cdot j, 1-\ell\cdot j,\cdots,p-1-\ell\cdot j\} = \{0,1,\cdots,p-1\} \bmod p.$$

Therefore, when $\lambda > 0$, we can transform λ parity columns of RDP(p,k,r) into the form of EVENODD(p,k,r) by

$$a_{p-1,k+\ell} = \sum_{i=0}^{p-2} b_{i,k+\ell} + b_{p-1-\ell\cdot k,k}$$

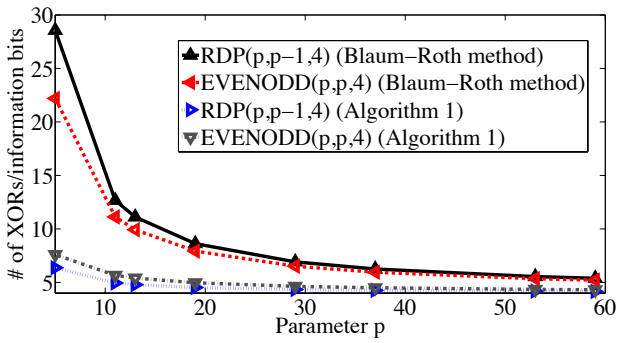
and

$$a_{i,k+\ell} = b_{i,k+\ell} + b_{i-\ell\cdot k,k} + a_{p-1,k+\ell}$$

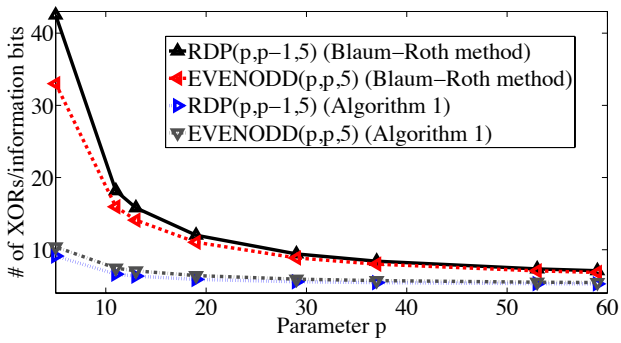
for $\ell = f_\lambda+1-k,\ldots,f_\lambda+\gamma-k$ and $i = 0,1,\ldots,p-2$. When $\lambda = 0$, we need transform only the bits for $\ell = 1,\ldots,\gamma-1$ and $i = 0,1,\ldots,p-2$ because column k of EVENODD(p,k,r) is the same as column k of RDP(p,k,r). Then we use the Blaum–Roth decoding method to obtain the erased γ information columns of EVENODD(p,k,r). Lastly, we recover the δ parity columns by (5). The decoding complexity is then

$$\gamma(k+\gamma)p+(3\gamma^2+3.5\gamma)p+\gamma^2-0.5\gamma+\delta(kp-2k)-3 \text{ for } \lambda > 0,$$
$$\gamma(k+\gamma)p+(3\gamma^2+3.5\gamma-3)p+\gamma^2-3.5\gamma+\delta(kp-2k)+3 \text{ for } \lambda = 0.$$

Note that we can recover the erased parity columns by encoding the parity bits according to the definition for both EVENODD(p,k,r) and RDP(p,k,r) after recovering all the erased information bits. Therefore, the main difference in decoding complexity between the proposed LU decoding method and the Blaum–Roth decoding method lies in the complexity of decoding the Vandermonde linear system, that is, the erasure decoding of information failures. In the following, we consider the special case of $\delta = 0$. We evaluate the decoding complexity of γ information erasures for the proposed LU decoding method and the Blaum–Roth decoding method.



(a) $r = 4$ and p ranges from 5 to 59.



(b) $r = 5$ and p ranges from 5 to 59.

Fig. 1: Normalized decoding complexities of $\gamma = r$ information erasures EVENODD(p,p,r) and RDP$(p,p-1,r)$ by Algorithm 1 and by Blaum–Roth decoding method for $r = 4,5$.

For fair comparison, we let $k = p$ for EVENODD(p,k,r) and $k = p-1$ for RDP(p,k,r). When $r = 3$, by Theorem 11, the decoding complexity of the proposed method

for $\mathsf{EVENODD}(p, p, 3)$ is at most $3p^2 + 2.5p - 5.5$. Recall that the decoding complexity for extended EVENODD codes given in [15] is $3p^3 + 7p - 19$, which is larger than that of the proposed method. The decoding complexity for STAR codes given in [14] is $(3p + 2\ell_d + \ell_h)(p-1)$, where the parameters $0 \leq \ell_d, \ell_h < p$ depend on the erasure patterns. The proposed method has lower decoding complexity than that of the decoding method given in [14] for STAR codes when $\ell_d \geq 2$.

By Theorem 11, the decoding complexity of the proposed method for $\mathsf{RDP}(p, p-1, 3)$ is $3p^2 - 1.5p - 2.5$, whereas the decoding complexity for RTP codes given in [8] is $(3p + \ell_1 - 1)(p-1) - 1$, where $1 \leq \ell_1 \leq p-1$. When $\ell_1 \geq 2$, the decoding complexity given in [8] is larger than that of the proposed method. Although the erasure decoding for RDP codes given in [17] is optimized with lower complexity compared with the proposed method, the decoding method is focused only on the specific codes with $r = 3$ and cannot be generalized to $r \geq 4$.

Next, we evaluate the decoding complexity for more than three information erasures. According to Theorem 11, the decoding complexities of γ information erasures of $\mathsf{EVENODD}(p, p, r)$ and $\mathsf{RDP}(p, p-1, r)$ by Algorithm 1 are

$$p(\gamma p + \frac{3\gamma^2}{4} - \frac{5\gamma}{4} - \frac{1}{2}) - \frac{\gamma^2}{4} - \frac{5\gamma}{4} + \frac{1}{2} \text{ and}$$

$$p(\gamma(p-1) + \frac{3\gamma^2}{4} - \frac{5\gamma}{4} - \frac{3}{2}) - \frac{\gamma^2}{4} - \frac{5\gamma}{4} + \frac{7}{2},$$

respectively.

When $r = 4$ and 5, the normalized decoding complexities of $\gamma = r$ information erasures of $\mathsf{EVENODD}(p, p, r)$ and $\mathsf{RDP}(p, p-1, r)$ by Algorithm 1 and by the Blaum–Roth decoding method are shown in Fig. 1. One can observe that decoding $\mathsf{EVENODD}(p, p, r)$ and $\mathsf{RDP}(p, p-1, r)$ by the LU decoding method is more efficient than by the Blaum–Roth decoding method. When $r = 4$ and p ranges from 5 to 59, the complexity of decoding $\mathsf{EVENODD}(p, p, 4)$ and $\mathsf{RDP}(p, p-1, 4)$ by Algorithm 1 is reduced by 20.1% to 71.3% and 22.7% to 77.7%, respectively, compared with that by the Blaum–Roth decoding method. When $r = 5$, the complexity reductions are 20.4% to 68.5% and 25.7% to 78.5% for $\mathsf{EVENODD}(p, p, 5)$ and $\mathsf{RDP}(p, p-1, 5)$, respectively. The reduction increases when p is small and r is large. For example, $\mathsf{RDP}(p, p-1, r)$ decoded by Algorithm 1 has 78.5% less decoding complexity than that by the Blaum–Roth decoding method when $p = 5$ and $r = 5$.

The reasons why the decoding complexity of $\mathsf{EVENODD}(p, p, r)$ and $\mathsf{RDP}(p, p-1, r)$ by the LU decoding method is lower than that by the Blaum–Roth decoding method are summarized as follows. First, the Blaum–Roth decoding method is operated over the ring $\mathbb{F}_2[x]/M_p(x)$; however, we show that the Vandermonde linear systems over $\mathbb{F}_2[x]/M_p(x)$ can be computed by first solving the Vandermonde linear systems over R_p and then reducing the results by modulo $M_p(x)$.[4] The operations of multiplication and division over R_p are more efficient than those over $\mathbb{F}_2[x]/M_p(x)$. Second, the proposed LU decoding method is more efficient than the Blaum–Roth decoding method.

In conclusion, our LU decoding method requires lower decoding complexity for EVENODD codes with $r \geq 3$ compared with the existing decoding method; for RDP codes with $r \geq 4$, our LU decoding method has lower decoding complexity compared with the existing decoding method.

[4]When there are only information erasures, modulo $M_p(x)$ is not needed in the decoding procedure.

VIII. Discussion and Conclusions

In this paper, we present a unified construction of EVENODD codes and RDP codes, which can be viewed as a generalization of extended EVENODD codes and generalized RDP codes. Moreover, an efficient LU decoding method is proposed for EVENODD codes and RDP codes. We show that the LU decoding method is applicable to (i) the erasure decoding of EVENODD codes and RDP codes with more than three parity columns if the number of continuous surviving parity columns does not exceed the number of erased information columns and the first parity column has not failed and (ii) all erasure patterns of EVENODD codes and RDP codes with three parity columns. For EVENODD codes with more than two parity columns, we show that the LU decoding method requires fewer XOR operations compared with all existing decoding methods. For RDP codes with more than three parity columns, we show that the decoding complexity of our decoding method is lower than that of all existing decoding methods. Furthermore, our LU decoding method is also applicable to the erasure decoding of other Vandermonde MDS array codes such as Blaum–Roth codes [19], STAR codes [14], and RTP codes [8], and it has lower decoding complexity compared with the existing methods.

Future work will include reducing the decoding complexity of the LU decoding method further and an efficient decoding method for all the erasure patterns of Vandermonde MDS array codes. Implementing the proposed method with a real system will also be interesting future work.

References

[1] D. A. Patterson, P. Chen, G. Gibson, and R. H. Katz, "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," in *Proc. IEEE COMPCON*, vol. 89, 1989, pp. 112–117.

[2] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: high-performance, reliable secondary storage," University of California at Berkeley, Berkeley, Tech. Rep. CSD 03-778, 1993.

[3] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Computers*, vol. 44, no. 2, pp. 192–202, 1995.

[4] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proc. of the 3rd USENIX Conf. on File and Storage Technologies (FAST)*, 2004, pp. 1–14.

[5] A. H. Leventhal, "Triple-parity RAID and beyond," *Comm. of the ACM*, vol. 53, no. 1, pp. 58–63, January 2010.

[6] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Trans. Information Theory*, vol. 42, no. 2, pp. 529–542, 1996.

[7] M. Blaum, J. Brady, J. Bruck, J. Menon, and A. Vardy, "The EVENODD code and its generalization: An effcient scheme for tolerating multiple disk failures in RAID architectures," in *High Performance Mass Storage and Parallel I/O*. Wiley-IEEE Press, 2002, ch. 8, pp. 187–208.

[8] A. Goel and P. Corbett, "RAID triple parity," in *ACM SIGOPS Operating Systems Review*, vol. 36, no. 3, December 2012, pp. 41–49.

[9] M. Blaum, "A family of MDS array codes with minimal number of encoding operations," in *IEEE Int. Symp. on Inf. Theory*, 2006, pp. 2784–2788.
[10] Z. Wang, A. G. Dimakis, and J. Bruck, "Rebuilding for array codes in distributed storage systems," in *IEEE GLOBECOM Workshops (GC Wkshps), 2010*, pp. 1905–1909.
[11] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage systems," in *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 38, no. 1. ACM, 2010, pp. 119–130.
[12] L. Xiang, Y. Xu, J. C. S. Lui, Q. Chang, Y. Pan, and R. Li, "A hybrid approach of failed disk recovery using RAID-6 codes: Algorithms and performance evaluation," *ACM Trans. on Storage*, vol. 7, no. 3, pp. 1–34, October 2011.
[13] Y. Zhu, P. P. C. Lee, Y. Xu, Y. Hu, and L. Xiang, "On the speedup of recovery in large-scale erasure-coded storage systems," *IEEE Transactions on Parallel & Distributed Systems*, vol. 25, no. 7, pp. 1830–1840, 2014.
[14] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Trans. Computers*, vol. 57, no. 7, pp. 889–901, 2008.
[15] H. Jiang, M. Fan, Y. Xiao, X. Wang, and Y. Wu, "Improved decoding algorithm for the generalized EVENODD array code," in *International Conference on Computer Science and Network Technology*, 2013, pp. 2216–2219.
[16] Y. Wang, G. Li, and X. Zhong, "Triple-Star: A coding scheme with optimal encoding complexity for tolerating triple disk failures in RAID," *International Journal of innovative Computing, Information and Control*, vol. 3, pp. 1731–1472, 2012.
[17] Z. Huang, H. Jiang, and K. Zhou, "An improved decoding algorithm for generalized RDP codes," *IEEE Communications Letters*, vol. 20, no. 4, pp. 632–635, 2016.
[18] H. Hou, K. W. Shum, and H. Li, "On the MDS condition of Blaum-Bruck-Vardy codes with large number parity columns," *IEEE Communications Letters*, vol. 20, no. 4, pp. 644–647, 2016.
[19] M. Blaum and R. M. Roth, "New array codes for multiple phased burst correction," *IEEE Trans. Information Theory*, vol. 39, no. 1, pp. 66–77, January 1993.
[20] Q. Guo and H. Kan, "On systematic encoding for Blaum-Roth codes," *Development*, vol. 42, no. 4, pp. 2353–2357, 2011.
[21] ——, "An efficient interpolation-based systematic encoder for low-rate Blaum-Roth codes," in *IEEE Int. Symp. on Inf. Theory*, 2013, pp. 2384–2388.
[22] H. Hou, Y. S. Han, K. W. Shum, and H. Li, "A unified form of EVENODD and RDP codes and their efficient decoding," https://arxiv.org/pdf/1803.03508.pdf, 2018.
[23] K. W. Shum, H. Hou, M. Chen, H. Xu, and H. Li, "Regenerating codes over a binary cyclic code," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, July 2014, pp. 1046–1050.
[24] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC codes: Low-complexity regenerating codes for distributed storage systems," *IEEE Trans. Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.
[25] S. T. J. Fenn, M. G. Parker, M. Benaissa, and D. Taylor, "Bit-serial multiplication in GF(2^m) using irreducible all-one polynomials," *IEEE Proceedings on Computers and Digital Techniques*, vol. 144, no. 6, pp. 391–393, 1997.
[26] J. H. Silverman, "Fast multiplication in finite fields GF(2^n)," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 1999, pp. 122–134.
[27] H. Hou and Y. S. Han, "A new construction and an efficient decoding method for Rabin-Like codes," *IEEE Trans. Communications*, vol. 66, no. 2, pp. 521–533, 2018.
[28] H. Hou, K. W. Shum, M. Chen, and H. Li, "New MDS array code correcting multiple disk failures," in *Global Communications Conference*, 2014, pp. 2369–2374.
[29] S.-L. Yang, "On the LU factorization of the Vandermonde matrix," *Discrete applied mathematics*, vol. 146, no. 1, pp. 102–105, 2005.
[30] P. Subedi and X. He, "A comprehensive analysis of XOR-based erasure codes tolerating 3 or more concurrent failures," in *IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, 2013, pp. 1528–1537.

Hanxu Hou (S'11-M'16) was born in Anhui, China, 1987. He received the B.Eng. degree in Information Security from Xidian University, Xian, China, in 2010, and Ph.D. degrees in the Dept. of Information Engineering from The Chinese University of Hong Kong in 2015 and in the School of Electronic and Computer Engineering, Peking University. He is now an Assistant Professor with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology and part-time Research Fellow with the Shenzhen Key Lab of Information Theory & Future Internet Architecture, Future Network PKU Lab of National Major Research Infrastructure, Peking University Shenzhen Graduate School. His research interests include erasure coding and coding for distributed storage systems.

Yunghsiang S. Han (S'90-M'93-SM'08-F'11) was born in Taipei, Taiwan, 1962. He received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993. He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004 and July 2012 to June 2013, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010 to January 2017, he was with the Department of Electrical Engineering at National Taiwan University of Science and Technology as Chair Professor. Now he is with School of Electrical Engineering & Intelligentization at Dongguan University of Technology, China. He is also a Chair Professor at National Taipei University from February 2015. His research interests are in error-control coding, wireless networks, and security.

Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize and a Fellow of IEEE. One of his papers won the prestigious 2013 ACM CCS Test-of-Time Award in cybersecurity.

Kenneth W. Shum (M00-SM16) received his B.Eng. degree in the Department of Information Engineering from The Chinese University of Hong Kong in 1993, and MSc and Ph.D. degrees in Department of Electrical Engineering from University of Southern California in 1995 and 2000, respectively. He is now a Research Associate Professor with Institute of Network Coding in The Chinese University of Hong Kong. His research interests include coding for distributed storage systems and sequence design for wireless networks.

Hui Li received the B.Eng. and M.S. degrees from School of Information Eng., Tsinghua University, Beijing, China, in 1986 and 1989 respectively, and Ph.D. degree in the Dept. of Information Engineering from The Chinese University of Hong Kong in 2000. He is now a Full Professor of Peking University, Shenzhen Graduate School. He is Director of Shenzhen Key Lab of Information theory & Future Internet architecture, Future Network PKU Lab of National Major Research Infrastructure, Shenzhen Eng. Lab of Converged Networks. His research interests include future network architecture, cyberspace security, distributed storage, and blockchain.