Punctured Shift-XOR Storage Codes

Yuanxin Guo*, Ximing Fu*†, Shenghao Yang* and Kenneth W. Shum*
*The Chinese University of Hong Kong, Shenzhen
†University of Science and Technology of China

***Abstract*—Shift-XOR storage codes satisfying the refined increasing difference (RID) property have the optimal decoding bandwidth costs and low decoding computation costs. However, a shift-XOR code has more total bits to store than an MDS code of the same parameter, which is called the storage overhead. In this paper, we propose *punctured shift-XOR codes*, which have a smaller storage overhead and lower encoding complexity compared with the corresponding shift-XOR storage codes satisfying the RID property. Punctured shift-XOR codes preserve the same decoding space and time complexity as the shift-XOR codes with the same generator matrix. We also show that certain Vandermonde generator matrices achieve the minimum storage overhead for both punctured shift-XOR codes and shift-XOR codes satisfying the RID property.**

I. Introduction

Storage codes can provide better trade-off between redundancy and reliability than repetition, and have witnessed extensive developments in the past decade with new applications, e.g., secret sharing and block chains. Reed-Solomon codes [1] are widely applied in storage systems. However, the high encoding/decoding computation costs due to finite field operations prevent Reed-Solomon codes from being applied in systems with low computation power and low latency requirements.

One approach for constructing low computation complexity storage codes is to employ (cyclic) shift and bitwise eXclusive-OR (XOR) operations instead of finite field operations. One class of codes uses cyclic shift and XOR operations, including EVENODD codes [2], Cauchy Reed-Solomon [3], Rabin-like codes [4], RDP codes [5] and their extensions [6]–[8]. Another class of codes proposed recently uses shift and XOR operations, called *shift-XOR codes* [9]–[11]. Codes using shift and XOR operations potentially have lower encoding/decoding complexity than codes using cyclic shift and XOR [10], and generates a broad interest for coding design [12]–[15].

Shift-XOR codes, however, have an intrinsic overhead issue due to the shift operation. Consider a storage code encoding k message sequences each of L bits into n storage nodes, which is also called an (n, k) code. The message sequences are required to be decoded from the storage content at any k storage nodes. To satisfy the above requirement, the minimal storage cost among all the n nodes is nL bits, which can be achieved by MDS codes, for example, Reed-Solomon codes. For a particular (n, k) storage code, define the *storage overhead* as the total number of the bits stored at the n storage nodes minus nL. Using shift-XOR codes, the number of bits stored at a storage node is more than L if the shift operation is applied when encoding the storage content. The storage overhead is always positive in the existing designs of shift-XOR codes, though it can be relatively small when L is large.

The storage overhead does not necessarily generate the bandwidth overhead for decoding, which is defined as the number of bits retrieved for decoding minus kL. Suppose an (n, k) shift-XOR code satisfies the *refined increasing difference (RID) property*. By retrieving subsequences of L bits from any k storage nodes, the k message sequences can be decoded using an algorithm called *shift-XOR elimination* [15], [16]. The shift-XOR elimination has the *in-place* property that it can store the output sequences in the same kL bit storage space of the input sequences without using any auxiliary space for intermediate XOR results.

In this paper, we focus on the storage overhead issue of shift-XOR codes satisfying the RID properties. We show that when the RID property is satisfied, the storage overhead of an (n, k) shift-XOR code is at least $\frac{1}{2}n(n-1)(k-1)$ bits, which can be achieved by certain Vandermonde generator matrices. For shift-XOR codes satisfying the RID property, we observe that some bits in the coded sequences are never used in the shift-XOR elimination process. Motivated by this observation, we propose a class of codes called *punctured shift-XOR codes* to reduce the storage overhead as well as the encoding complexity of shift-XOR codes satisfying the RID property. Our codes can also be decoded using the shift-XOR elimination and hence preserve the same decoding complexity and the same decoding capability, in other words, the message sequences can be decoded by retrieving kL bits from any k out of the n nodes.

We further show that an (n, k) punctured shift-XOR code can have at least $\frac{1}{2}k(k-1)(n-1)$ bits smaller storage overhead than that of the corresponding shift-XOR code with the same generator matrix. For the same Vandermonde generator matrix, the overhead of the punctured shift-XOR code is $1 - \frac{k}{n}$ fraction of that of the shift-XOR code. When L is larger than the largest distance of shifting in encoding (which implies $L > (n-1)(k-1)$), the encoding of an (n, k) punctured shift-XOR code takes at most $n(k-1)L - \frac{1}{6}(n-1)(k-1)k(k+1)$ XOR operations, while the encoding of the shift-XOR code with the same generator matrix takes at most $n(k-1)L$ XOR operations. Moreover, we prove that certain Vandermonde generator matrices achieve the minimum storage overhead for punctured shift-XOR codes. We compare our codes with the shift-XOR codes satisfying the RID property in Table I.

The remainder of the paper is organized as follows. Shift-XOR codes and the storage overhead will be discussed in Section II. A detailed description of the punctured shift-XOR

TABLE I

COMPARISON BETWEEN (n,k) PUNCTURED SHIFT-XOR CODES AND (n,k) SHIFT-XOR CODES SATISFYING THE RID PROPERTY. A VANDERMONDE GENERATOR MATRIX $\mathbf{\Psi} = (z^{(i-1)(j-1)})$ IS ASSUMED HERE. L IS THE LENGTH OF THE MESSAGE SEQUENCES WITH $L > (n-1)(k-1)$.

Codes	Storage Overhead	Encoding Cost
Punctured shift-XOR codes (this paper)	$\frac{1}{2}(n-k)(n-1)(k-1)$	$\leq n(k-1)L - \frac{1}{6}(n-1)(k-1)k(k+1)$
Shift-XOR codes [15], [16]	$\frac{1}{2}n(n-1)(k-1)$	$\leq n(k-1)L$

codes will be given in Section III. We conclude our paper in Section IV.

II. SHIFT-XOR CODES AND STORAGE OVERHEAD

A. Notation

We introduce some notations following those in [15]. Denote binary sequences as boldface, lowercase letters, e.g, $\boldsymbol{a}$. For a binary sequence $\boldsymbol{a}$ of L bits, the i-th entry $(1 \leq i \leq L)$ is denoted by $\boldsymbol{a}[i]$. By convention, $\boldsymbol{a}[i] = 0$ if $i < 1$ or $i > L$. For integers $i \leq j$, the symbol $[i:j]$ represents the vector $(i, i+1, \ldots, j)$. When $i > j$, $[i:j]$ is the empty sequence $\emptyset$ by convention. The subsequence of $\boldsymbol{a}$ from the i-th entry to the j-th entry is denoted by $\boldsymbol{a}[i:j]$.

For a sequence $\boldsymbol{a}$ of L bits and an integer $t \geq 0$, the *right-shift operator* z^t pads t zeros in front of $\boldsymbol{a}$, so that $z^t\boldsymbol{a}$ is a binary sequence containing $(L+t)$ bits with

$$(z^t\boldsymbol{a})[m] = \boldsymbol{a}[m-t], \quad m = 1, \ldots, L+t.$$

For two binary sequences $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ with length L_1 and L_2, respectively, where L_1 may not be equal to L_2, the sum $\boldsymbol{a}_1 + \boldsymbol{a}_2$ is defined as a binary sequence of length $\max(L_1, L_2)$ obtained by bitwise eXclusive-OR (XOR)

$$(\boldsymbol{a}_1 + \boldsymbol{a}_2)[m] = \boldsymbol{a}_1[m] \oplus \boldsymbol{a}_2[m], \ m = 1, \ldots, \max(L_1, L_2),$$

where $\oplus$ denotes the XOR operator of two bits.

B. Shift-XOR Codes

We briefly introduce the storage codes using shift and XOR operations, called *shift-XOR codes*. Consider an (n,k) shift-XOR code that encodes k message sequences to n coded sequences which are stored in a system of n storage nodes. Denote the k message sequences by $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$, each of which consists of L bits. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$, let $t_{i,j}$ be non-negative integers corresponding to the distance of the right-shift operation of the j-th message sequence in encoding the i-th coded sequence. The k message sequences $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ are encoded into n coded sequences $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ by

$$\boldsymbol{y}_i = \sum_{j=1}^{k} z^{t_{i,j}} \boldsymbol{x}_j. \tag{1}$$

Each coded sequence $\boldsymbol{y}_i$ is then stored in the i-th node of an n-node distributed storage system.

The encoding function can be represented as a matrix multiplication. Denote by $\boldsymbol{X}$ and $\boldsymbol{Y}$ the column vectors with the i-th entries $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, respectively, i.e.,

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_k \end{pmatrix}, \quad \boldsymbol{Y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_n \end{pmatrix}.$$

Let $\mathbf{\Psi} = (z^{t_{i,j}})$ be the $n \times k$ generator matrix whose (i,j) entry is $z^{t_{i,j}}$, i.e.,

$$\mathbf{\Psi} = \begin{pmatrix} z^{t_{1,1}} & z^{t_{1,2}} & \ldots & z^{t_{1,k}} \\ z^{t_{2,1}} & z^{t_{2,2}} & \ldots & z^{t_{2,k}} \\ \vdots & \vdots & \ddots & \vdots \\ z^{t_{n,1}} & z^{t_{n,2}} & \ldots & z^{t_{n,k}} \end{pmatrix}.$$

The encoding algorithm can now be written in a matrix-vector multiplication form

$$\boldsymbol{Y} = \mathbf{\Psi}\boldsymbol{X}.$$

When talking about a shift-XOR code, a generator matrix is supposed to be associated with it.

Consider the above encoding process of an (n,k) shift-XOR code. It is worth noting that the entries of $\boldsymbol{Y}$ are not of the same length due to the shift operation. The i-th coded sequence $\boldsymbol{y}_i$ has $L + \max_{j=1}^{k} t_{i,j}$ bits, and the total number of bits stored at all the n nodes is $nL + \sum_{i=1}^{n} \max_{j=1}^{k} t_{i,j}$ bits. We call

$$N_s(\mathbf{\Psi}) \triangleq \sum_{i=1}^{n} \max_{j=1}^{k} t_{i,j} \tag{2}$$

the *storage overhead* of the (n,k) shift-XOR code with the generator matrix $\mathbf{\Psi}$. Note that the storage overhead does not depend on L, and hence when L is large, the storage overhead is relatively small.

Example 1. Consider the $(5,3)$ shift-XOR code, where 5 coded sequences $\boldsymbol{y}_i$, $i = 1, \ldots, 5$ are generated from 3 message sequences $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ using a Vandermonde matrix as the generator matrix:

$$\mathbf{\Psi} = \begin{bmatrix} 1 & z & z^2 \\ 1 & z^2 & z^4 \\ 1 & z^3 & z^6 \\ 1 & z^4 & z^8 \\ 1 & z^5 & z^{10} \end{bmatrix}.$$

For $i = 1, \ldots, 5$, we have

$$\boldsymbol{y}_i = \boldsymbol{x}_1 + z^i\boldsymbol{x}_2 + z^{2i}\boldsymbol{x}_3.$$

The lengths of the coded sequences are $L+2$, $L+4$, $L+6$, $L+8$, and $L+10$, respectively. Summing up, the storage system requires $5L+30$ bits for storing the message in total, i.e., the storage overhead of the code is 30 bits.

C. Shift-XOR Elimination

In this paper, we focus on shift-XOR codes with generator matrices satisfying the following property such that an efficient decoding algorithm can be applied.

Definition 1. A generator matrix $\mathbf{\Psi} = (z^{t_{i,j}})$ is said to satisfy the *refined increasing difference (RID) property* [16] if for any i, i', j, j' such that $i < i'$ and $j < j'$,

$$0 \leq t_{i,j'} - t_{i,j} < t_{i',j'} - t_{i',j}. \tag{3}$$

A shift-XOR code is said to satisfy the RID property if its generator matrix satisfies the RID property.

A decoding approach free of decoding overhead is proposed for shift-XOR codes satisfying the RID property [15], [16], where exactly kL bits are needed from the storage nodes to decode the message sequences. The decoding process is divided into two stages: transmission and decoding. In the transmission stage, k nodes with indices $i_1, i_2, \ldots, i_k$ are chosen, where $i_1 > i_2 > \cdots > i_k$. For $1 \leq u \leq k$. The decoder retrieves a subsequence $\hat{\boldsymbol{x}}_u$ of $\boldsymbol{y}_{i_u}$ of L bits:

$$\hat{\boldsymbol{x}}_u \triangleq \boldsymbol{y}_{i_u}[(t_{i_u,u}+1) : (t_{i_u,u}+L)]. \tag{4}$$

Clearly, the total number of bits retrieved by the decoder is kL so that the decoding overhead is zero.

After retrieving the kL bits from the storage nodes, an algorithm called *shift-XOR elimination* is used to decode the message sequences [15], [16], which has both lower space and time complexities compared with the zigzag decoding in [9]. The shift-XOR elimination uses the same number of XOR operations as that for encoding the input subsequences from the message sequences. Moreover, the shift-XOR elimination is *in-place* as it can be implemented to store the output sequences in the same kL bit storage space of the input sequences without using any auxiliary space for intermediate XOR results.

We can check that the generator matrix in Example 1 satisfies the RID property, and hence the code can be decoded using the algorithm mentioned above. In the following of this subsection, we characterize the storage overheads of shift-XOR codes satisfying the RID property. For an $n \times k$ generator matrix $\mathbf{\Psi} = (z^{t_{i,j}})$ satisfying the RID property, we have $\max_{j=1}^{k} t_{i,j} = t_{i,k}$ for all i, and hence the storage overhead $N_s(\mathbf{\Psi})$ defined in (2) becomes

$$N_s(\mathbf{\Psi}) = \sum_{i=1}^{n} t_{i,k}. \tag{5}$$

Lemma 1. *For an generator matrix $\mathbf{\Psi} = (z^{t_{i,j}})$ satisfying the RID property, we have for all i and $1 \leq j < j' \leq k$,*

$$t_{i,j'} - t_{i,j} \geq (i-1)(j'-j), \tag{6}$$

where the equality holds if and only if

$$t_{i,a+1} - t_{i,a} = i-1, \quad \forall a = j, \ldots, j'-1. \tag{7}$$

Proof. By the definition of refined increasing difference property (3):

$$0 \leq t_{a,b'} - t_{a,b} < t_{a',b'} - t_{a',b}, \tag{8}$$

for all $1 \leq a < a' \leq n$, $1 \leq b < b' \leq n$. By the second inequality in (8) with $a' = i$, $a = i-1$, $b' = j+1$, $b = j$, we have

$$t_{i,j+1} - t_{i,j} \geq t_{i-1,j+1} - t_{i-1,j} + 1.$$

Repeating the above argument, we eventually get

$$t_{i,j+1} - t_{i,j} \geq t_{1,j+1} - t_{1,j} + i - 1.$$

Now since $t_{1,j+1} - t_{1,j} \geq 0$ by the first inequality in (8), we have

$$t_{i,j+1} - t_{i,j} \geq i-1. \tag{9}$$

Similarly, we can derive inequalities

$$t_{i,a+1} - t_{i,a} \geq i-1, \quad \forall a = j+1, \ldots, j'-1. \tag{10}$$

By a telescoping sum of these inequalities in (9) and (10), we obtain (6).

The above argument also implies that if the equality in (6) holds, then the equality holds for all the inequalities in (9) and (10). To complete the proof, when (7) holds, we obtain (6) by summing the equalities in (7) for a from j to $j'-1$. □

Theorem 2. *The storage overhead of an (n,k) shift-XOR code satisfying the RID property is at least $\frac{1}{2}(k-1)(n-1)n$. The lower bound is achieved if and only if the generator matrix is the Vandermonde matrix $\mathbf{\Psi} = (z^{(i-1)(j-1)})$, i.e.,*

$$\mathbf{\Psi} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 1 & z & \ldots & z^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z^{n-1} & \ldots & z^{(n-1)(k-1)} \end{pmatrix}.$$

Proof. Following (5), when the generator matrix $\mathbf{\Psi} = (z^{t_{i,j}})$, we have

$$\begin{aligned} N_s(\mathbf{\Psi}) &= \sum_{i=1}^{n} t_{i,k} \geq \sum_{i=1}^{n} (t_{i,k} - t_{i,1}) \\ &\geq \sum_{i=1}^{n} (i-1)(k-1) = \frac{1}{2}(k-1)(n-1)n, \end{aligned}$$

where the first inequality follows from $t_{i,1} \geq 0$, and the second inequality follows from Lemma 1.

To have equality in both inequalities above, the sufficient and necessary condition is

$$t_{i,1} = 0, \quad \forall 1 \leq i \leq n, \tag{11}$$

and

$$t_{i,k} - t_{i,1} = (i-1)(k-1), \quad \forall 1 \leq i \leq n, \tag{12}$$

TABLE II
INPUT SEQUENCES TO SHIFT-XOR ELIMINATION FOR DIFFERENT CHOICES OF THE STORAGE NODES USING $(5,3)$ SHIFT-XOR CODE.

indices	$\boldsymbol{y}_1$	$\boldsymbol{y}_2$	$\boldsymbol{y}_3$	$\boldsymbol{y}_4$	$\boldsymbol{y}_5$
$1,2,3$	$[3:L+2]$	$[3:L+2]$	$[1:L]$	$\emptyset$	$\emptyset$
$1,2,4$	$[3:L+2]$	$[3:L+2]$	$\emptyset$	$[1:L]$	$\emptyset$
$1,2,5$	$[3:L+2]$	$[3:L+2]$	$[1:L]$	$[1:L]$	$[1:L]$
$1,3,4$	$[3:L+2]$	$\emptyset$	$[4:L+3]$	$[1:L]$	$\emptyset$
$1,3,5$	$[3:L+2]$	$\emptyset$	$[4:L+3]$	$\emptyset$	$[1:L]$
$1,4,5$	$[3:L+2]$	$\emptyset$	$\emptyset$	$[5:L+4]$	$[1:L]$
$2,3,4$	$\emptyset$	$[5:L+4]$	$[4:L+3]$	$[1:L]$	$\emptyset$
$2,3,5$	$\emptyset$	$[5:L+4]$	$[4:L+3]$	$\emptyset$	$[1:L]$
$2,4,5$	$\emptyset$	$[5:L+4]$	$\emptyset$	$[5:L+4]$	$[1:L]$
$3,4,5$	$\emptyset$	$\emptyset$	$[7:L+6]$	$[5:L+4]$	$[1:L]$

which follows from the inequality in Lemma 1. By the necessary and sufficient condition such that equality holds in Lemma 1, (12) is equivalent to

$$t_{i,j+1} - t_{i,j} = i - 1, \ \forall 1 \le i \le n, \ \forall 1 \le j < k. \tag{13}$$

By (11) and (13), $t_{i,j} = (i-1)(j-1)$ for all i and j, and hence $\boldsymbol{\Psi} = (z^{(i-1)(j-1)})$. □

III. PUNCTURED SHIFT-XOR CODES

In this section, we propose a new class of shift-XOR codes, called *punctured shift-XOR codes*, that have lower encoding computation costs and lower storage overheads compared with shift-XOR codes satisfying the RID property. Punctured shift-XOR codes can also be decoded using the shift-XOR elimination, and hence have the same decoding complexity and decoding capability as shift-XOR codes satisfying the RID property.

A. A Motivating Example

The idea of punctured shift-XOR codes comes from the observation that some bits in the coded sequences $\boldsymbol{y}_i$ generated by a shift-XOR code satisfying the RID property are not required in the shift-XOR elimination for any choice of the storage nodes, and hence it is not necessary to store all the bits in $\boldsymbol{y}_i$ to guarantee decodability. We use an example to illustrate this observation.

Example 2. Consider a storage system using the $(5,3)$ shift-XOR code with the Vandermonde generator matrix as in Example 1. To decode using the shift-XOR elimination, exactly L bits are used from each of the 3 chosen storage nodes. For example, if nodes $2,3$ and 5 are chosen, the decoder only needs $\boldsymbol{y}_2[4:(L+4)]$, $\boldsymbol{y}_3[3:(L+3)]$ and $\boldsymbol{y}_5[1:L]$. Table II lists the subsequences to retrieve for all the 10 combinations of 3 out of 5 storage nodes. Each row of Table II indicates a choices of the three nodes and the corresponding subsequences retrieved for decoding. For example, we see from the second column of Table II that node 1 only needs to store the subsequence $\boldsymbol{y}_1[3:(L+2)]$ as other bits are never used. Likewise, nodes $2,3,4,5$ can store only $\boldsymbol{y}_2[3:(L+4)]$, $\boldsymbol{y}_3[1:(L+6)]$, $\boldsymbol{y}_4[1:(L+4)]$, and $\boldsymbol{y}_5[1:L]$, respectively. In total, $5L+12$ bits are required. The storage overhead of the code is reduced from 30 bits to 12 bits.

B. Encoding of Punctured Shift-XOR Codes

We describe an (n,k) punctured shift-XOR code C that encodes message sequences $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k$ to coded sequences $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n$ stored in a storage system of n nodes. Code C has an $n \times k$ generator matrix $\boldsymbol{\Psi} = (z^{t_{i,j}})$ satisfying the RID property where $t_{i,1} = 0$ for $i = 1,\ldots,n$.

1) Two-step Encoding: We first introduce a two-step encoding approach. In the first step, the n coded sequences $\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_n$ are generated using the same approach as in the shift-XOR code C' with the generator matrix $\boldsymbol{\Psi}$. The second encoding step is to puncture the coded sequences $\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_n$. Let $\tilde{\boldsymbol{y}}_1,\tilde{\boldsymbol{y}}_2,\ldots,\tilde{\boldsymbol{y}}_n$ denote the punctured coded sequences. The punctuation rule is given as follows:

$$\tilde{\boldsymbol{y}}_i = \boldsymbol{y}_i[(t_{i,a_i}+1):(t_{i,b_i}+L)], \tag{14}$$

where

$$a_i \triangleq \max(1, k-i+1), \quad b_i \triangleq \min(k, n-i+1). \tag{15}$$

The two-step encoding approach of C has the same computation cost as the shift-XOR code C'.

As the ranges of the punctured coded sequences are predetermined as in (14) and (15), it is not necessary to perform XOR operations on the bits that are discarded in the second step. So we have the following one-step encoding approach to calculate the punctured coded sequences directly.

2) One-step Encoding: Substituting (1) into (14), we obtain for $\ell = 1,\ldots,t_{i,b_i} - t_{i,a_i} + L$,

$$\tilde{\boldsymbol{y}}_i[\ell] = \sum_{j=1}^{k} \boldsymbol{x}_j[\ell + t_{i,a_i} - t_{i,j}], \tag{16}$$

where we use the convention that $\boldsymbol{x}_j[l] = 0$ when $l < 0$ or $l > L$. The one-step encoding approach just implements the formula in (16). A pseudocode of the encoding algorithm is given in Algorithm 1. Using this algorithm, the encoding complexity of a punctured shift-XOR code can also be reduced compared with the encoding of the corresponding shift-XOR code, as less bits are calculated.

Algorithm 1 Encoding of shift-XOR codes

Input: message sequences $\boldsymbol{x}_j$, where $1 \le j \le k$.
Output: encoded sequences $\tilde{\boldsymbol{y}}_i$ $(1 \le i \le n)$.

```
for i ← 1, 2, ..., n do
    ỹ_i ← x_{a_i}[1 : (t_{i,b_i} − t_{i,a_i} + L)] (Initialize ỹ_i)
    for j ← 1, 2, ..., a_i − 1, a_i + 1, ..., k do
        for ℓ ← 1, 2, ..., (t_{i,b_i} − t_{i,a_i} + L) do
            if t_{i,j} − t_{i,a_i} < ℓ ≤ t_{i,j} − t_{i,a_i} + L then
                ỹ_i[ℓ] ← x_j[ℓ + t_{i,a_i} − t_{i,j}] ⊕ ỹ_i[ℓ]
```

C. Decoding of Punctured shift-XOR Codes

The decoding process is formed by two stages. In the transmission stage, a subset of k nodes is fixed for retrieving the data for decoding. The indices $i_1, i_2, \ldots, i_k$ of the k nodes are sorted in the descending order, i.e., $i_1 > i_2 > \cdots > i_k$. For $1 \leq u \leq k$, the decoder retrieves a subsequence $\tilde{\boldsymbol{x}}_u$ of $\tilde{\boldsymbol{y}}_{i_u}$ of L bits:

$$\tilde{\boldsymbol{x}}_u = \tilde{\boldsymbol{y}}_{i_u}[(t_{i_u,u} - t_{i_u,a_{i_u}} + 1) : (t_{i_u,u} - t_{i_u,a_{i_u}} + L)]. \quad (17)$$

Since $i_1 > i_2 > \cdots > i_k$, i_u must be at most $n+1-u$. Hence $u \leq n+1-i_u$, which together with $u \leq k$, implies $u \leq b_{i_u}$. Due to the increasing property, $t_{i_u,u} \leq t_{i_u,b_{i_u}}$. Substituting (14) into (17), we obtain

$$\tilde{\boldsymbol{x}}_u = \boldsymbol{y}_{i_u}[(t_{i_u,u} + 1) : (t_{i_u,u} + L)] = \hat{\boldsymbol{x}}_u, \quad (18)$$

where the second equality follows from (4). Obviously, the total bits retrieved by the decoder is kL so that the decoding overhead is zero.

In the second stage, after retrieving the kL bits from the storage nodes, the shift-XOR elimination can be applied on $\tilde{\boldsymbol{x}}_u = \hat{\boldsymbol{x}}_u$, $u = 1, \ldots, k$ to decode the message sequences. The decoding process is justified in the following theorem.

Theorem 3. *Consider a distributed storage system using an (n, k) punctured shift-XOR code. The shift-XOR elimination can decode the message sequences using exactly kL bits retrieved from any k out of the n storage nodes.*

Proof. Consider that a storage system S_1 uses an (n, k) punctured shift-XOR code C_1 and a storage system S_2 uses an (n, k) shift-XOR code which has the same generator matrix as C_1 to encode the same message sequences $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$. Suppose k nodes of the same indices $i_1 > i_2 > \cdots > i_k$ are chosen from S_1 and S_2 respectively. By (18), we have $\tilde{\boldsymbol{x}}_u = \hat{\boldsymbol{x}}_u$ for all u, i.e., the inputs to the shift-XOR elimination of both systems are the same. Therefore, S_1 has the same decoding result as S_2, which is corrected according to Theorem 1 in [16]. □

D. Storage Overhead Analysis

We study the storage overheads of the punctured shift-XOR codes, and compare with the corresponding shift-XOR codes of the same generator matrix. For the (n, k) punctured shift-XOR code with generator matrix $\boldsymbol{\Psi} = (z^{t_{i,j}})$, node i only stores the subsequence $\boldsymbol{y}_i[(t_{i,a_i}+1) : (t_{i,b_i}+L)]$, and hence the storage overhead is $t_{i,b_i} - t_{i,a_i} \leq t_{i,k}$ bits. Denote by $N_p(\boldsymbol{\Psi})$ the storage overhead of the (n, k) punctured shift-XOR code with generator matrix $\boldsymbol{\Psi}$. We know that

$$N_p(\boldsymbol{\Psi}) = \sum_{i=1}^{n} (t_{i,b_i} - t_{i,a_i}). \quad (19)$$

Theorem 4. *The storage overhead of an (n, k) punctured shift-XOR code with generator matrix $\boldsymbol{\Psi} = (z^{t_{i,j}})$ is lower bounded by $\frac{1}{2}(k-1)(n-1)(n-k)$, and the lower bound is achieved if the generator matrix is the Vandermonde matrix $\boldsymbol{\Psi} = (z^{(i-1)(j-1)})$.*

Proof. By (19),

$$N_p(\boldsymbol{\Psi}) = \sum_{i=1}^{n} (t_{i,b_i} - t_{i,a_i}) \geq \sum_{i=1}^{n} (i-1)(b_i - a_i), \quad (20)$$

Recall the definition (15), we have

$$\begin{aligned}
N_p(\boldsymbol{\Psi}) =& \sum_{i=1}^{n} (i-1)(b_i - a_i) \\
=& \sum_{i=1}^{n} (i-1)(k-1) + \sum_{i=1}^{k} (i-1)(k-i) \\
& - \sum_{i=n-k+1}^{n} (i-1)(i-n-k+1) \\
=& \frac{(k-1)(n-1)n}{2} + \sum_{i=1}^{k} (i-1)(k-i) \\
& - \sum_{i=1}^{k} (n-i)(k-i) \\
=& \frac{(k-1)(n-1)n}{2} - (n-1)\sum_{i=1}^{k} (i-1) \\
=& \frac{(k-1)(n-1)(n-k)}{2}
\end{aligned}$$

□

Remark 1. The generator matrix being the Vandermonde matrix $(z^{(i-1)(j-1)})$ is not necessary for achieving the lower bound in the above theorem. Consider the $(3, 2)$ punctured shift-XOR code with the generator matrix

$$\boldsymbol{\Psi} = \begin{pmatrix} 1 & 1 \\ 1 & z \\ 1 & z^3 \end{pmatrix}.$$

Upon checking, we have

$$\begin{aligned}
&a_1 = 2,\ a_2 = 1,\ a_3 = 1, \\
&b_1 = 2,\ b_2 = 1,\ b_3 = 1,
\end{aligned}$$

and hence

$$\begin{aligned}
t_{1,b_1} - t_{1,a_1} &= 0 = (1-1)(2-2), \\
t_{2,b_2} - t_{2,a_2} &= 1 = (2-1)(2-1), \\
t_{3,b_3} - t_{3,a_3} &= 0 = (3-1)(1-1).
\end{aligned}$$

Therefore the equality in (20) holds, which implies the lower bound is achieved.

Corollary 5. *For an (n, k) punctured shift-XOR code and an (n, k) shift-XOR code with the same Vandermonde generator matrix $\boldsymbol{\Psi}_{n\times k} = (z^{(i-1)(j-1)})$,*

$$\frac{N_p(\boldsymbol{\Psi}_{n\times k})}{N_s(\boldsymbol{\Psi}_{n\times k})} = 1 - \frac{k}{n}.$$

Example 3. We consider the $(n, 4)$ punctured shift-XOR codes with the Vandermonde generator matrix $\boldsymbol{\Psi} = (z^{(i-1)(j-1)})$. The following table gives the values of a_i and b_i for different values of n. Note that for $n < 7$, all the coded sequences are

TABLE III
VALUES OF a_i, b_i FOR DIFFERENT n.

	$n=4$		$n=5$		$n=6$		$n=7$		$n=8$	
i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i
1	4	4	4	4	4	4	4	4	4	4
2	3	3	3	4	3	4	3	4	3	4
3	2	2	2	3	2	4	2	4	2	4
4	1	1	1	2	1	3	1	4	1	4
5			1	1	1	2	1	3	1	4
6					1	1	1	2	1	3
7							1	1	1	2
8									1	1
N_p	0		6		15		27		42	
N_s	18		30		45		63		84	
$N_s - N_p$	18		24		30		36		42	

punctured. When $n > 7$, for $i \in \{k, \ldots, n-k+1\}$, $\boldsymbol{y}_i$ is not punctured.

Theorem 6. *For an (n,k) punctured shift-XOR code and an (n,k) shift-XOR code with the same generator matrix $\boldsymbol{\Psi}$ (which satisfies the RID property),*

$$N_s(\boldsymbol{\Psi}) - N_p(\boldsymbol{\Psi}) \geq \frac{1}{2}(n-1)k(k-1),$$

where the lower bound is achieved by $\boldsymbol{\Psi} = (z^{(i-1)(j-1)})$.

Proof. Consider the two-step encoding of the punctured shift-XOR code. From the left, the number of bits to be punctured is

$$\begin{aligned} N_{\text{left}}(\boldsymbol{\Psi}) &\triangleq \sum_{i=1}^{n} t_{i,a_i} \\ &\geq \sum_{i=1}^{n} (i-1)\max(0, k-i) \\ &= \sum_{i=1}^{k-1} (i-1)(k-i) \end{aligned}$$

where the inequality follows from Lemma 1. Similarly, the number of bits punctured from the right is

$$\begin{aligned} N_{\text{right}}(\boldsymbol{\Psi}) &\triangleq \sum_{i=1}^{n} t_{i,k} - t_{i,b_i} \\ &\geq \sum_{i=1}^{n} (i-1)(k-1-\min(k-1, n-i)) \\ &= \sum_{i=1}^{k-1} (n-i)(k-i). \end{aligned}$$

Hence

$$\begin{aligned} N_s(\boldsymbol{\Psi}) - N_p(\boldsymbol{\Psi}) &= N_{\text{left}} + N_{\text{right}} \\ &\geq \sum_{i=1}^{k-1} (k-i)[(n-i) + (i-1)] \\ &= \frac{1}{2}(n-1)k(k-1). \end{aligned}$$

Combining Theorem 2 and Theorem 4, the lower bound can be achieved when $\boldsymbol{\Psi} = (z^{(i-1)(j-1)})$. □

E. Encoding Complexity Analysis

Consider the encoding of an (n,k) punctured shift-XOR code with the generator matrix $\boldsymbol{\Psi} = (z^{t_{i,j}})$. If we use the two-step encoding approach, the number of XOR operations is the same as that of the shift-XOR code with the generator matrix $\boldsymbol{\Psi}$, where the latter is upper bounded by $n(k-1)L$ XOR operations. Using Algorithm 1 to encode the punctured shift-XOR code costs a lower number XOR operations as some bits are not calculated.

Theorem 7. *For an (n,k) punctured shift-XOR code with generator matrix $\boldsymbol{\Psi} = (z^{t_{i,j}})$ and $L > t_{n,k}$, the total number of XOR operations N^{enc} required in Algorithm 1 satisfies*

$$N^{enc} \leq n(k-1)L - \frac{1}{6}(n-1)(k-1)k(k+1). \tag{21}$$

Proof. Assume $L > t_{n,k}$ henceforth. We directly calculate the XOR operations in Algorithm 1. For fixed integer i with $1 \leq i \leq n$ and integer j with $1 \leq j \leq k$, $j \neq a_i$, the number of XOR operations for calculating $\tilde{\boldsymbol{y}}_i$ from Line 4 to 6 of Algorithm 1 is given by the number of ℓ satisfying the following system of inequalities:

$$\begin{cases} 1 \leq \ell \leq t_{i,b_i} - t_{i,a_i} + L, \\ t_{i,j} - t_{i,a_i} + 1 \leq \ell \leq t_{i,j} - t_{i,a_i} + L. \end{cases}$$

The above system can be simplified for three cases: when $1 \leq j \leq a_i - 1$,

$$1 \leq \ell \leq t_{i,j} - t_{i,a_i} + L; \tag{22}$$

when $a_i + 1 \leq j \leq b_i$,

$$t_{i,j} - t_{i,a_i} + 1 \leq \ell \leq t_{i,j} - t_{i,a_i} + L; \tag{23}$$

when $b_i + 1 \leq j \leq k$,

$$t_{i,j} - t_{i,a_i} + 1 \leq \ell \leq t_{i,b_i} - t_{i,a_i} + L. \tag{24}$$

Note that by $L > t_{n,k}$, $t_{i,j} - t_{i,a_i} + L \geq t_{i,j} + 1 \geq 1$ and $t_{i,b_i} - t_{i,a_i} + L \geq t_{i,b_i} - t_{i,a_i} + t_{i,j} + 1 \geq t_{i,j} - t_{i,a_i} + 1$. Hence by (22), (23) and (24), the total number of XOR operations N_i^{enc} for encoding $\tilde{\boldsymbol{y}}_i$ is

$$\begin{aligned} N_i^{\text{enc}} &= \sum_{j=1}^{a_i-1} (t_{i,j} - t_{i,a_i} + L) + \sum_{j=a_i+1}^{b_i} L + \sum_{j=b_i+1}^{k} (t_{i,b_i} - t_{i,j} + L) \\ &= (k-1)L - \sum_{j=1}^{a_i-1} (t_{i,a_i} - t_{i,j}) - \sum_{j=b_i+1}^{k} (t_{i,j} - t_{i,b_i}). \end{aligned}$$

By Lemma 1, we have

$$\sum_{j=1}^{a_i-1} (t_{i,a_i} - t_{i,j}) \geq \sum_{j=1}^{a_i-1} (i-1)(a_i - j) \tag{25}$$

and

$$\sum_{j=b_i+1}^{k} (t_{i,j} - t_{i,b_i}) \geq \sum_{j=b_i+1}^{k} (i-1)(j-b_i). \tag{26}$$

Now consider

$$\begin{aligned} N_l^- &\triangleq \sum_{i=1}^{n}\sum_{j=1}^{a_i-1}(t_{i,a_i}-t_{i,j}) \\ &\geq \sum_{i=1}^{n}\sum_{j=1}^{a_i-1}(i-1)(a_i-j) \qquad (27) \\ &= \sum_{i=1}^{k-1}\sum_{j=1}^{a_i-1}(i-1)(a_i-j) \qquad (28) \\ &= \frac{1}{2}\sum_{i=1}^{k-1}(i-1)(k-i+1)(k-i), \qquad (29) \end{aligned}$$

where (27) follows from (25), and (28) is because $a_i = 1, \forall\, i > k-1$. Consider

$$\begin{aligned} N_r^- &\triangleq \sum_{i=1}^{n}\sum_{j=b_i+1}^{k}(t_{i,j}-t_{i,b_i}) \\ &\geq \sum_{i=1}^{n}\sum_{j=b_i+1}^{k}(i-1)(j-b_i) \qquad (30) \\ &= \sum_{i=n-k+2}^{n}\sum_{j=b_i+1}^{k}(i-1)(j-b_i) \qquad (31) \\ &= \sum_{i'=1}^{k-1}\sum_{j=i'+1}^{k}(n-i')(j-i') \qquad (32) \\ &= \frac{1}{2}\sum_{i=1}^{k-1}(n-i)(k-i+1)(k-i), \qquad (33) \end{aligned}$$

where (30) follows from (26), (31) is because $b_i = k$, $\forall\, i < n-k+2$, and (32) is based on the transformation $i' = n+1-i$ and the observation that $n+1-i = b_i$, $\forall\, i < n-k+2$. Hence

$$\begin{aligned} N^{\text{enc}} &= \sum_{i=1}^{n} N_i^{\text{enc}} \\ &= n(k-1)L - N_l^- - N_r^- \\ &\leq n(k-1)L - \frac{1}{2}\sum_{i=1}^{k-1}(n-1)(k-j+1)(k-j) \qquad (34) \\ &= n(k-1)L - \frac{1}{6}(n-1)(k-1)k(k+1). \end{aligned}$$

When the generator matrix $\mathbf{\Psi} = (z^{(i-1)(j-1)})$, (25) and (26) become equalities, and hence (27) and (30) become equalities. Therefore, the equality in (34) holds when $\mathbf{\Psi} = (z^{(i-1)(j-1)})$. □

IV. Concluding Remarks

We refined the shift-XOR storage codes satisfying the RID property by removing the unnecessary bits from the storage nodes, while preserving the same decoding algorithm. Our approach can reduce both the storage overhead and the encoding complexity.

As a future work, we may consider whether the punctuation technique can be applied in shift-XOR regenerating codes to reduce the overhead. We are also interested to study shift-XOR codes that do not satisfy the RID property.

References

[1] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, pp. 300–304, Jun. 1960.

[2] 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, feb 1995.

[3] J. Bloemer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-based erasure-resilient coding scheme," *IGSI Technical Report No. TR-95-048*, Aug. 1995.

[4] G.-L. Feng, R. H. Deng, F. Bao, and J.-C. Shen, "New efficient MDS array codes for RAID. part ii. Rabin-like codes for tolerating multiple (≥ 4) disk failures," *IEEE Transactions on Computers*, vol. 54, no. 12, pp. 1473–1483, 2005.

[5] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *USENIX Conf. File and Storage Technologies*, Mar. 2004, pp. 1–14.

[6] M. Blaum, J. Brady, J. Bruck, J. Menon, and A. Vardy, "The EVENODD code and its generalization," *High Performance Mass Storage and Parallel I/O*, pp. 187–208, 2001.

[7] J. S. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications," in *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*. IEEE, 2006, pp. 173–180.

[8] 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.

[9] C. W. Sung and X. Gong, "A ZigZag-decodable code with the MDS property for distributed storage systems," in *IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 341–345.

[10] X. Gong and C. W. Sung, "Zigzag decodable codes: Linear-time erasure codes with applications to data storage," *J. Comput. Syst. Sci.*, vol. 89, pp. 190–208, 2017.

[11] M. Dai, C. W. Sung, H. Wang, X. Gong, and Z. Lu, "A new zigzag-decodable code with efficient repair in wireless distributed storage," *IEEE Trans. Mob. Comput.*, vol. 16, no. 5, pp. 1218–1230, 2017.

[12] C. W. Sung and X. Gong, "Combination network coding: Alphabet size and zigzag decoding," in *IEEE Int. Symp. on Information Theory and Applications*. IEEE, 2014, pp. 699–703.

[13] T. Nozaki, "Fountain codes based on zigzag decodable coding," in *IEEE Int. Symp. on Information Theory and Applications*. IEEE, 2014, pp. 274–278.

[14] M. Dai, C. W. Sung, H. Wang, X. Gong, and Z. Lu, "A new Zigzag-decodable code with efficient repair in wireless distributed storage," *IEEE Trans. Mob. Comput.*, vol. 16, no. 5, pp. 1218–1230, 2017.

[15] X. Fu, S. Yang, and Z. Xiao, "Recovery and repair schemes for shift-XOR regenerating codes," *CoRR*, vol. abs/1907.05058, 2019. [Online]. Available: http://arxiv.org/abs/1907.05058

[16] X. Fu, Z. Xiao, and S. Yang, "Overhead-free in-place recovery scheme for XOR-based storage codes," in *13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2014, Beijing, China, September 24-26, 2014*, 2014, pp. 552–557.