Maximum Distance Separable Codes Based on Circulant Cauchy Matrices

Christian Schindelhauer and Christian Ortolf

Department of Computer Science, University of Freiburg, Germany
{schindel,ortolf}@informatik.uni-freiburg.de
http://cone.informatik.uni-freiburg.de/

Abstract. We present a maximum-separable-distance (MDS) code suitable for computing erasure resilient codes for large word lengths. Given n data blocks (words) of any even bit length w the Circulant Cauchy Codes compute $m \leq w+1$ code blocks of bit length w using XOR-operations, such that every combination of n data words and code words can reconstruct all data words. The number of XOR bit operations is at most $3nmw$ for encoding all check blocks. The main contribution is the small bit complexity for the reconstruction of $u \leq m$ missing data blocks with at most $9nuw$ XOR operations.

We show the correctness for word lengths of form $w = p - 1$ where p is a prime number for which two is a primitive root. We call such primes Artin numbers. We use efficiently invertible Cauchy matrices in a finite field $GF[2^p]$ for computing the code blocks To generalize these codes for all even word lengths w we use ℓ independent encodings by partitioning each block into sub-blocks of size $p_i - 1$, i.e. $w = \sum_{i=1}^{\ell} p_i - \ell$ for Artin numbers p_i. While it is not known whether infinitely many Artin numbers exist we enumerate all Circulant Cauchy Codes for $w \leq 10^5$ yielding MDS codes for all $m + n \leq \frac{10}{62}w$.

Keywords: RAID, erasure codes, storage, fault-tolerance.

1 Introduction

Computer systems are prone to data loss in many situations, ranging from the failure of a hard disk, communication errors in the physical layer, to the incomplete transmission of large files in overlay networks. Data is often stored in fixed block sizes and many systems rely on creating extra code blocks, from which one can recover the original data. A wide-spread example of such codes are RAID storage systems [14], where the parity of data blocks are stored on extra hard disks to recover the data. If the number of code blocks and data blocks necessary to restore a message equals the original number of data blocks, say n, such coding schemes are called *maximum distance separable (MDS)*.

We consider the input data partitioned into n blocks of w bits. We generate m additional check blocks with w bits each. MDS codes allow by definition to retrieve all n data blocks from any combination of the $n+m$ data and check blocks. MDS codes have been studied extensively and the standard method is Reed-Solomon codes [17]. These codes can be represented as a matrix multiplication over a finite field $\mathbb{F}[2^w]$. Addition in a Galois field is a bitwise XOR operation, while multiplication is a product of polynomials modulo a polynomial (and modulo base 2). The multiplication operation is the most time consuming part of such codes and therefore there have been research efforts to establish MDS codes only based on XOR operations.

It should be noted that Luby has found more efficient codes [11] based on XOR operations, spawning a lot of work in this area. However, these LT codes and successors are not MDS codes, and are thus not relevant for many uses, e.g. in RAID systems.

$$\begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \\ H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ c_1 \\ c_2 \end{pmatrix} \tag{1}$$

1.1 State of the Art

For RAID systems, MDS codings are considered where the original data is part of the encoding and the check blocks are generated by a matrix multiplication. Such MDS codes use a $nw \times (n+m)w$ *systematic* generator matrix G where the first nw columns are occupied with the identity matrix I, see (1). For the complexity of such an approach Blaum has proved in Proposition 3.4 [5] that at least $nw(m+1)$ nonzero entries are in the generator matrix, where n is the number of original blocks, m is the number of check blocks and w is the word length. For $\mathbb{F}_2$ he improves his bound in Proposition 5.2 to a minimum of $(m+1)nw + \frac{1}{2}\frac{mn}{1-1/n}$ entries of value 1 in the generator matrix. This corresponds to $nmw + \frac{1}{2}\frac{mn}{1-1/n}$ XOR operations for computing the check blocks, but does not imply a lower bound, since a small number of XOR operations can construct full matrices. Note that sparser matrices exist, if one drops the systematic matrix property, see [19].

In [4] MDS codes based on Galois-fields $\mathbb{F}[2^p]$ with generator polynomials $1 + x + \ldots + x^p$, where p is a prime number and 2 is a generator, i.e. $\{2^0, 2^1 \bmod p, \ldots, 2^{p-1} \bmod p\} = \{1, \ldots, p-1\}$ are shown. The authors use this approach to establish efficient coding for up to 8 check blocks. In this paper, we use the same generator polynomial and extend it to general block sizes and number of check blocks.

Many MDS codes like Even-Odd [3], Row-Diagonal Parity (RDP) [7], and Liberation Codes [15] construct only two parities and optimize on the number of XOR operation for computing the code blocks. In [2] RDP was generalized to compute up to eight check blocks while maintaining the optimal encoding complexity of RDP. An MDS code optimizing the reconstruction complexity has been presented in [12].

Blömer et al. [6] use Cauchy Reed-Solomon matrices to construct check matrices for systematic MDS codes. It turns out that the number of operations over the Galois field for reconstructing n data words is bounded by $O(nm)$ operations of the Galois field. In general, a multiplication in a Galois field can be performed in time $w \log w 2^{O(\log^* w)}$ using Fürer‘s integer multiplication algorithm [10]. For small word lengths, such multiplication can be performed on modern computers in constant time using table lookups with table size $O(2^w)$ and corresponding precomputing time. In [6] the authors recommend to precompute the factors for coding and decoding and word-parallel computation which reduces the complexity of computing m check blocks to $O(nmw)$ word-wise

XOR operations assuming the processor has word size w. Based on the approach of [6] Plank uses exhaustive search techniques to reduce the XOR complexity of such Cauchy Reed-Solomon matrices [16].

In [8] the usage of circular Boolean matrices helped to find an efficient MDS code for tolerating three disk failures. They showed that encoding takes $3wn$ XOR operations and decoding can be done within $3wn + 9(w + 1)$ XOR operations. Using these Cauchy Reed-Solomon matrices (aka. Rabin Codes) the same authors generalized this result in [9] to a MDS code with complexity of $9wn$ XOR operations for encoding and $O(w^3m^4)$ XOR operations for decoding when m disks fail (and $(9n + 95)(w + 1)$ for $m = 4$). The approach presented here has a similar coding complexity of $9mn$, but reduces the decoding complexity to $9nuw$ XOR operations.

1.2 Contribution

We present MDS codes, called Circulant Cauchy Codes, with an asymptotic optimal bit complexity for computing the check blocks and reconstructing data blocks. Bit complexity denotes the overall number of bit operations used in the calculation. In particular, we have an encoding bit complexity of $3nmw$ for computing m check blocks from n data blocks of word length w. The reconstruction of u data blocks from any n data or check blocks needs $9unw$ XOR bit operations. We prove that Circulant Cauchy Codes are defined for any even word length w.

However the maximum number of check blocks m depends on n and w. We prove that any number of check blocks can be generated if a conjecture of Emil Artin is true. It states that there are infinitely many prime numbers with two as primitive root. This conjecture does not give a good bound on the number of check blocks unless these prime numbers $A = \{3, 5, 11, 13, \ldots\}$ are dense. At least we can show that for all even $w \leq 10^5$ we can generate at least $m = \frac{10}{62}w - n$ check blocks. If w is a power of two we have evaluated that at least $\frac{29}{128}w - n$ parity check blocks can be generated for all $w \leq 2^{20}$. We conjecture that if w tends to infinity the number of possibly parity check blocks approaches $\frac{1}{4}w - n$.

An implementation can be downloaded from our website [18].

2 Circulant Boolean Matrices

The key to our efficient MDS codes are circulant Boolean matrices. A circulant matrix $n \times n$ matrix $A = \text{Cir}_n(s_1, \cdots, s_n)$ has the following form

$$\text{Cir}(s_0, \cdots, s_{n-1}) := (s_{i-j \mod n})_{i,j \in [n]} \tag{2}$$

We consider Boolean circulant matrices encoding bit-strings of length n. For $s_i \in \{0, 1\}$ we denote by $s = \sum_{i=0}^{p-1} s_i 2^i$ the matrix $\text{Cir}_n(s) = \text{Cir}(s_0, \ldots, s_{n-1})$ and operations are defined over $\mathbb{F}_2$ (XOR is addition and AND denotes the multiplication of bits). So, the cyclic shift by one position is denoted by a multiplication with $\text{Cir}(2)$. $\text{Cir}(1)$ is the neutral element for multiplication and $\text{Cir}(0)$ is the neutral element for addition, see Fig. 1.

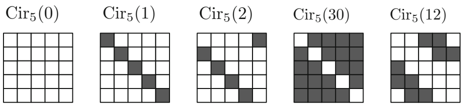

Fig. 1. Examples for $\mathrm{Cir}_5(x)$

It is well known, that Boolean circulant matrices describe a semiring. $\mathrm{Cir}(2)^i = \mathrm{Cir}(2^i)$ describe cyclic rotations by i positions. Since the rank of all Boolean circulant matrices is at most $n-1$, Boolean circulant matrices do not describe a finite field.

An alternative way to describe these semigroups is to consider them as monic polynomials $\sum_{i=0}^{n-1} s_i z^i$ modulo the monic polynomial $z^n + 1$ which is reducible, since $z^n + 1 = (z^{n-1} + \ldots + z + 1)(z + 1) \bmod 2$. Now, $z^{n-1} + \ldots + z + 1$ is irreducible if n is a prime number and 2 is a primitive root for p. Such finite fields have been already considered by Blaum et al. [4]. From now on we assume p to be such a prime with primitive root 2 and we call this set of numbers A.

Lemma 1. *For $p \in A$, i.e. a prime number where 2 is a primitive root modulo p, the set of all matrices $\mathcal{E}_p$ of $\mathrm{Cir}_p(s)$ for binary strings $s \in \{0,1\}^p$ with even parity forms a finite field for addition and multiplication as described above.*

Proof. If p is a prime number and 2 is a primitive root for m, then the polynomial $1 + z + \ldots + z^{p-1}$ is irreducible according to [13].

Note that $(z+1)(1 + z + \ldots + z^{p-1}) \equiv z^p + 1 \pmod 2$. Operations modulo the polynomial $z^p + 1$ do not describe a field, since the polynomial is not irreducible. However, addition and multiplication modulo $z^p + 1$ correspond to the addition and multiplication of circulant Boolean matrices.

Given $\mathrm{Cir}_p(a)$ with binary representation $a_0, \ldots, a_{p-1}$ we prove an operation preserving isomorphism described by the mapping $f : \mathcal{E}_p \to \{0,1\}^{p-1}$ where

$$f(\mathrm{Cir}_p(a_0, \ldots, a_{p-1})) = (a_i + a_{p-1})_{i \in \{0,\ldots,p-2\}} \tag{3}$$

and the inverse function is

$$f^{-1}((a_i)_{i\in\{0,\ldots,p-2\}}) = \mathrm{Cir}_p\left(\left(\sum\nolimits_{j\in\{0,\ldots,p-2\}\setminus\{i\}} a_j\right)_{i\in\{0,\ldots,p-1\}}\right) \tag{4}$$

We prove that

$$f(\mathrm{Cir}_p(a) + \mathrm{Cir}_p(b)) \equiv f(a) + f(b) \pmod{1 + z + \ldots + z^{p-1}} \tag{5}$$
$$f(\mathrm{Cir}_p(a) \cdot \mathrm{Cir}_p(b)) \equiv f(a) f(b) \pmod{1 + z + \ldots + z^{p-1}} \tag{6}$$

Note that for $a, b \in \mathcal{E}_p$ we have $a + b \in \mathcal{E}_p$ and $a \cdot b \in \mathcal{E}_p$. From the irreducibility of $1 + z + \ldots + z^{p-1}$ the claim follows since we have an isomorphism to a finite field.

It remains to prove (5) and (6).

1. Addition

$$\begin{aligned} & f(\mathrm{Cir}_p(a) + \mathrm{Cir}_p(b)) \\ &= (a_i + b_i + a_{p-1} + b_{p-1})_{i\in\{0,\ldots,p-2\}} \\ &= f(\mathrm{Cir}_p(a)) + f(\mathrm{Cir}_p(b)) \end{aligned} \tag{7}$$

2. Multiplication: Note that $f(\text{Cir}_p(2^p-3)) = (0,1,0,\ldots,0) = x$, $f(\text{Cir}_p(2^p-1)) = 1$, $f(\text{Cir}_p(0)) = 0$, $f(\text{Cir}_p(2^p-3)^i) = f(\text{Cir}_p(2^p-1-2^{i \bmod p-2})) = x^i$. Furthermore,

$$\begin{aligned} f(\text{Cir}_p(2^p-3)) \cdot a &= f(\text{Cir}_p(a_{i-1 \bmod p})_{i\in\{0,\ldots,p-1\}} \\ &= (a_{i-1 \bmod p} + a_{p-2})_{i\in\{0,\ldots,p-2\}} \\ &= \sum_{i=0}^{p-2} (a_i + a_{p-1}) x^{i+1} \\ &= f(\text{Cir}_p(a)) \cdot x \end{aligned} \tag{8}$$

since $x^{p-1} \equiv \sum_{i=0}^{p-2} x^i \pmod{1 + x + x^2 + \ldots + x^{p-1}}$.
So, for $\text{Cir}_p(b) = \sum_{i=0}^{p-1} b_i \text{Cir}_p(2)^i$ we have

$$\begin{aligned} f(\text{Cir}_p(a) \cdot \text{Cir}_p(b)) &= f\left(\text{Cir}_p(a) \cdot \sum_{i=0}^{p-1} b_i \text{Cir}_p(2)^i\right) \\ &= f\left(\sum_{i=0}^{p-1} b_i \text{Cir}_p(a) \text{Cir}_p(2^p-3)^i\right) \\ &= \sum_{i=0}^{p-1} b_i f(\text{Cir}_p(a)\text{Cir}_p(2^p-3)^i)) \\ &= \sum_{i=0}^{p-1} f(\text{Cir}_p(a)) b_i x^i \\ &= \sum_{i=0}^{p-2} f(\text{Cir}_p(a))(b_i + b_{p-2}) x^i \\ &= f(\text{Cir}_p(a)) \cdot f(\text{Cir}_p(b)) \end{aligned} \tag{9}$$

Using the observation that Circulant matrices with even parity form a finite field, one can reduce the number of XOR operations. For this, we use the complement of an element as an alternative representation. So, define for each element $a \in [0, 2^{p-1} - 1]$: $[a] := \{\text{Cir}(a), \text{Cir}(2^p - 1 - a)\}$. Define the addition over these sets $[a] + [b] = \{x + y \mid x \in [a], y \in [b]\}$ and similarly, $[a] \cdot [b] = \{x \cdot y \mid x \in [a], y \in [b]\}$. Now, the observation of Figures 2 and 3 can be generalized as follows.

$$[a + b] = [a] + [b] \,, \tag{10}$$

$$[a \cdot b] = [a] \cdot [b] \,. \tag{11}$$

When coding or decoding all inputs $a \in \{0,1\}^{p-1}$ of word size $p - 1$ are mapped to such sets $[a]$ of word size p at the beginning. All subsequent operations will be done on the increased word size p since in the representation fewer XOR operations are needed. Therefore, we do not distinguish between both representations $[a] = \{a, 2^p - 1 - a\}$. Eventually, we eliminate the ambiguity by a final computation where we choose the

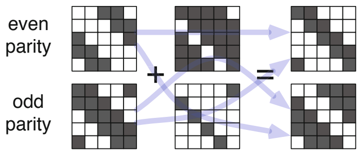

Fig. 2. Example for addition $[a + b]$

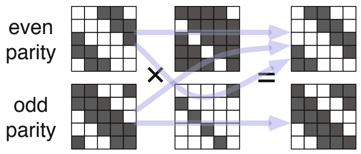

Fig. 3. Example for multiplication $[a] \cdot [b]$

element of which the least significant bit is 0. This final operation costs $p - 1$ XOR operations and reduces the output to the original word length $p - 1$. See Fig. 4. The transition from the data to one of its representation can be denoted by multiplication with matrix R, which is just a copy operation and its reverse operation with matrix L where $p - 1$ XORs are needed.

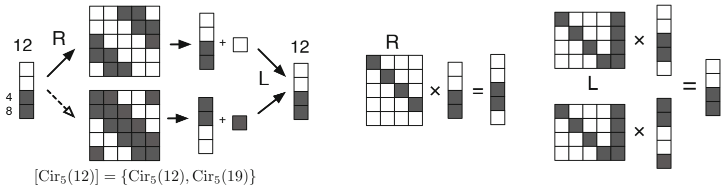
12
R
4
8
+
+
L
12
R
×
=
L
×
×
=
$[\mathrm{Cir}_5(12)] = \{\mathrm{Cir}_5(12), \mathrm{Cir}_5(19)\}$

Fig. 4. Example for representation of 12 in $[\mathrm{Cir}_5(12)]$

Lemma 2. *The basic operations in the table below need the given number of XOR operations for input word length $w = p - 1$, where i and j are constants and a, b input variables.*

Operation	*XOR operations*
$R \cdot a = \mathrm{Cir}_p(a)$	0
$L \cdot \mathrm{Cir}(a) = a$	w
$\mathrm{Cir}(a) + \mathrm{Cir}(b)$	$w + 1$
$\mathrm{Cir}(2)^i \cdot a$	0
$\mathrm{Cir}(2^i + 2^j) \cdot a$	$w + 1$
$[\mathrm{Cir}(2^i + 2^j)^{-1} \cdot a]$	$2w - 1$

Proof. 1. $a \mapsto \mathrm{Cir}_p(a)$: For this operation we simply append a constant 0 to the representation.
2. $Cir(a) \mapsto (a_i + a_p)_{i \in \{0,\ldots,p-1\}}$: This operation is necessary to produce the output and to transform the representation of a by Circulant Boolean matrices to the corresponding finite field element. Clearly, p XORs are sufficient to compute the result.
3. $\mathrm{Cir}(a) + \mathrm{Cir}(b)$: We compute pairwise XORs of the corresponding bits of a and b.
4. $\mathrm{Cir}(2)^i \cdot a$: For constant i this constitutes a clock-wise right shift by i steps. No XOR operations are necessary.

5. $\mathrm{Cir}(2^i + 2^j) \cdot a$: The complexity follows by computing $\mathrm{Cir}(2)^i \cdot a + \mathrm{Cir}(2)^j \cdot a$.
6. $[\mathrm{Cir}(2^k + 2^\ell)^{-1} \cdot a]$: This is the only non-trivial case. Note that

$$\mathrm{Cir}(2^k + 2^\ell) \cdot \begin{pmatrix} b_0 \\ \vdots \\ b_{p-1} \end{pmatrix} = \begin{pmatrix} a_0 \\ \vdots \\ a_{p-1} \end{pmatrix} . \tag{12}$$

This is equivalent to

$$a_{i+\ell} + \sum_j a_j = b_{i \bmod p} + b_{i+(\ell-k) \bmod p} \tag{13}$$
$$\text{for all } i \in \{0, \ldots, p-1\}.$$

Since, any of the two elements of $[a] = \{a, 2^p - 1 - a\}$ is allowed as result we choose $b_0 = 0$. So we get for $i \in \{0, \ldots, p-2\}$

$$\begin{aligned} b_{(i+1)2(\ell-k) \bmod p} = {} & b_{2i(\ell-k) \bmod p} \\ & + a_{2(i+1)(\ell-k)+k \bmod p} \\ & + a_{(2i+1)(\ell-k)+k \bmod p} \,. \end{aligned} \tag{14}$$

The calculation for an example can be seen in Fig. 5

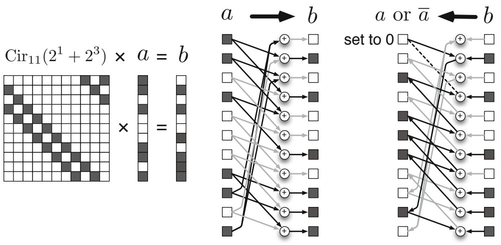

Fig. 5. Multiplication and division by $\mathrm{Cir}_{11}(9)$

It is straightforward that the result is either b or $2^p - b$. The result might be inverted, but this is acceptable, as we have described above.

3 Circulant Cauchy Matrix

Cauchy Reed-Solomon matrices have been introduced for MDS codes by Blömer et al. [6]. For n data blocks and m check blocks of word length w we use a $m \times n$ Cauchy matrix where each entry is a Boolean Circulant matrix defined as

$$M_{ij} = \frac{1}{x_i + y_j} \tag{15}$$

for $x_i \neq y_j$ for all $i \in \{1,\dots,m\}, j \in \{1,\dots,n\}$. For the Circulant Cauchy matrix we choose $x_i = \mathrm{Cir}_p(2^{i-1})$ for $i \in \{1,\dots,m\}$ and $y_1 = 0$, $y_j = \mathrm{Cir}_p(2^{p-j})$ for $j \in \{2,\dots,n\}$. Therefore for $n+m \leq p+1$ the sets $\{x_1,\dots,x_n\}$ and $\{y_1,\dots,y_m\}$ are distinct which is the prerequisite for the full rank property of the Cauchy matrix.

So the check blocks $c_1,\dots,c_m \in \{0,1\}^{p-1}$ are computed from the data block $d_1,\dots,d_n \in \{0,1\}^{p-1}$ by

$$\begin{pmatrix} L & & 0 \\ & \ddots & \\ 0 & & L \end{pmatrix} \cdot M \cdot \begin{pmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{pmatrix} \cdot \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} \tag{16}$$

where L is a $(p-1) \times p$ matrix with $L_{ii} = 1$, $L_{i,p} = 1$ for $i \in \{1,\dots,p-1\}$ and $L_{i,j} = 0$ elsewhere. R is a $p \times (p-1)$ matrix with $R_{ii} = 1$ and $R_{i,p} = 0$ elsewhere.

Theorem 1. *The computation of m check blocks of the Circulant Cauchy matrix from n data blocks of block size w can be computed with $(3m-2)nw+n-m = (3+o(1))nmw$ XOR-operations.*

Proof. The transformation from d_i to a circulant matrix does not use any operations. The multiplication with the first row of the Circulant Cauchy matrix is a cyclic shift operation since $x_1 = \mathrm{Cir}_p(0)$ which does not involve any XOR bit operation. For the multiplication of the residual $m-1$ operations we divide by terms of the form $\mathrm{Cir}_p(2^k+2^\ell)$. Lemma 2 states that $2w-1$ XOR operations are sufficient resulting in $(m-1)n(2w-1)$ XOR operations. For adding all results we need $(n-1)m(w+1)$ XOR operations. A multiplication with an L matrix needs w XOR operations resulting in mw operations.

So, the overall number of XOR operations is $(n-1)m(w+1) + n(m-1)(2w-1) + mw = 3mnw - 2nw - m + n$.

Theorem 2. *The Circulant Cauchy matrix is an MDS code, i.e. from every set of n data or check blocks the data can be recovered, if $n+m \leq p+1$.*

Proof. The determinant of a $n \times n$ Circulant Cauchy matrix $M_{ij} = \left(\frac{1}{x_i+y_j}\right)$ is

$$\det(M) = \frac{\prod_{i<j}(x_i+x_j)(y_i+y_j)}{\prod_{i,j} x_i + y_j} \tag{17}$$

Let $C_{rs} = M_{rs}$ be the adjugate matrix of M (deleting row i and column j). Note that the adjugate matrix is again a Circulant Cauchy matrix.

$$\det(C_{rs}) = \frac{\prod_{i<j,i,j\neq r}(x_i+x_j)\prod_{i<j,i,j\neq s}(y_i+y_j)}{\prod_{i\neq r,j\neq s} x_i + y_j} \tag{18}$$

Now for the inverse matrix d_{ij} we have

$$d_{ij} = \frac{\det(C_{ji})}{\det(M)} = \frac{\prod_u x_j + y_u \prod_u x_u + y_i}{\prod_{u\neq j}(x_u+x_j)\prod_{u\neq i}(y_u+y_i)} \frac{1}{(x_i+y_j)} \tag{19}$$

For computing the data blocks, we choose the square submatrix B of the generator matrix corresponding to the indices of the u unknown data blocks and the k given check blocks. Note that B is a Circulant Cauchy matrix and thus invertible. The $k \times n - k$ submatrix A corresponds to the given data blocks and unknown check blocks. So, we can calculate the missing data blocks by

$$\begin{pmatrix} d_{i_1} \\ \vdots \\ d_{i_u} \end{pmatrix} = B^{-1} \left(\begin{pmatrix} c_{j_1} \\ \vdots \\ c_{j_u} \end{pmatrix} + A \cdot \begin{pmatrix} d_{g_1} \\ \vdots \\ d_{g_{n-u}} \end{pmatrix} \right) \tag{20}$$

where $i_1, \ldots, i_u$ are the indices of the unknown data blocks, $j_1, \ldots j_u$ the indices of the given check blocks, and $g_1, \ldots, g_{n-u}$ the indices of the given data blocks.

Our main contribution is the efficiency of this operation.

Theorem 3. *Given $n - u$ data blocks and u check blocks of a Circulant Cauchy Code the missing u data blocks can be computed with at most $3nuw + 6u^2w \leq 9nuw$ XOR bit operations.*

Proof. Using Equation 20 the given input $d_{g_1}, \ldots, d_{g_{n-k}}$ is multiplied with the submatrix $k \times (n-k)A$. From Theorem 1 this can be done with $3u(n-u)w - 2(n-u)w + n - 2u = 3nuw - 3u^2w - 2nw + 2uw - 2u$ XOR operations. With another uw XOR operations the result is added to the given u check blocks of word size w giving the intermediate result $y_1, \ldots, y_u \in \{0,1\}^{w+1}$.

Then, the Circulant Cauchy matrix is reduced to the $u \times u$ sub-matrix B corresponding to the check blocks which results in a smaller Circulant Cauchy matrix. Following the approach in [6] the inverse $(d_{ij})_{i,j \in 1,\ldots,n}$ of a Circulant Cauchy matrix $(\frac{1}{x_i+y_j})_{i,j}$ can be computed as follows.

$$\begin{aligned} a_k &= \prod_{i \neq k} (x_i + x_k) \, , \; b_k = \prod_{i \neq k} (y_i + y_k) \, , \quad e_k = \prod_{i=1}^{n} (x_k + y_i) \, , \\ f_k &= \prod_{i=1}^{n} (x_i + y_k) \, , \; d_{ij} = \frac{e_i f_j}{a_i b_j (x_i + y_j)} \, . \end{aligned} \tag{21}$$

We multiply the inverse with the intermediate result $y_1, \ldots, y_u$. We have to compute the data blocks $d_1, \ldots, d_u$ (WLOG we assume that the first u data blocks need to be restored) such that for all $j \in \{1, \ldots, u\}$

$$d_i = \sum_{j=1}^{u} d_{ij} y_j = \sum_{j=1}^{u} \frac{e_i f_j}{a_i b_j (x_i + y_j)} y_j = \frac{e_i}{a_i} \sum_{j=1}^{u} \frac{1}{(x_i + y_j)} \frac{f_j}{b_j} y_j \, . \tag{22}$$

First we compute for all $i \in \{1, \ldots, u\}$

$$C'_j = \frac{f_j}{b_j} C_j = \frac{\prod_{i=1}^{n} (x_i + y_j)}{\prod_{i \neq j} (y_i + y_j)} C_j \tag{23}$$

This results in u^2 multiplications with terms of the form Cir$(2^\nu + 2^\eta)$ and $u(u-1)$ divisions by Cir$(2^\nu + 2^\eta)$. Some factors or divisors may be of form Cir(2^ν) since we choose $x_1 = 0$. This case only reduces the complexity and is from now on omitted for simplicity. The number of XOR operations is therefore at most $u(u-1)(w+1) + u^2(2w-1) = 3u^2w - uw - u$. Then, we compute all terms of the form $\frac{C'_j}{x_i+y_j}$ which takes at most $u^2(2w-1) = 2u^2w - u^2$ XOR operations.

The following sum costs $u(u-1)(w+1) = u^2w - uw + u^2 - u$ XOR operations. Finally each of the u sums need to be multiplied by

$$\frac{e_i}{a_i} = \frac{\prod_j (x_i + y_j)}{\prod_{j \neq i}^{u} (x_i + x_j)} \tag{24}$$

So, u^2 multiplications by terms of the form $\mathrm{Cir}(2^i + 2^j)$ are necessary ($w+1$ single XOR operations each) and $u(u-1)$ divisions by terms of form $\mathrm{Cir}(2^i + 2^j)$ with ($2w-1$ XOR operations). Hence, $u^2(w+1) + u(u-1)(2w-1) = 3u^2w - 2uw + u$ operations suffice.

Finally, the size of the u data bits of word length $w+1$ need to be reduced to size w adding another uw Xor operations.

The overall number of XOR bit operations is therefore at most $3nuw + 6u^2w - 2nw - 4u \leq 3nuw + 6u^2w \leq 9nuw$.

Because of the run-time of $3nmw$ for computing the check blocks and $9nuw$ for reconstructing the data, the length of w does not play any role. It is advisable to choose w as large as possible, since it increases the number of possible code words and does not change the run-time, e.g. if the input consist of N bits, then the complexity for computing all the m check blocks is $3\frac{N}{w}nw = 3Nm$ and similarly for reconstruction N data bits from u check blocks: $9\frac{N}{w}uw = 9Nu$. One might argue that usually the input size is not a multiple of w (since we have a restriction on $w+1$ being an Artin number). However, we overcome this problem in the next section where we show that w can be chosen to be any even number.

4 Generalizing the Word Length

While Artin numbers do not appear to be scarce, it is an open problem first conjectured by Emil Artin in 1927 [1] (p. 246) whether an infinite number of such prime numbers exist. On the positive side there are efficient methods to test the Artin number property.

Most notably, the run-time grows only linearly with the length of the words. Therefore, for the time complexity partitioning the data in word size $w = 4$ or $w = 1018$ does not make a difference for the run-time. Since larger word sizes allow more redundancy and the combination of more data blocks, it is advisable to increase it as large as the CPU cache and the block size of the data allows.

We now overcome the restriction that the word size has the form $w = p - 1$, where p is an Artin number.

Fact 1. *A partition of an integer w is valid if*

$$w = \sum_{i=1}^{\ell} (p_i - 1) \,, \tag{25}$$

where p_i are Artin primes, i.e. 2 is a primitive root of p_i.

- *There exists a valid partition for every even number $w \leq 10^5$ such that the minimum term is at least $\frac{10}{62}w$.*
- *If $w \leq 2^{20}$ is a power of two the minimum term of a valid partition is at least $\frac{28}{128}w$.*

We have verified this fact using computer algebra programs and exhaustive testing.

To overcome the word size restriction we us a valid partition and apply the Circulant Cauchy codes for each of the sub-words separately. The MDS property is preserved and also for the run-time we get $3nm(w_1 + \ldots + w_k) = 3nmw$ XOR operations for encoding and $9nuw$ for decoding. Since, the number of reconstructable blocks is limited by $w_i + 1$ it is desirable to maximize the smallest term. While it is an open question whether infinitely many Artin primes exist, the above fact shows that the density for numbers up to 10^5 is high enough to guarantee good partitions.

Clearly, the most interesting word lengths are powers of two. The following valid partitions maximize the size of the smallest subword size.

$$\begin{array}{rr} 8 = 4 + 4 & 16 = 4 + 12 \\ 32 = 10 + 10 + 12 & 64 = 28 + 36 \\ 128 = 28 + 100 & 256 = 60 + 196 \\ 512 = 196 + 316 & 1024 = 372 + 652 \\ 2048 = 940 + 1108 & 4096 = 2028 + 2068 \\ 8192 = 3796 + 4396 & 16384 = 8116 + 8268 \end{array} \tag{26}$$

5 Conclusions

We have presented a new approach to MDS codes using long word lengths w which after adding a parity bit to a given word uses only cyclic shift operations and bit-wise XOR-operations. Our method allows the generation of arbitrarily many check blocks and arbitrarily large word sizes. It is based on the isomorphism between Boolean circulant matrices and finite fields where $w + 1$ is a prime number and 2 is a primitive root modulo $w + 1$. We show how this method can be applied to arbitrary word length by a partitioning of the words without an impact to the coding and decoding complexity. We use Cauchy Reed-Solomon codes and present the first MDS scheme with asymptotical optimal XOR complexity for computing the check blocks and recovering data blocks. Furthermore, the constant factors are small and these codes can be easily implemented on existing computer architectures.

References

1. Artin, E., Hasse, H., Frei, G., Roquette, P., Lemmermeyer, F.: Emil Artin und Helmut Hasse: Die Korrespondenz, 1923-1934. In: Einleitung in engl. Sprache, Universitätsverlag Göttingen (2008)
2. Blaum, M.: A family of mds array codes with minimal number of encoding operations. In: 2006 IEEE International Symposium on Information Theory, pp. 2784–2788 (July 2006)
3. Blaum, M., Brady, J., Bruck, J., Menon, J.: EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. IEEE Trans. Computers 44(2), 192–202 (1995)

4. Blaum, M., Bruck, J., Vardy, A.: MDS array codes with independent parity symbols. IEEE Transactions on Information Theory 42(2), 529–542 (1996)
5. Blaum, M., Roth, R.M.: On lowest density MDS codes. IEEE Transactions on Information Theory 45(1), 46–59 (1999)
6. Blömer, J., Kalfane, M., Karpinski, M., Karp, R.M., Luby, M., Zuckerman, D.: An XOR-based erasure-resilient coding scheme. ICSI Technical Report TR-95-048, ICSI (August 1995)
7. Corbett, P., English, B., Goel, A., Grcanac, T., Kleiman, S., Leong, J., Sankar, S.: Row-diagonal parity for double disk failure correction. In: Proceedings of the USENIX FAST 2004 Conference on File and Storage Technologies, pp. 1–14. Network Appliance, Inc., USENIX Association, San Francisco (2004)
8. Feng, G.-L., Deng, R.H., Bao, F., Shen, J.-C.: New efficient mds array codes for raid part i: Reed-solomon-like codes for tolerating three disk failures. IEEE Trans. Comput. 54(9), 1071–1080 (2005)
9. Feng, G.-L., Deng, R.H., Bao, F., Shen, J.-C.: New efficient mds array codes for raid part ii: Rabin-like codes for tolerating multiple (greater than or equal to 4) disk failures. IEEE Trans. Comput. 54(12), 1473–1483 (2005)
10. Fürer, M.: Faster integer multiplication. In: ACM (ed.) STOC 2007: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, pages 57–66. ACM Press (2007); pub-ACM:adr
11. Luby, M.: L.T Codes. In: IEEE (ed.) Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2002, Vancouver, BC, Canada, November 16-19, pp. 271–280. IEEE Computer Society Press (2002); pub-IEEE:adr
12. Luo, J., Xu, L.: Scan: An efficient decoding algorithm for raid-6 codes. In: 2011 10th IEEE International Symposium on Network Computing and Applications (NCA), pp. 91–98 (August 2011)
13. Menezes, A.J., Blake, I.F., Gao, X., Mullin, R.C., Vanstone, S.A., Yaghoobian, T. (eds.): Applications of Finite Fields. Kluwer Academic Publishers (1993)
14. Patterson, D.A., Gibson, G., Katz, R.H.: A Case for Redundant Arrays of Inexpensive Disks (RAID). In: Proceedings of the ACM Conference on Management of Data (SIGMOD), pp. 109–116 (1988)
15. Plank, J.S.: The RAID-6 liberation codes. In: Baker, M., Riedel, E. (eds.) 6th USENIX Conference on File and Storage Technologies, FAST 2008, pp. 97–110, February 26-29. USENIX, San Jose (2008)
16. Plank, J.S., Xu, L.: Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications. In: Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications, NCA 2006, pp. 173–180. IEEE Computer Society, Washington, DC (2006)
17. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics 8(2), 300–304 (1960)
18. Schindelhauer, C.: C implementation of our circulant cauchy based codes, `http://archive.cone.informatik.uni-freiburg.de/pubs/CirculantCauchy.zip`
19. Zaitsev, G.V., Zinovev, V.A., Semakov, N.V.: Minimum check-density codes for correcting bytes of errors, erasures, or defects. Probl. Inform. Transm. 19(3), 29–37 (1983)