Two-tone Shift-XOR Storage Codes

Ximing Fu, Chenhao Wu, Yuanxin Guo and Shenghao Yang†

***Abstract*—Storage codes using shift and XOR operations have been studied to achieve lower encoding and decoding computation costs, compared with the codes using large finite field operations. In this paper, we introduce a new class of shift-XOR codes using two-tone generator matrices, which generalize the existing increasing-difference generator matrices. Compared with the latter, our codes only have $1/3$ to $1/2$ storage overhead for practical cases, and have a decoding algorithm that preserves the desired properties. For two-tone shift-XOR codes, the reflected Vandermonde matrices achieve the smallest storage overhead; and for increasing-difference shift-XOR codes, the Vandermonde matrices achieve the smallest storage overhead. To verify the practical performance, we implement two-tone shift-XOR storage codes using C++ and compare the encoding/decoding throughput with the state-of-the-art implementation of Reed-Solomon codes. For certain practical cases, our codes can achieve from 50% to 100% higher encoding/decoding throughput than that of Reed-Solomon codes.**

I. Introduction

Finite field operations have been extensively studied for distributed storage codes. For example, Reed-Solomon (RS) codes [1] and Cauchy RS codes [2] are widely used MDS (maximum-distance separable) codes in distributed storage systems [3]–[5]. Though optimal in terms of erasure correction capability, researchers have studied various alternative storage codes that entail lower computational costs. For example, cyclic shift over finite fields and XOR operations were employed in array codes [6]. An $[n, k]$ erasure code enables correct decoding from any k out of n coded sequences. For any $k \leq n$, $[n, k]$ erasure codes using cyclic shift and XOR operations have been constructed using Vandermonde-type generator matrices [7], [8]. Another coding scheme using cyclic shift and integer addition was proposed in [9].

In this paper, we focus on codes employing (noncyclic) shift and XOR operations [10]–[17], which potentially have the lowest encoding and decoding computational costs among the existing $[n, k]$ erasure coding techniques. In an $[n, k]$ shift-XOR code, a data file formed by k message sequences each of L bits is encoded into n sequences, where encoding one coded sequences requires at most $(k-1)L$ XOR operations.

The low computation cost of shift-XOR codes as the price of the extra storage cost. Due to the shift operation, each coded sequence may be longer than L bits, so that the shift-XOR codes are not strictly MDS codes. As nL is the minimum number of coded bits to be generated by an $[n, k]$ MDS code, the total number of coded bits generated by an $[n, k]$ shift-XOR code minus nL is called the *storage overhead*. For any $k \leq n$, $[n, k]$ shift-XOR codes with the *refined increasing difference (RID)* generator matrices have been studied [10]–[12], [17], where the storage overhead of the systematic version is at least $\frac{(n-k)(n-k-1)(k-1)}{2}$ bits (to be proved in this paper). For $n \leq 2k$, systematic $[n, k]$ shift-XOR codes can be constructed using a circulant generator matrix [15] with storage overhead of $\frac{(n-k)k(k-1)}{2}$ bits for $k \geq 4$ and $(n-k)$ bits for $k = 2, 3$.

In this paper, we propose a class of generator matrices for shift-XOR codes, called *two-tone matrices*, that includes RID generator matrices as special cases and achieves lower storage overheads. In a RID generator matrix [12], the numbers of bit shift specified in each row are increasing. In a two-tone generator matrix, the numbers of bit shift specified in each row can be both increasing and decreasing. For decoding, we generalize the shift-XOR elimination to handle both the increasing and decreasing order of the numbers of bit shift of the message sequences. Our algorithm preserves the advantages of the shift-XOR elimination, including in-place and bandwidth-overhead free.

We further prove that for two-tone shift-XOR codes, the *reflected Vandermonde matrices* achieve the smallest storage overhead; and for RID shift-XOR codes, the Vandermonde matrices achieve the smallest storage overhead. Compared with the Vandermonde RID generator matrices, the corresponding two-tone generator matrices can reduce up to half of the storage overheads. Moreover, based on two-tone matrices, we propose systematic shift-XOR storage codes that can further reduce the storage overheads to less than 10% of that of the non-systematic codes for some practical parameters. The minimum storage overhead of systematic $[n, k]$ two-tone codes is $\frac{((n-k)^2-1)(k-1)}{4}$ when $n-k$ is odd and $\frac{(n-k)^2(k-1)}{4}$ when $n-k$ is even.

Compared with a storage system using RS codes, one using two-tone codes may gain higher encoding/decoding throughput with the price of a slightly higher storage cost. We implement two-tone codes using C++ and compare the encoding/decoding throughput with the state-of-the-art implementations of RS codes and Cauchy RS codes. Our codes achieve from 50 to 100 percent higher encoding/decoding throughput than ISA-L [18], which is considered the state-of-the-art RS encoding/decoding library. Compared with Cauchy-RS codes implemented in the Longhair library [19], our codes can achieve 80 percent higher

The authors are with The Chinese University of Hong Kong, Shenzhen. X. Fu is also with University of Science and Technology of China. S. Yang is also with Shenzhen Key Laboratory of IoT Intelligent Systems and Wireless Network Technology, and Shenzhen Research Institute of Big Data, Shenzhen, China. This work was funded in part by Shenzhen Science and Technology Innovation Committee (Grant ZDSYS20170725140921348, JCYJ20180508162604311).

† Corresponding author. Email: shyang@cuhk.edu.cn

encoding/decoding throughputs for small file size (128KB) and from 5 to 8 times the encoding/decoding throughputs for large file size (512MB). Compared with the RS codes implemented in Jerasure library [20], our codes can achieve 10 times the encoding/decoding throughputs.

II. Shift-XOR Codes with Two-tone Generator Matrices

We denote a range of integers from i to j by $i:j$. When $i>j$, $i:j$ is the empty set. For a binary sequence $\mathbf{a}$, denoted by bold lowercase letters, the i-th entry is denoted by $\mathbf{a}[i]$. The subsequence of $\mathbf{a}$ from the i-th entry to the j-th entry is denoted by $\mathbf{a}[i:j]$. For a sequence $\mathbf{a}$ of length L, we use the convention that $\mathbf{a}[l]=0$ for $l<1$ or $l>L$.

For a sequence $\mathbf{a}$ of L bits and a natural number $t\geq 0$, the shift operator z^t pads t zeros in front of $\mathbf{a}$, so that the $(l+t)$-th entry of $z^t\mathbf{a}$ is equal to the l-th of $\mathbf{a}$, i.e., for $l=1,\ldots,L+t$,

$$\left(z^t\mathbf{a}\right)[l]=\mathbf{a}\left[l-t\right].$$

Let $\mathbf{a}_1$ and $\mathbf{a}_2$ be two sequences of length L_1 and L_2, respectively. Their addition $\mathbf{a}_1+\mathbf{a}_2$ is bit-wise exclusive-or (XOR). If these two sequences are not of the same length, zeros are appended after the shorter one before the addition so that for $l=1,\ldots,\max\{L_1,L_2\}$, $(\mathbf{a}_1+\mathbf{a}_2)[l]=\mathbf{a}_1[l]\oplus\mathbf{a}_2[l]$.

A. Shift-XOR Codes

Consider k binary message sequences, each of L bits, where the j-th sequence is denoted as $\mathbf{x}_j$. The *generator matrix* used to encode the message sequences is an $n\times k$ matrix $\mathbf{\Psi}=(z^{t_{i,j}})$, where $t_{i,j}\geq 0$ determines the number of bit shifts of the j-th message sequence in the i-th *coded sequence* $\mathbf{y}_i$. We use the convention that $z^\infty=0$, and $t_{i,j}=\infty$ means that $\mathbf{x}_j$ is not involved in the encoding of $\mathbf{y}_i$. So,

$$\mathbf{y}_i=\sum_{j=1}^{k} z^{t_{i,j}}\mathbf{x}_j,\ \forall\, 1\leq i\leq n.$$

Denote by t_i^* the maximum finite value among $t_{i,1},\ldots,t_{i,k}$. The length of $\mathbf{y}_i$ is $L+t_i^*$. Denote the vectors $\mathbf{Y}=(\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_n)^\top$ and $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_k)^\top$. The encoding can be written as

$$\mathbf{Y}=\mathbf{\Psi}\mathbf{X}, \tag{1}$$

which is also called an $n\times k$ shift-XOR system.

Existing works have studied the generator matrices satisfying the refined increasing difference (RID) properties [10], [12], [17], for which an efficient decoding algorithm exists. Here we introduce a more general class of generator matrices, called *two-tone* matrices.

Definition 1 (Two-tone Matrix)**.** An $n\times k$ matrix $\mathbf{\Psi}=(z^{t_{i,j}})$ is said to be *two-tone* if for certain integer $0\leq d\leq n$ the following conditions hold:

1) for any $1\leq i_1<i_2\leq n$, $1\leq j_1<j_2\leq k$, $t_{i_1,j_2}-t_{i_1,j_1}<t_{i_2,j_2}-t_{i_2,j_1}$;
2) for any $1\leq i\leq d$, $1\leq j_1<j_2\leq k$, $t_{i,j_2}-t_{i,j_1}\leq 0$;
3) for any $d<i\leq n$, $1\leq j_1<j_2\leq k$, $t_{i,j_2}-t_{i,j_1}>0$.

Here d is called the *divide* of $\mathbf{\Psi}$.

By condition 2) of the definition, if $t_{i,j_2}-t_{i,j_1}=0$ for certain $j_1<j_2$, then $i=d$. In other words, only the divide row can have zero differences. It is easy to verify that any submatrix of a two-tone matrix is still a two-tone matrix. The RID matrices are special cases of two-tone matrices with divide $d=0,1$.

B. Examples of Decoding Algorithms

We first use examples to illustrate how to solve two-tone shift-XOR systems. As our algorithm is based on the shift-XOR elimination, which was proposed for RID systems [17], we first use an example to explain how does it work.

Example 1. Consider the 2×2 shift-XOR system

$$\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}=\begin{bmatrix}1 & z\\ 1 & z^2\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}. \tag{2}$$

As the generator matrix is RID, the shift-XOR elimination can be applied to solve the system using subsequences of $\mathbf{y}_1,\mathbf{y}_2$:

$$\hat{\mathbf{x}}_1=\mathbf{y}_2[1:L],$$
$$\hat{\mathbf{x}}_2=\mathbf{y}_1[2:L+1].$$

Expanding the shift operator, we get for $1\leq l\leq L$,

$$\hat{\mathbf{x}}_1[l]=\mathbf{x}_1[l]+\mathbf{x}_2[l-2],$$
$$\hat{\mathbf{x}}_2[l]=\mathbf{x}_2[l]+\mathbf{x}_1[l+1].$$

The system is solved by multiple iterations. An iteration index ℓ is initialized as 1, and is increased by 1 after each iteration. The following processes are done in each iteration: When $\ell=1$, $\mathbf{x}_1[1]=\hat{\mathbf{x}}_1[1]$ is solved. For each iteration $\ell=2,3,\ldots,L+1$, $\mathbf{x}_1[\ell]$ and $\mathbf{x}_2[\ell-1]$ are solved sequentially by equations

$$\mathbf{x}_1[\ell]=\hat{\mathbf{x}}_1[\ell]+\mathbf{x}_2[\ell-2],$$
$$\mathbf{x}_2[\ell-1]=\hat{\mathbf{x}}_2[\ell-1]+\mathbf{x}_1[\ell],$$

respectively, where we can check inductively that all the message bits used on the RHS have been solved previously.

Example 2. The 3×3 system

$$\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\\ \mathbf{y}_3\end{bmatrix}=\begin{bmatrix}z^4 & z^2 & 1\\ z^2 & z & 1\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\end{bmatrix} \tag{3}$$

does not have a RID generator matrix, but it is equivalent to one with a RID generator matrix:

$$\begin{bmatrix}\mathbf{y}_3\\ \mathbf{y}_2\\ \mathbf{y}_1\end{bmatrix}=\begin{bmatrix}1 & 1 & 1\\ 1 & z & z^2\\ 1 & z^2 & z^4\end{bmatrix}\begin{bmatrix}\mathbf{x}_3\\ \mathbf{x}_2\\ \mathbf{x}_1\end{bmatrix}.$$

Therefore, system (3) can also be solved by the shift-XOR elimination.

In general, for a $k\times k$ two-tone matrix with divide k, by reversing the order of rows and columns, we obtain a two-tone matrix with divide 1, i.e., an RID matrix.

Example 3. Next, we consider a more general example:

$$\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\\ \mathbf{y}_3\\ \mathbf{y}_4\\ \mathbf{y}_5\end{bmatrix} = \begin{bmatrix} z^8 & z^6 & z^4 & z^2 & 1\\ z^4 & z^3 & z^2 & z & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & z & z^2 & z^3 & z^4\\ 1 & z^2 & z^4 & z^6 & z^8\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\\ \mathbf{x}_4\\ \mathbf{x}_5\end{bmatrix},$$

where the generator matrix has divide 3. The system can be further written as two systems of the same set of variables:

$$\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\\ \mathbf{y}_3\end{bmatrix} = \begin{bmatrix} z^4 & z^2 & 1\\ z^2 & z & 1\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}_3\\ \mathbf{x}_4\\ \mathbf{x}_5\end{bmatrix} + \begin{bmatrix} z^8 & z^6\\ z^4 & z^3\\ 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}, \quad (4)$$

$$\begin{bmatrix}\mathbf{y}_4\\ \mathbf{y}_5\end{bmatrix} = \begin{bmatrix} 1 & z\\ 1 & z^2\end{bmatrix}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} + \begin{bmatrix} z^2 & z^3 & z^4\\ z^4 & z^6 & z^8\end{bmatrix}\begin{bmatrix}\mathbf{x}_3\\ \mathbf{x}_4\\ \mathbf{x}_5\end{bmatrix}. \quad (5)$$

We observe that without the second term on the RHS, both systems above can be solved by the shift-XOR elimination (ref. (2) and (3)). The second term in each system can be regarded as the interference from other one. Our idea is to solve (4) and (5) using two individual shift-XOR eliminations, together with interference cancellation between each other.

Same as the shift-XOR elimination, we define subsequences

$$\begin{aligned}\hat{\mathbf{x}}_1 &= \mathbf{y}_5[1:L], & \hat{\mathbf{x}}_3 &= \mathbf{y}_3[1:L],\\ \hat{\mathbf{x}}_2 &= \mathbf{y}_4[2:L+1], & \hat{\mathbf{x}}_4 &= \mathbf{y}_2[2:L+1],\\ & & \hat{\mathbf{x}}_5 &= \mathbf{y}_1[1:L],\end{aligned}$$

where $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ are used by solving (5), and $\hat{\mathbf{x}}_3, \hat{\mathbf{x}}_4, \hat{\mathbf{x}}_5$ are used by solving (4). Expanding the shift operator, we further have for $1 \le l \le L$,

$$\begin{aligned}\hat{\mathbf{x}}_1[l] &= \mathbf{x}_1[l] + \mathbf{x}_2[l-2] + \mathbf{x}_3[l-4] + \mathbf{x}_4[l-6] + \mathbf{x}_5[l-8],\\ \hat{\mathbf{x}}_2[l] &= \mathbf{x}_2[l] + \mathbf{x}_1[l+1] + \mathbf{x}_3[l-1] + \mathbf{x}_4[l-2] + \mathbf{x}_5[l-3],\\ \hat{\mathbf{x}}_3[l] &= \mathbf{x}_3[l] + \mathbf{x}_4[l] + \mathbf{x}_5[l] + \mathbf{x}_1[l] + \mathbf{x}_2[l],\\ \hat{\mathbf{x}}_4[l] &= \mathbf{x}_4[l] + \mathbf{x}_3[l-1] + \mathbf{x}_5[l+1] + \mathbf{x}_1[l-3] + \mathbf{x}_2[l-2],\\ \hat{\mathbf{x}}_5[l] &= \mathbf{x}_5[l] + \mathbf{x}_3[l-4] + \mathbf{x}_4[l-2] + \mathbf{x}_1[l-8] + \mathbf{x}_2[l-6].\end{aligned}$$

We use ℓ^+ and ℓ^- as the iteration indices of system (5) and (4), respectively. In the first iteration of both systems, we see that some bits can be solved directly:

- For (5), we solve $\mathbf{x}_1[1] = \hat{\mathbf{x}}_1[1]$.
- For (4), we solve $\mathbf{x}_5[1] = \hat{\mathbf{x}}_5[1]$.

Now, $\ell^+ = \ell^- = 2$. The following operations are performed for each following iteration assuming $\mathbf{x}_1[1:\ell^+-1]$, $\mathbf{x}_2[1:\ell^+-2]$, $\mathbf{x}_3[1:\ell^--2]$, $\mathbf{x}_4[1:\ell^--2]$ and $\mathbf{x}_5[1:\ell^--1]$ are already solved:

- (System (5)) Solve $\mathbf{x}_1[\ell^+]$ and $\mathbf{x}_2[\ell^+-1]$ sequentially using $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$, respectively, where the interference from $\mathbf{x}_3[\ell^+-4]$, $\mathbf{x}_3[\ell^+-2]$, $\mathbf{x}_4[\ell^+-6]$, $\mathbf{x}_4[\ell^+-3]$, $\mathbf{x}_5[\ell^+-8]$ and $\mathbf{x}_5[\ell^+-4]$ can be cancelled as these bits are solved previously.
- (System (4)) Solve $\mathbf{x}_5[\ell^-]$, $\mathbf{x}_4[\ell^--1]$, $\mathbf{x}_3[\ell^--1]$ sequentially using $\hat{\mathbf{x}}_5, \hat{\mathbf{x}}_4, \hat{\mathbf{x}}_3$, respectively, where the interference from $\mathbf{x}_1[l^--8]$, $\mathbf{x}_1[l^--8]$, $\mathbf{x}_1[l^--1]$, $\mathbf{x}_2[l^--6]$, $\mathbf{x}_2[l^--3]$ and $\mathbf{x}_2[l^--1]$ can be cancelled as they are solved previously.
- Increase the iteration indices ℓ^+ and ℓ^- by 1.

C. Two-tone Elimination

Now we consider the $k \times k$ system of shift-XOR equations

$$\begin{bmatrix}\mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_k\end{bmatrix}^\top = \mathbf{\Psi}\begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_k\end{bmatrix}^\top, \quad (6)$$

where $\mathbf{\Psi} = (z^{t_{i,j}})$ is a two-tone matrix with divide d. We give an algorithm to solve the above general system when $\mathbf{y}_1, \ldots, \mathbf{y}_k$ are given. Our algorithm generalizes the shift-XOR elimination in [17], and is called *two-tone elimination*.

The system (6) can be written as two sub-systems:

$$\begin{bmatrix}\mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_d\end{bmatrix}^\top = \mathbf{\Psi}^-\begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_k\end{bmatrix}^\top, \quad (6^-)$$

$$\begin{bmatrix}\mathbf{y}_{d+1} & \mathbf{y}_{d+2} & \cdots & \mathbf{y}_k\end{bmatrix}^\top = \mathbf{\Psi}^+\begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_k\end{bmatrix}^\top. \quad (6^+)$$

Define for $u = 1, 2, \ldots, k$ the subsequence

$$\hat{\mathbf{x}}_u = \mathbf{y}_{k+1-u}[(t_{k+1-u,u}+1) : (t_{k+1-u,u}+L)].$$

Substituting into (6) and expanding the shift operator, we have

$$\hat{\mathbf{x}}_u[l] = \mathbf{x}_u[l] + \sum_{j\neq u}\mathbf{x}_j[l - t_{k+1-u,j} + t_{k+1-u,u}],$$

where we see that $\hat{\mathbf{x}}_u$, $u = 1, \ldots, k$ involve all the bits we want to decode.

These two sub-systems are solved by two modified shift-XOR elimination interactively, where $\mathbf{x}_{k-d+1}, \ldots, \mathbf{x}_k$ are variables for (6^-), and $\mathbf{x}_1, \ldots, \mathbf{x}_{k-d}$ are variables for (6^+). The two sub-systems are solved by multiple iterations. We use ℓ^+ and ℓ^- as the iteration indices of system (6^+) and (6^-), respectively. Initially, $\ell^+ = \ell^- = 1$. After each iteration in each sub-system, the corresponding index is increased by 1. The operations of each iteration depend on the value of ℓ^+ and ℓ^-. We define the following parameters that can help us to separate iterations into segments:

$$\begin{aligned}T_i^+ &= t_{k-i,i+1} - t_{k-i,i},\ 1 \le i < k-d,\\ T_i^- &= t_{i+1,i} - t_{i+1,i+1},\ 1 \le i < d.\end{aligned}$$

Define $T_{i:j}^+ = \sum_{l=i}^j T_l^+$ and $T_{i:j}^- = \sum_{l=i}^j T_l^-$.

For a number of iterations at the beginning, both sub-systems can be solved separately with the following operations respectively for each iteration:

- For $b = 1, 2, \ldots, k-d-1$, for each iteration ℓ^+ in $T_{1:b-1}^+ + (1 : T_b^+)$, solve $\mathbf{x}_u[\ell^+ - T_{1:u-1}^+]$ sequentially for $u = 1, 2, \ldots, b$.
- For $b = 1, 2, \ldots, d-1$, for each iteration ℓ^- in $T_{1:b-1}^- + (1 : T_b^-)$, solve $\mathbf{x}_{k-u+1}[\ell^- - T_{1:u-1}^-]$ sequentially for $u = 1, 2, \ldots, b$.

In the above process, solving one bit implies that it is back substituted into the subsequences it involves in. After the above iterations, $\ell^+ = T_{1:k-d-1}^+ + 1$ and $\ell^- = T_{1:d-1}^- + 1$. Then the following operations are performed sequentially for each iteration:

- Solve $\mathbf{x}_u[\ell^+ - T^+_{1:u-1}]$ sequentially using $\hat{\mathbf{x}}_u$ and previously solved bits, for $u = 1, 2, \ldots, k-d$.
- Solve $\mathbf{x}_{k-u+1}[\ell^- - T^-_{1:u-1}]$ sequentially using $\hat{\mathbf{x}}_{k-u+1}$ and previously solved bits, $u = 1, 2, \ldots, d$.

The algorithm stops after all the bits are solved.

Same as the shift-XOR elimination, we can show that the two-tone elimination can be implemented in-place, i.e., no auxiliary space is required to store the intermediate shift-XOR results. Similar to the analysis of shift-XOR elimination, the two-tone elimination needs less than $k(k-1)$ XOR operations and $O(k^2L)$ integer operations. The correctness of the two-tone elimination can be guaranteed by the following Theorem.

Theorem 1. *Consider a $k \times k$ system of shift-XOR equations $(\mathbf{y}_1 \ \cdots \ \mathbf{y}_k)^\top = \mathbf{\Psi}(\mathbf{x}_1 \ \cdots \ \mathbf{x}_k)^\top$ with $\mathbf{\Psi}$ being a two-tone matrix. The two-tone elimination can successfully decode $\mathbf{x}_u$, $u = 1, \ldots, k$ using*

$$\hat{\mathbf{x}}_u = \mathbf{y}_{k+1-u}[(t_{k-u+1,u} + 1) : (t_{k-u+1,u} + L)].$$

Theorem 1 can be verified by expressing each bit to decode using $\hat{\mathbf{x}}_u$, $u = 1, 2, \ldots, k$ and the previously decoded bits.

III. Two-tone Storage Codes

Now we consider a storage system of n storage nodes employing an $[n, k]$ shift-XOR code as defined in (1). The n coded sequences are stored at n distinct storage nodes. Using two-tone matrices, both systematic and non-systematic codes can be constructed.

A. Non-systematic Storage Code

We first consider that the generator matrix Ψ is a two-tone generator matrix. We show that the file can be decoded from any k out of the n storage nodes.

Assume that the decoder has access to k nodes with the indices in descending order, i.e., $i_1 > i_2 > \cdots > i_k$. As any submatrix of a two-tone generator matrix is also a two-tone matrix, the k coded sequences that can be accessed by the decoder form a $k \times k$ two-tone shift-XOR system, which can be solved using the two-tone elimination.

Our decoding scheme consists of two stages: the transmission stage and the decoding stage. In the transmission stage, node i_u transmits $\mathbf{y}_{i_u}$ with the range of $[(t_{i_u,u} + 1) : (t_{i_u,u} + L)]$ to the decoder and stores in $\hat{\mathbf{x}}_u$, $u = 1, 2, \ldots, k$, i.e., $\hat{\mathbf{x}}_u = \mathbf{y}_{i_u}[(t_{i_u,u}+1) : (t_{i_u,u}+L)]$. As each node transmits exactly L bits to the decoder, the decoding scheme has no bandwidth overhead. In the decoding stage, the decoder applies the two-tone elimination on $\hat{\mathbf{x}}_u$, $u = 1, 2 \ldots, k$. Then $\hat{\mathbf{x}}_u$ can be decoded into $\mathbf{x}_u$ in-place.

B. Systematic Two-tone Code

An $[n, k]$ *systematic* two-tone code has the generator matrix

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{I} \\ \mathbf{\Phi} \end{bmatrix}, \tag{7}$$

where $\mathbf{I}$ is the $k \times k$ identity matrix and $\mathbf{\Phi}$ is an $(n-k) \times k$ two-tone matrix. Note that the first k coded sequences are identical to the message sequences, i.e., $\mathbf{y}_i = \mathbf{x}_i$ for $i = 1, 2, \ldots, k$, and are also called *systematic sequences*. The remaining $n-k$ coded sequences are called *parity sequences*.

Let us discuss the decoding algorithm of a storage system employing an $[n, k]$ systematic shift-XOR code. Assume that the decoder has access to k nodes with the indices in descending order $i_1 > i_2 > \cdots > i_k$. The k nodes have k_m systematic sequences and $k - k_m$ parity sequences. As the systematic sequences have smaller indices than the parity sequences, node i_u, $k - k_m + 1 \le u \le k$ stores the message sequence $\mathbf{x}_{i_u}$. Denote the indices of the remaining $k - k_m$ message sequence to decode as $1 \le h_1 < h_2 < \cdots < h_{k-k_m} \le k$.

The decoding scheme consists of two stages: the transmission stage and the decoding stage. In the transmission stage: first, the decoder retrieves $\mathbf{x}_{i_u}$ from node i_u for $k - k_m < u \le k$; second, the decoder retrieves $\hat{\mathbf{x}}_v = \mathbf{y}_{i_v}[t_{i_v,h_v} + (1 : L)]$ from node i_v for $1 \le v \le k - k_m$. In the decoding stage, $\mathbf{x}_{i_u}$, $k - k_m < u \le k$ are first substituted into $\hat{\mathbf{x}}_v$, $1 \le v \le k - k_m$. After the substitution, $\hat{\mathbf{x}}_v$, $1 \le v \le k - k_m$ form a two-tone system. Then the two-tone elimination is executed on $\hat{\mathbf{x}}_v$, $1 \le v \le k - k_m$ to decode $\mathbf{x}_{h_v}$ for $1 \le v \le k - k_m$.

For decoding the systematic code, the substitution costs no more than $(k - k_m)k_mL$ XOR operations and the two-tone elimination on $k - k_m$ sequences costs no more than $(k - k_m)(k - k_m - 1)L$ XOR operations. As a consequence, the number of XOR costs is at most $(k - k_m)k_mL + (k - k_m)(k - k_m - 1)L = (k - k_m)(k - 1)L$. Similar to the two-tone elimination, decoding the systematic code costs $O(k^2L)$ integer operations. As the substitution and two-tone elimination are in-place, decoding the systematic two-tone code is in-place implementable.

C. Storage Overhead Optimized Generator Matrices

Consider a storage system of n storage nodes employing shift-XOR codes described in the previous section. Due to the shift operation, each storage node may store more than L bits, the length of a message sequence. Each coded sequence $\mathbf{y}_i$ has $L + \max_{j=1}^{k} t_{i,j}$ bits, and hence the total number of bits stored at n codes is $nL + \sum_{i=1}^{n} \max_{j=1}^{k} t_{i,j}$. The *storage overhead* of the shift-XOR codes with generator matrix $\mathbf{\Psi} = (z^{t_{i,j}})$ is defined as

$$S(\mathbf{\Psi}) = \sum_{i=1}^{n} \max_{j=1}^{k} t_{i,j}. \tag{8}$$

In this section, we give specific constructions of two-tone matrices to minimize the storage overhead.

We define a special class of two-tone matrices that generalize the Vandermonde matrices.

Definition 2 (Two-tone and Reflected Vandermonde Matrices)**.** The $n \times k$ two-tone Vandermonde matrix $\mathbf{\Psi} = (z^{t_{i,j}})$ with divide d is defined as

$$t_{i,j} = \begin{cases} (d - i)(k - j), & 1 \le i \le d, \\ (i - d)(j - 1), & d < i \le n. \end{cases} \tag{9}$$

Further, $\mathbf{\Psi}$ is called a *reflected Vandermonde matrix* if

$$d = \begin{cases} \frac{n+1}{2}, & n \text{ is odd}, \\ \frac{n}{2} + 1 \text{ or } \frac{n}{2}, & n \text{ is even}. \end{cases}$$

TABLE I
STORAGE OVERHEAD COMPARISON OF DIFFERENT SHIFT-XOR CODES. IN THE TABLE, THE GENERATORS ARE THE ONES AS SPECIFIED IN THEOREM 2 AND 3, COROLLARY 1 AND 2.

Codes	$[8,6]$	$[11,8]$	$[14,10]$
RID code [12], [17]	140	385	819
two-tone code	80	210	441
systematic incre. diff. code [11]	15	42	90
systematic two-tone code	5	14	36

It is easy to check that the generator matrix defined by (9) is a two-tone matrix. When $d = 0, 1$, a two-tone Vandermonde matrix is also called a Vandermonde matrix. The reflected Vandermonde matrix can achieve minimal storage overhead among all two-tone ones of the same size, as shown in the next theorem.

Theorem 2. *The storage overhead of an $[n, k]$ shift-XOR code with two-tone generator matrix is lower bounded by $\frac{(n^2-1)(k-1)}{4}$ when n is odd, and $\frac{n^2(k-1)}{4}$ when n is even, and the lower bound is achieved if and only if the generator matrix is the reflected Vandermonde matrix.*

Similar to Theorem 2, we have the following result about the storage overhead of *systematic* two-tone shift-XOR codes.

Corollary 1. *The storage overhead of an $[n, k]$ shift-XOR code with* systematic *two-tone generator matrix $\mathbf{\Psi} = \begin{bmatrix} \mathbf{I} \\ \mathbf{\Phi} \end{bmatrix}$ is lower bounded by $\frac{((n-k)^2-1)(k-1)}{4}$ when $n-k$ is odd, and $\frac{(n-k)^2(k-1)}{4}$ when $n-k$ is even, and the lower bound is achieved if and only if $\mathbf{\Phi}$ is the reflected Vandermonde matrix.*

By contrast, the Vandermonde matrices (two-tone matrices) can achieve smallest storage overhead among all RID ones.

Theorem 3. *The storage overhead of an $[n, k]$ shift-XOR code with RID generator matrix is lower bounded by $\frac{(n(n-1))(k-1)}{2}$ and the lower bound is achieved if and only if the generator matrix is the Vandermonde matrix (two-tone matrices with divide 0).*

Corollary 2. *The storage overhead of an $[n, k]$ shift-XOR code with* systematic *RID generator matrix $\mathbf{\Psi} = \begin{bmatrix} \mathbf{I} \\ \mathbf{\Phi} \end{bmatrix}$ is lower bounded by $\frac{((n-k)(n-k-1))(k-1)}{2}$ and the lower bound is achieved if and only if $\mathbf{\Phi}$ is the Vandermonde matrix (two-tone matrices with divide 0).*

Systematic codes have the advantage of smaller storage overheads compared with the corresponding non-systematic codes. The storage overheads of different shift-XOR codes, including non-systematic and systematic versions, with some typical parameters $[n, k]$ are shown in Table I. From the table, we see that the storage overheads of systematic codes are less than 10% of that of the corresponding non-systematic codes. Moreover, two-tones systematic codes have about $1/3$ storage overheads of that of previous systematic codes.

TABLE II
COMPARISON OF ENCODING/DECODING THROUGHPUT FOR $[11, 8]$ CODES IN MEGABYTES PER SECOND.

(a) COMPARISON OF ENCODING THROUGHPUT

File Size	Two-tone	ISA-L	Jerasure	Longhair
128 KB	11,823	7,381	1,078	4,106
512 KB	12,256	7,628	1,272	4,097
1 MB	12,702	6,843	1,154	3,613
32 MB	12,086	6,436	773	2,283
64 MB	8,602	5,293	680	1,523
128 MB	7,952	5,177	659	1,455
256 MB	7,926	5,059	660	1,193
512 MB	7,888	4,992	592	1,110

(b) COMPARISON OF DECODING THROUGHPUT

File Size	Two-tone	ISA-L	Jerasure	Longhair
128 KB	8,021	5,205	783	3,368
512 KB	7,442	5,773	837	2,953
1 MB	7,474	4,762	840	2,519
32 MB	7,145	3,477	564	1,149
64 MB	5,898	3,089	467	859
128 MB	4,542	2,919	437	827
256 MB	4,531	2,730	476	710
512 MB	4,522	2,556	387	650

IV. IMPLEMENTATION AND PERFORMANCE ANALYSIS

Compared with a storage system using RS codes, one using two-tone codes may gain higher encoding/decoding throughput with the price of a slightly higher storage cost. In this section, we implement the two-tone shift-XOR codes and demonstrate the superior encoding/decoding throughputs compared with the state-of-the-art implementation of RS codes and Cauchy RS codes.

For storage codes employed by commercial distributed storage systems [4], [5], [21], n is commonly set to $1.33k$ to $2k$. Here we choose the parameter $[11, 8]$ to meet the settings in the real scenarios. We implement the two-tone codes and other libraries for comparison on an Intel Xeon CPU E5-2699 v4 at 2.2GHz. Our experiment evaluates the performance on 7 different file sizes from 128KB to 512MB. The comparison of encoding and decoding throughputs with existent coding libraries is shown in Table II.

From Table II, we observe that the two-tone code implementation outperforms all the other coding libraries in both encoding and decoding throughputs. The two-tone code implementation achieves from 50% to 100% more throughputs than the state-of-the-art coding library ISA-L for both encoding and decoding performance. Compared with Cauchy-RS codes implemented in the Longhair library, our codes can achieve 130% more encoding/decoding throughputs for small file size (128KB) and $5 \sim 8$ times the encoding/decoding throughputs for large file size (512MB). Compared with the RS codes implemented in Jerasure library, our codes can achieve 10 times the encoding/decoding throughput.

We observe that the throughputs drop significantly when the file size increases from 32MB to 64MB. This is because the size of cache in the experiment system is around 60MB.

REFERENCES

[1] I. 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] 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*, 1995.

[3] A. Fikes, "Storage architecture and challenges," *Talk at the Google Faculty Summit*, vol. 535, 2010.

[4] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, October 4-6, 2010, Vancouver, BC, Canada, Proceedings*, R. H. Arpaci-Dusseau and B. Chen, Eds. USENIX Association, 2010, pp. 61–74.

[5] D. Borthakur, R. Schmidt, R. Vadali, S. Chen, and P. Kling, "HDFS RAID," in *Hadoop User Group Meeting*, 2010.

[6] 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, 1993.

[7] M. Xiao, M. Médard, and T. Aulin, "A binary coding approach for combination networks and general erasure networks," in *IEEE International Symposium on Information Theory, ISIT 2007, Nice, France, June 24-29, 2007*. IEEE, 2007, pp. 786–790.

[8] M. Xiao, T. Aulin, and M. Médard, "Systematic binary deterministic rateless codes," in *2008 IEEE International Symposium on Information Theory, ISIT 2008, Toronto, ON, Canada, July 6-11, 2008*, F. R. Kschischang and E. Yang, Eds. IEEE, 2008, pp. 2066–2070.

[9] W. Shum and H. Hou, "Network coding based on byte-wise circular shift and integer addition," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 1641–1645.

[10] C. 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.

[11] X. Fu, Z. Xiao, and S. Yang, "Overhead-free in-place recovery scheme for XOR-based storage codes," in *IEEE Int. Conf. Trust, Security and Privacy in Computing and Communications*, Sep. 2014, pp. 552–557.

[12] ——, "Overhead-free in-place recovery and repair schemes of XOR-based regenerating codes," in *IEEE Int. Symp. Inf. Theory*, Jul. 2015, pp. 341–345.

[13] M. Dai, X. Wang, H. Wang, X. Lin, and B. Chen, "Bandwidth overhead-free data reconstruction scheme for distributed storage code with low decoding complexity," *IEEE Access*, vol. 5, pp. 6824–6832, 2017.

[14] 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.

[15] 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.

[16] M. Dai, B. Mao, X. Gong, C. W. Sung, W. Zhuang, and X. Lin, "Zigzag-division multiple access for wireless networks with long and heterogeneous delays," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 6, pp. 2822–2835, 2019.

[17] X. Fu, S. Yang, and Z. Xiao, "Decoding and repair schemes for shift-xor regenerating codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7371–7386, 2020.

[18] Intel, "Intel(R) Intelligent Storage Acceleration Library," 2020. [Online]. Available: https://github.com/intel/isa-l

[19] C. A. Taylor, "Longhair: Fast Cauchy Reed-Solomon Erasure Codes in C," 2018. [Online]. Available: https://github.com/catid/longhair

[20] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in c/c++ facilitating erasure coding for storage applications-version 1.2," *University of Tennessee, Tech. Rep. CS-08-627*, vol. 23, 2008.

[21] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *2012 USENIX Annual Technical Conference, Boston, MA, USA, June 13-15, 2012*, G. Heiser and W. C. Hsieh, Eds. USENIX Association, 2012, pp. 15–26.