ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Distributed Storage Networks and Computer Forensics

5 Raid-6 Encoding

Christian Schindelhauer

University of Freiburg
Technical Faculty
Computer Networks and Telematics
Winter Semester 2011/12

CoNe Freiburg

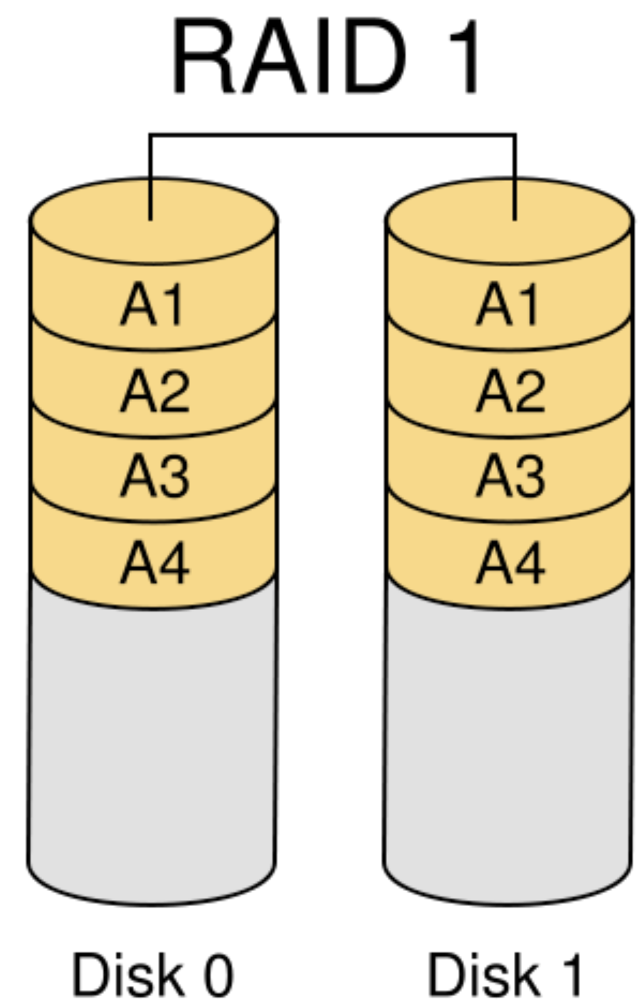
IIF INSTITUT FÜR INFORMATIK FREIBURG

RAID

- **Redundant Array of Independent Disks**
  - Patterson, Gibson, Katz, „A Case for Redundant Array of Inexpensive Disks“, 1987
- **Motivation**
  - Redundancy
    - error correction and fault tolerance
  - Performance (transfer rates)
  - Large logical volumes
  - Exchange of hard disks, increase of storage during operation
  - Cost reduction by use of inexpensive hard disks

Raid 1

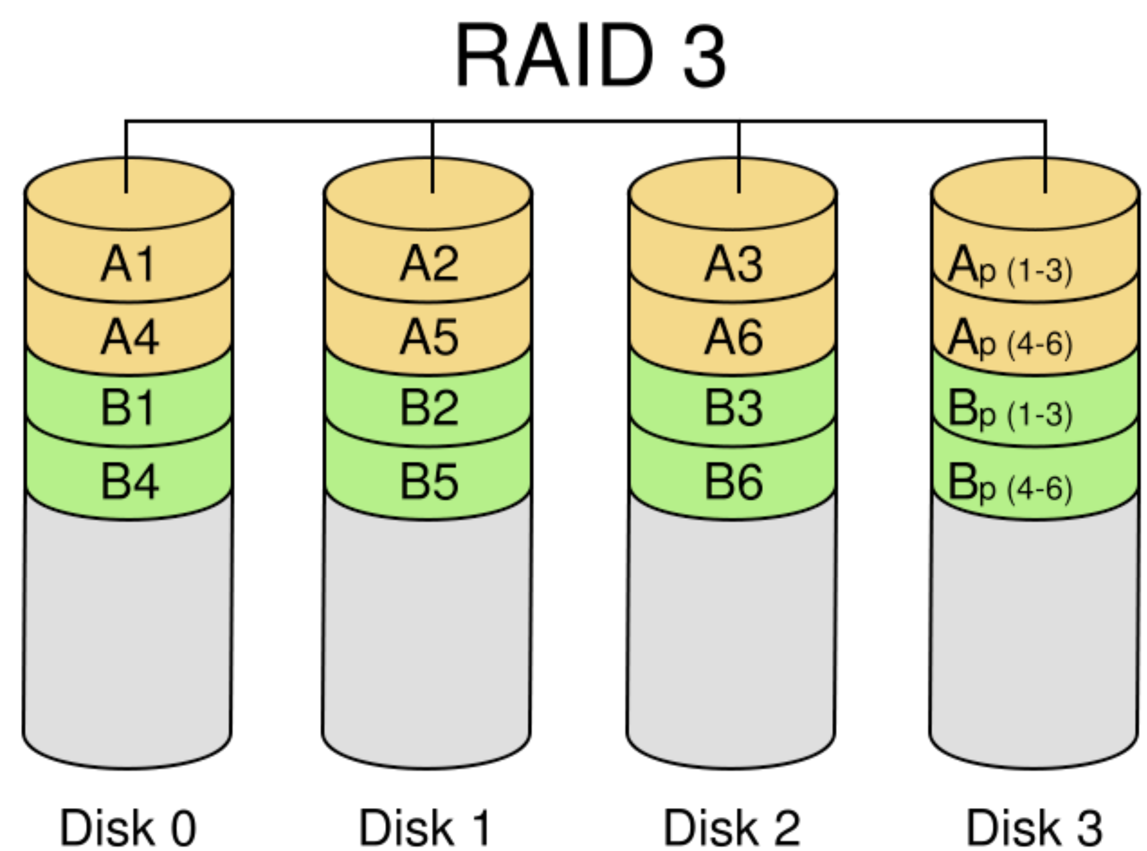
- **Mirrored set without parity**
  - Fragments are stored on all disks
- **Performance**
  - if multi-threaded operating system allows split seeks then
  - faster read performance
  - write performance slightly reduced
- **Error correction or redundancy**
  - all but one hard disks can fail without any data damage
- **Capacity reduced by factor 2**



http://en.wikipedia.org/wiki/RAID

Raid 3

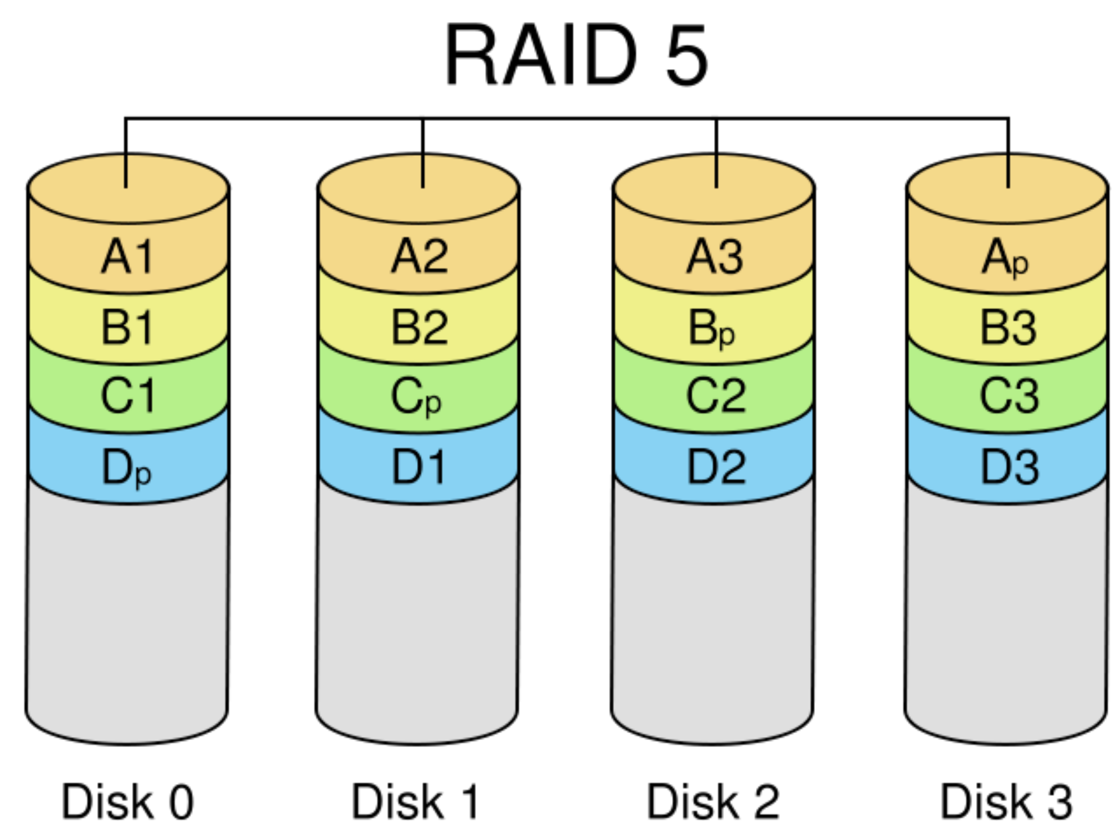
- **Striped set with dedicated parity (byte level parity)**
  - Fragments are distributed on all but one disks
  - One dedicated disk stores a parity of corresponding fragments of the other disks
- **Performance**
  - improved read performance
  - write performance reduced by bottleneck parity disk
- **Error correction or redundancy**
  - one hard disks can fail without any data damage
- **Capacity reduced by 1/n**



http://en.wikipedia.org/wiki/RAID

Raid 5

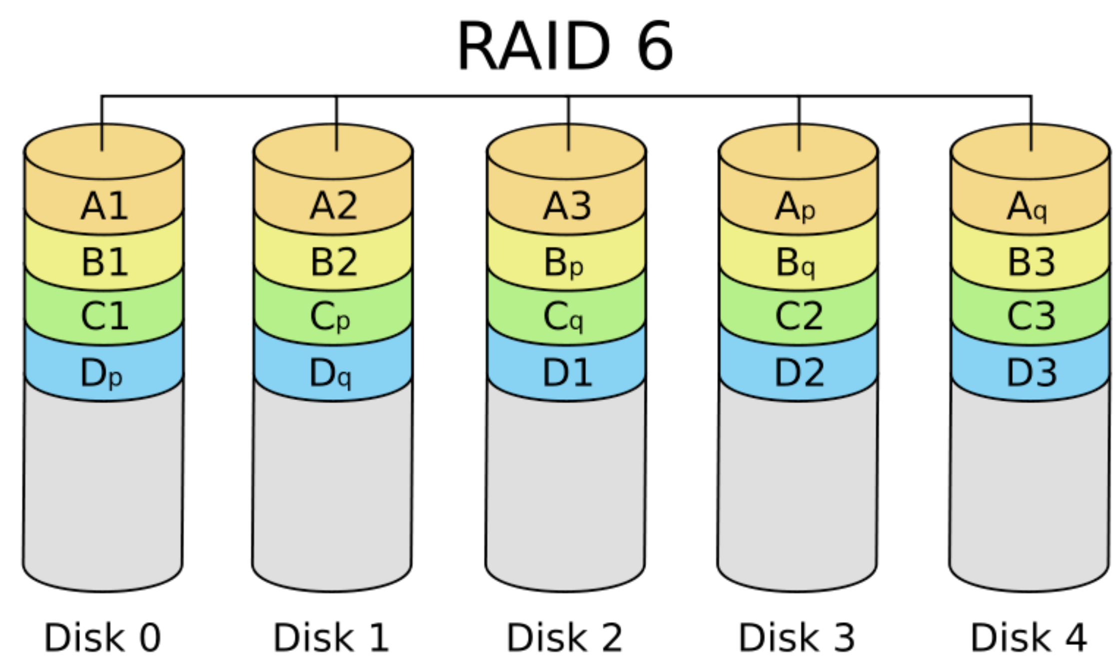
- **Striped set with distributed parity (interleave parity)**
  - Fragments are distributed on all but one disks
  - Parity blocks are distributed over all disks
- **Performance**
  - improved read performance
  - improved write performance
- **Error correction or redundancy**
  - one hard disks can fail without any data damage
- **Capacity reduced by 1/n**



http://en.wikipedia.org/wiki/RAID

Raid 6

- **Striped set with dual distributed parity**
  - Fragments are distributed on all but two disks
  - Parity blocks are distributed over two of the disks
    - one uses XOR other alternative method
- **Performance**
  - improved read performance
  - improved write performance
- **Error correction or redundancy**
  - two hard disks can fail without any data damage
- **Capacity reduced by 2/n**



http://en.wikipedia.org/wiki/RAID

Algorithms and Methods for
Distributed Storage Networks

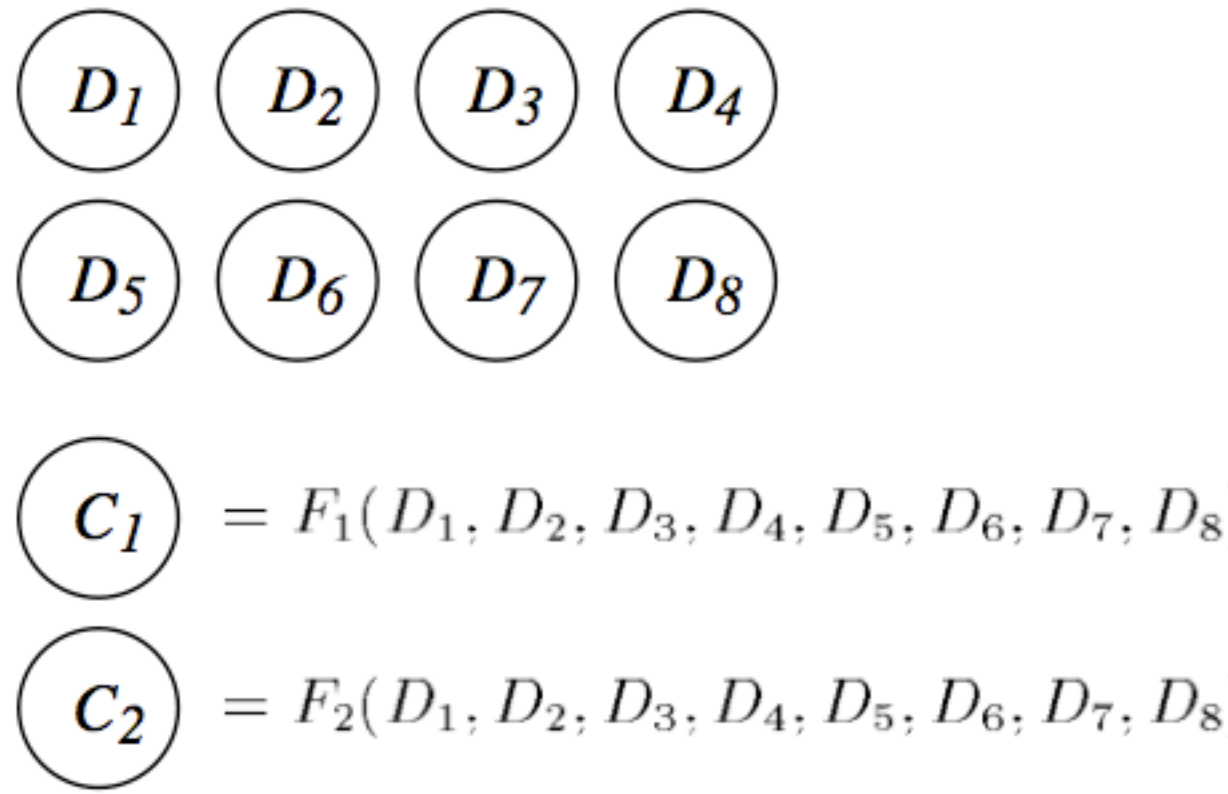
RAID 6 - Encodings

Literature

- **A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank , 1999**
- **The RAID-6 Liberation Codes, James S. Plank, FAST ´08, 2008**

Principle of RAID 6

- **Data units $D_1, \ldots, D_n$**
  - w: size of words
    - w=1 bits,
    - w=8 bytes, ...
- **Checksum devices $C_1, C_2, \ldots, C_m$**
  - computed by functions $C_i = F_i(D_1, \ldots, D_n)$
- **Any n words from data words and check words**
  - can decode all n data units



A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank , 1999

Principle of RAID 6

D_1	D_2	C_1	C_2
$d_{1,1}$	$d_{2,1}$	$c_{1,1} = F_1(d_{1,1}, d_{2,1})$	$c_{2,1} = F_2(d_{1,1}, d_{2,1})$
$d_{1,2}$	$d_{2,2}$	$c_{1,2} = F_1(d_{1,2}, d_{2,2})$	$c_{2,2} = F_2(d_{1,2}, d_{2,2})$
$d_{1,3}$	$d_{2,3}$	$c_{1,3} = F_1(d_{1,3}, d_{2,3})$	$c_{2,3} = F_2(d_{1,3}, d_{2,3})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$d_{1,l}$	$d_{2,l}$	$c_{1,l} = F_1(d_{1,l}, d_{2,l})$	$c_{2,l} = F_2(d_{1,l}, d_{2,l})$

Figure 2: Breaking the storage devices into words ($n = 2, m = 2, l = \frac{8k}{w}$)

A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank , 1999

Operations

- **Encoding**
  - Given new data elements, calculate the check sums
- **Modification (update penalty)**
  - Recompute the checksums (relevant parts) if one data element is modified
- **Decoding**
  - Recalculate lost data after one or two failures
- **Efficiency**
  - speed of operations
  - check disk overhead
  - ease of implementation and transparency

RAID 6 Encodings

Reed-Solomon

Vandermonde-Matrix

$$\begin{bmatrix} f_{1,1} & f_{1,2} & \dots & f_{1,n} \\ f_{2,1} & f_{2,2} & \dots & f_{2,n} \\ \vdots & \vdots & & \vdots \\ f_{m,1} & f_{m,2} & \dots & f_{m,n} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \dots & n^{m-1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$

A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank , 1999

Complete Matrix

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \dots & n^{m-1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$

A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank , 1999

Galois Fields

- **$GF(2^w)$ = Finite Field over 2^w elements**
  - Elements are all binary strings of length w
  - $0 = 0^w$ is the neutral element for addition
  - $1 = 0^{w-1}1$ is the neutral element for multiplication
- **u + v = bit-wise Xor of the elements**
  - e.g. 0101 + 1100 = 1001
- **a b= product of polynomials modulo 2 and modulo an irreducible polynomial q**
  - i.e. $(a_{w-1} \ldots a_1\ a_0)\ (b_{w-1} \ldots b_1\ b_0) =$

$$((a_0 + a_1 x + \ldots + a_{w-1} x^{w-1})(b_0 + b_1 x + \ldots + b_{w-1} x^{w-1}) \bmod q(x)) \bmod 2)$$

Example: $GF(2^2)$

Generated Element of $GF(4)$	Polynomial Element of $GF(4)$	Binary Element b of $GF(4)$	Decimal Representation of b
0	0	00	0
x^0	1	01	1
x^1	x	10	2
x^2	$x+1$	11	3

+	0 = 00	1 = 01	2 = 10	3 = 11
0 =00	0	1	2	3
1 =01	1	0	3	2
2 =10	2	3	0	1
3 =11	3	2	1	0

$q(x) = x^2+x+1$

*	0 = 0	1 = 1	2 = x	3 = x+1
0 = 0	0	0	0	0
1 = 1	0	1	2	3
2 = x	0	2	3	1
3 = x+1	0	3	1	2

$2 \cdot 3 = x(x+1) = x^2+x = 1 \bmod x^2+x+1 = 1$

$2 \cdot 2 = x^2 = x+1 \bmod x^2+x+1 = 3$

Irreducible Polynomials

- **Irreducible polynomials cannot be factorized**
  - counter-example: $x^2+1 = (x+1)^2 \bmod 2$
- **Examples:**
  - w=2: x^2+x+1
  - w=4: x^4+x+1
  - w=8: $x^8+x^4+x^3+x^2+1$
  - w=16: $x^{16}+x^{12}+x^3+x+1$
  - w=32: $x^{32}+x^{22}+x^2+x+1$
  - w=64: $x^{64}+x^4+x^3+x+1$

Fast Multiplication

- **Powers laws**
  - Consider: $\{2^0, 2^1, 2^2, \ldots\}$
  - $= \{x^0, x^1, x^2, x^3, \ldots$
  - $=$ exp(0), exp(1), ...
- **exp(x+y) = exp(x) exp(y)**
- **Inverse: log(exp(x)) = x**
  - $\log(x \cdot y) = \log(x) + \log(y)$
- **x y = exp(log(x) + log(y))**
  - Warning: integer addition!!!
- **Use tables to compute exponential and logarithm function**

Example: GF(16)

$q(x)= x^4+x+1$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
exp(x)	1	x	x^2	x^3	$1+x$	$x+x^2$	x^2+x^3	$1+x+x^3$	$1+x^2$	$x+x^3$	$1+x+x^2$	$x+x^2+x^3$	$1+x+x^2+x^3$	$1+x^2+x^3$	$1+x^3$	1
exp(x)	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9	1

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
log(x)	0	1	4	2	8	5	10	3	14	9	7	6	13	11	12

- $5 \cdot 12 = \exp(\log(5)+\log(12)) = \exp(8+6) = \exp(14) = 9$
- $7 \cdot 9 = \exp(\log(7)+\log(9)) = \exp(10+14) = \exp(24) = \exp(24-15)$
  $= \exp(9) = 10$

Example: Reed Solomon for GF[2^4]

- **Compute carry bits for three hard disks by computing**

$$F = \begin{bmatrix} 1^0 & 2^0 & 3^0 \\ 1^1 & 2^1 & 3^1 \\ 1^2 & 2^2 & 3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 5 \end{bmatrix}$$

- **F D = C**
  - where D is the vector of three data words
  - C is the vector of the three parity words
- **Store D and C on the disks**

Complexity of Reed-Solomon

- **Encoding**
  - Time: O(k n) GF[2^w]-operations for k check words and n disks
- **Modification**
  - like Encoding
- **Decoding**
  - Time: O(n^3) for matrix inversion
- **Ease of implementation**
  - check disk overhead is minimal
  - complicated decoding

Cauchy-Reed-Solomon

- **An XOR-Based Erasure-Resilient Coding Scheme, Blömer, Kalfane, Karp, Karpinski, Luby, Zuckerman, 1995**

Definition 5.1 *Let F be a field and let $\{x_1, \ldots, x_m\}$, $\{y_1, \ldots, y_n\}$ be two sets of elements in F such that*

(i) $\forall i \in \{1, \ldots, m\} \forall j \in \{1, \ldots, n\}: \quad x_i + y_j \neq 0.$

(ii) $\forall i, j \in \{1, \ldots, m\}, i \neq j: \quad x_i \neq x_j$ *and* $\forall i, j \in \{1, \ldots, n\}, i \neq j: \quad y_i \neq y_j.$

The matrix

$$\begin{bmatrix} \frac{1}{x_1+y_1} & \frac{1}{x_1+y_2} & \cdots & \frac{1}{x_1+y_n} \\ \frac{1}{x_2+y_1} & \frac{1}{x_2+y_2} & \cdots & \frac{1}{x_2+y_n} \\ & & \ddots & \\ \frac{1}{x_{m-1}+y_1} & \frac{1}{x_{m-1}+y_2} & \cdots & \frac{1}{x_{m-1}+y_n} \\ \frac{1}{x_m+y_1} & \frac{1}{x_m+y_2} & \cdots & \frac{1}{x_m+y_n} \end{bmatrix}$$

is called a Cauchy matrix over F.

Theorem 5.3 *The inverse of an $(n \times n)$-Cauchy matrix over a field F can be computed using $\mathcal{O}(n^2)$ arithmetic operations in F.*

Complexity of Cauchy-Reed-Solomon

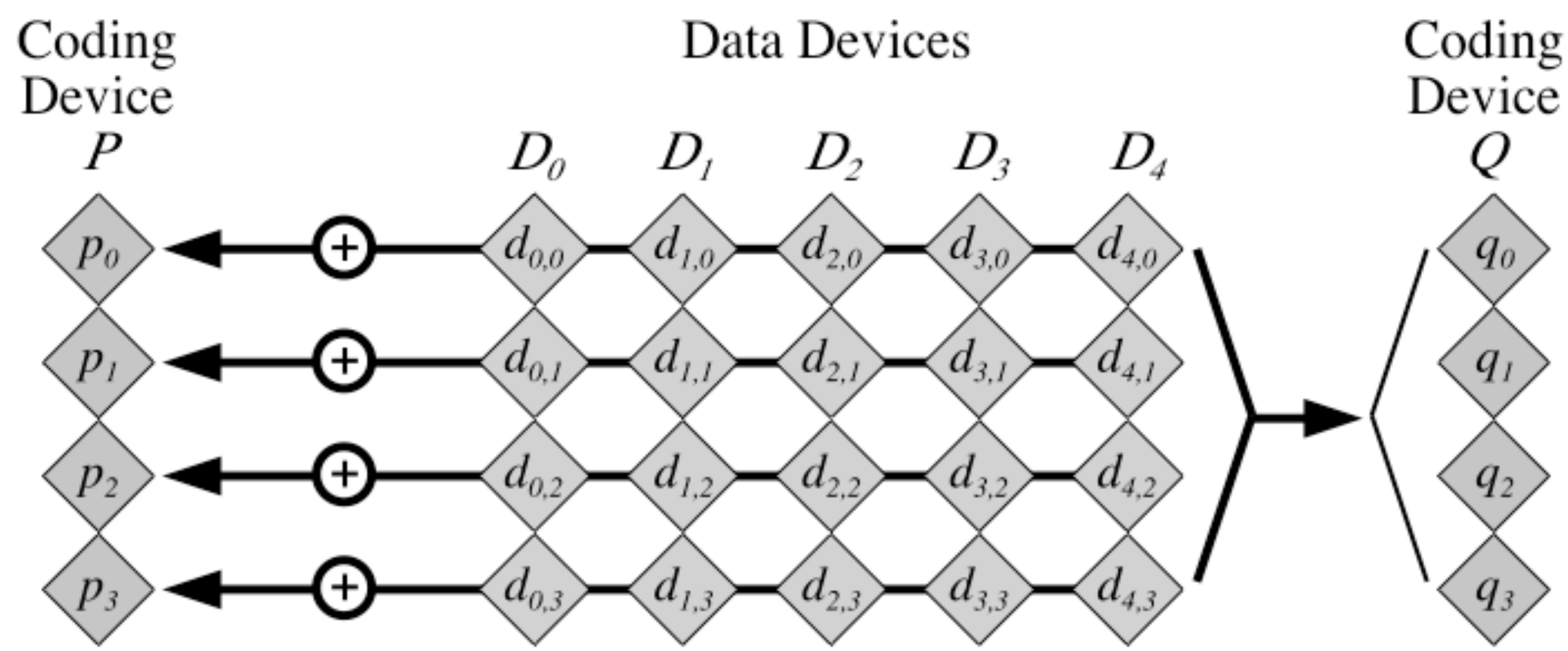
- **Encoding**
  - Time: O(k n) GF[2^w]-operations for k check words and n disks
- **Modification**
  - like Encoding
- **Decoding**
  - Time: O(n^2) for matrix inversion
- **Ease of implementation**
  - check disk overhead is minimal
  - less complicated decoding, still not transparent

RAID 6 Encodings

Parity Arrays

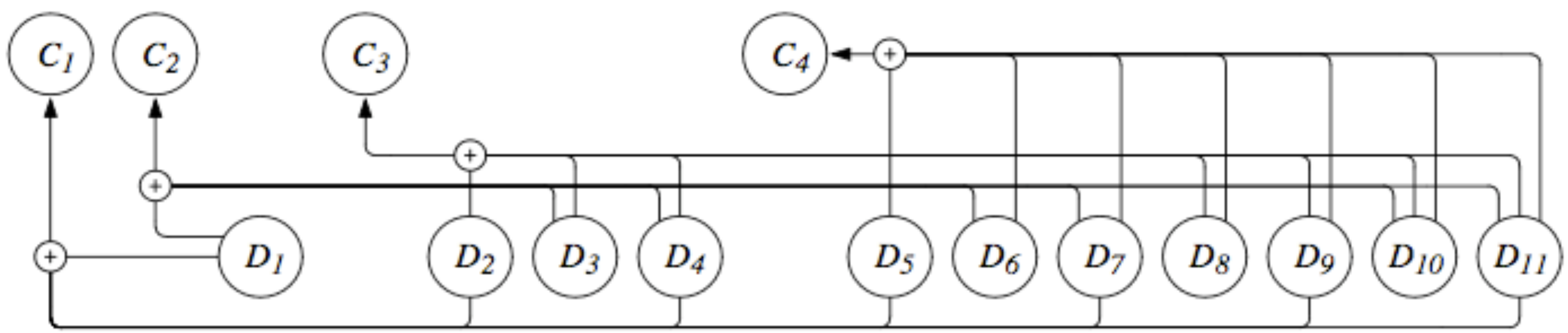
Parity Arrays

- **Uses Parity of data bits**
- **Each check bit collects different subset of data bits**
- **Examples**
  - Evenodd
  - RDP



Hamming Code

- **Use adapted version of Hamming code to compute check bits**
- **Problem: not flexible encoding for various number of disks or check codes**

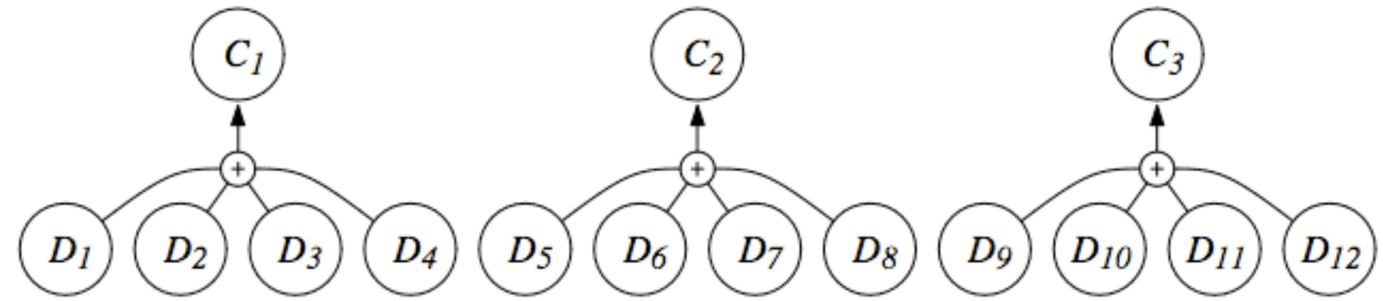


Hamming code, $n = 11, m = 4$

The RAID-6 Liberation Codes
James S. Plank

One-Dimensional Parity

- **Organize data bits as n/m groups**
  - compute parity for each group
- **Results in m check bits**
- **Fast and simple computation for**
  - Coding, Decoding, Modification
- **Problem**
  - tolerates not all combinations of failures
  - unsafe solution for combined failure of check disk and data disk

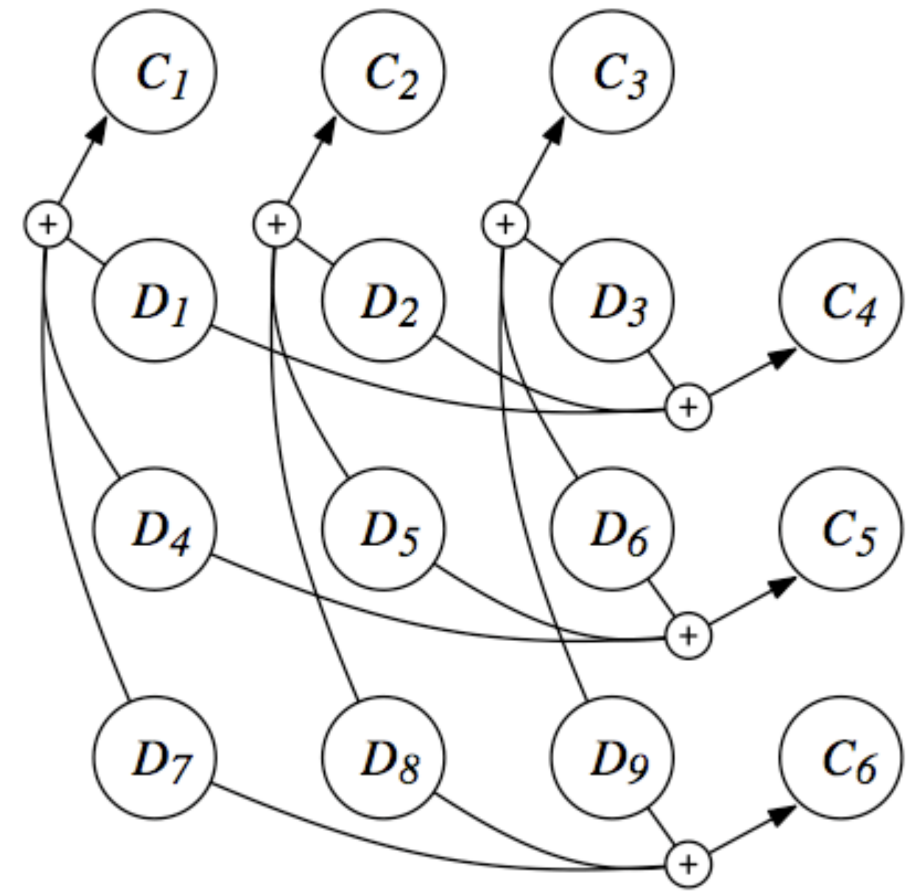


One-dimensional parity, $n = 12, m = 3$

A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank , 1999

Two-Dimensional Parity

- **Organize data disks as a k*k-square**
  - compute k parities for all rows
  - compute k parities for all columns
- **Results in 2k check bits**
- **Fast computation for**
  - Coding, Decoding, Modification
- **Safety**
  - tolerate only all combinations for two failures
  - tolerates not all combinations for three failures
- **Problem**
  - large number of hard disks
  - check disk overhead

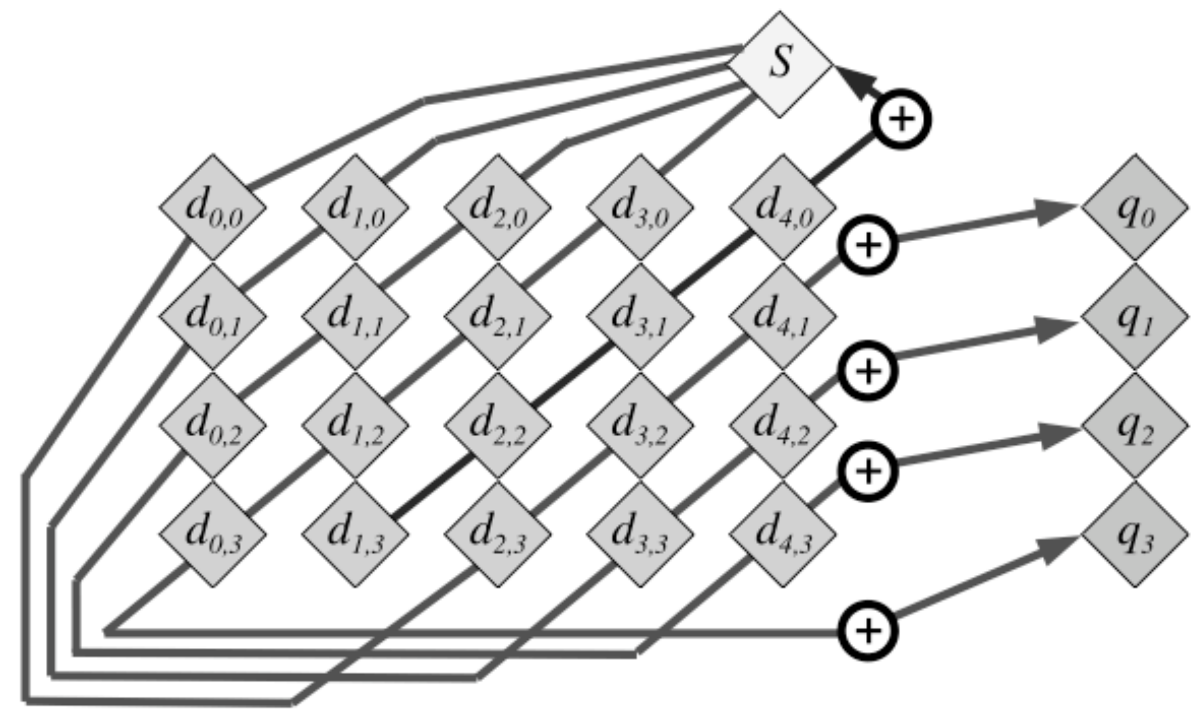


Two-dimensional parity, $n = 9, m = 6$

A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank , 1999

EVENODD-Encoding

- **Computes exactly two check words**
- **P = parity check word**
- **Q = parity over the diagonal elements**
- **Fast Encoding**
- **Decoding**
  - $O(n^2)$ time for n disks and n data bits
- **Optimal check disk overhead**
- **Generalized versions**
  - STAR code (Huang, Xu, FAST'05)
  - Feng, Deng, Bao, Shen, 2005



The RAID-6 Liberation Codes
James S. Plank

EVENODD

$$S = \sum_{t=1}^{n-1} d_{n-1-t,t}$$

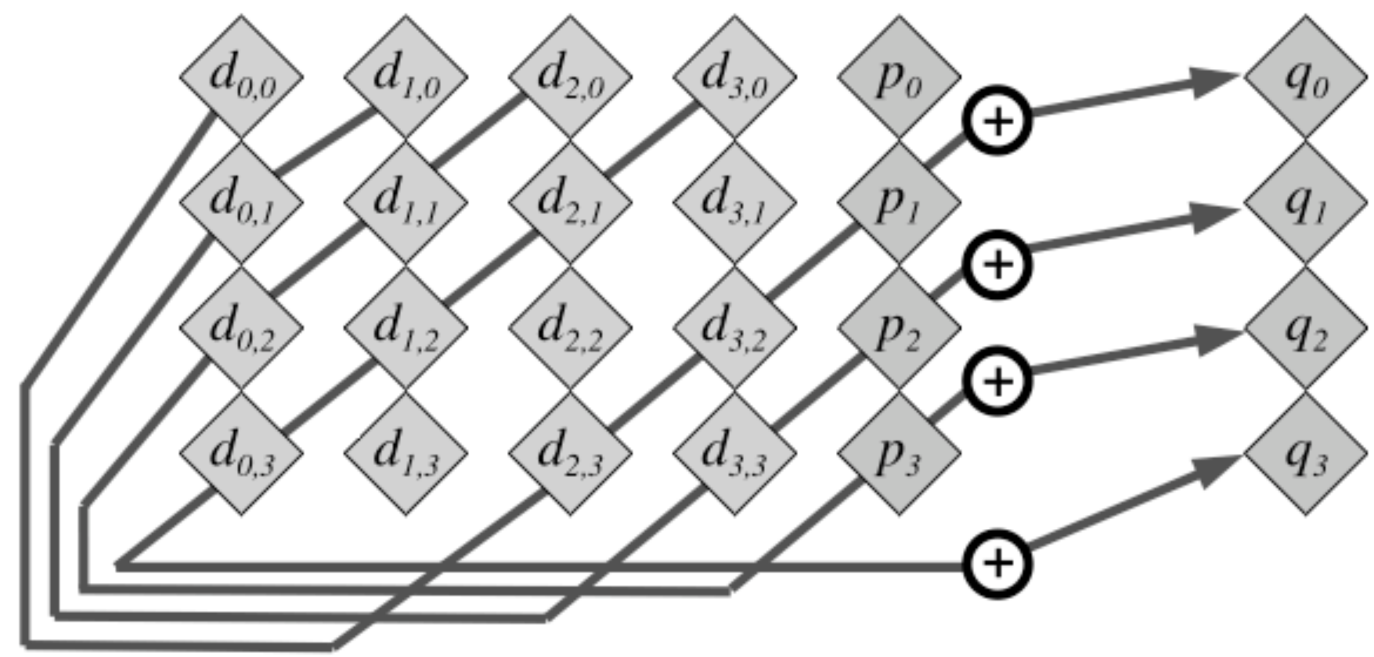
$$q_0 = S + d_{0,0} + \sum_{t=2}^{n-1} d_{t,n-t}$$

$$i \le n-2: \quad q_i = S + \sum_{t=0}^{i} d_{t,i-t} + \sum_{t=2+i}^{n-1} d_{t,n-t+i}$$

$$q_{n-1} = S + \sum_{t=0}^{n-2} d_{t,n-2-t}$$

RDP Coding

- **Row Diagonal Parity**
  - improved version of EVENODD
- **Two check words**
  - Parity over words
  - Use diagonal parities
- **Easier code**
- **Creates only two check words**



The RAID-6 Liberation Codes
James S. Plank

Liberation Codes

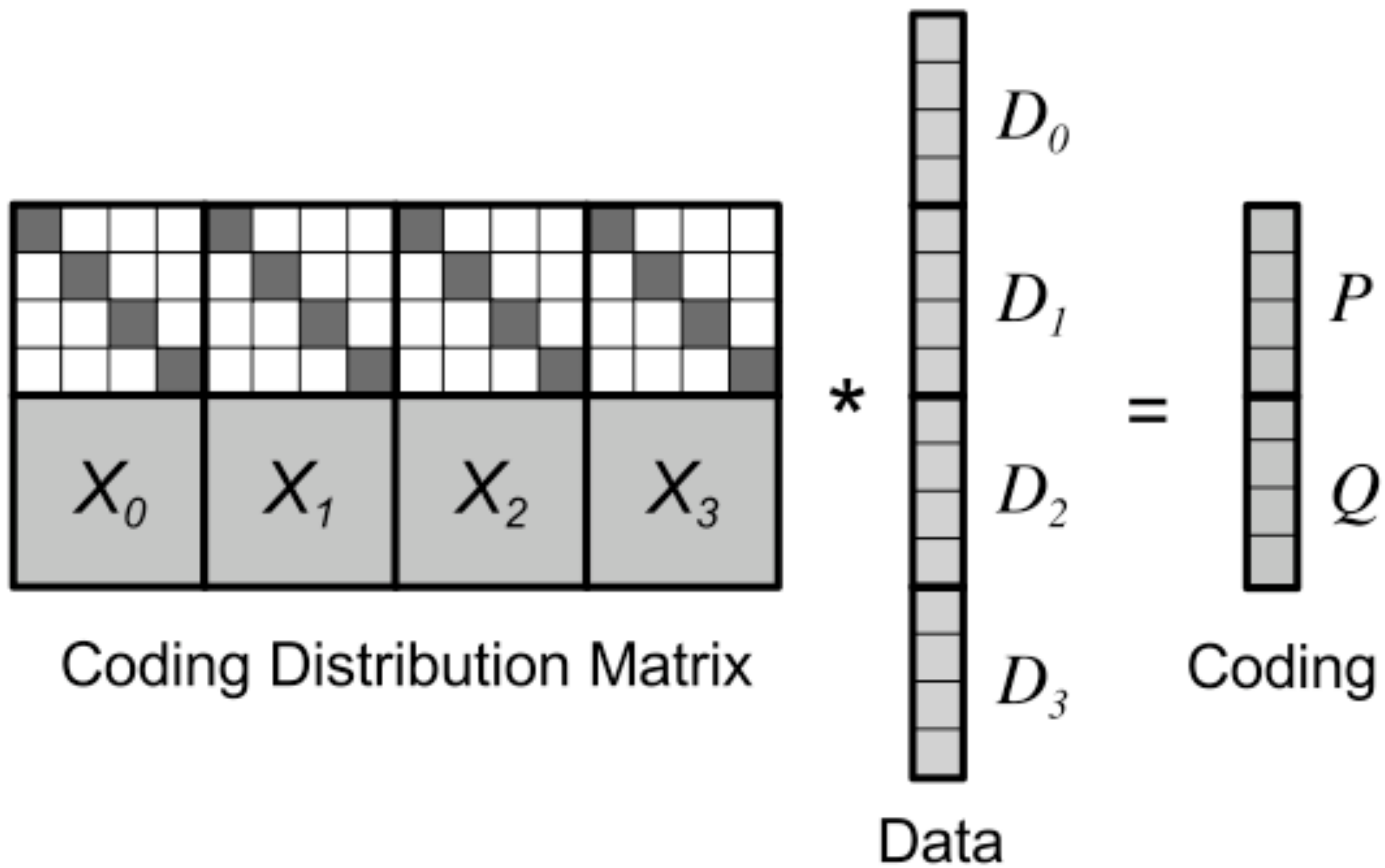
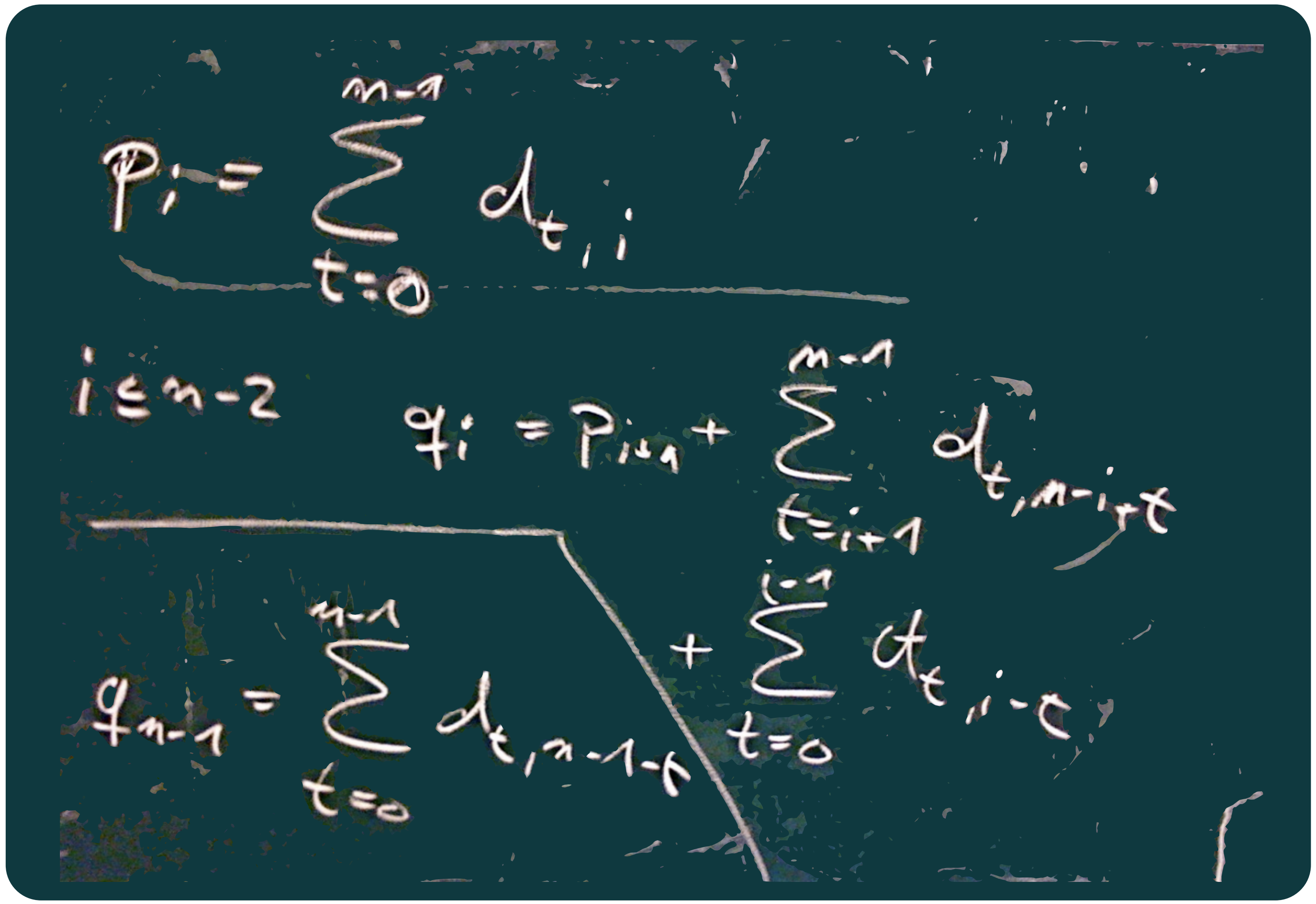


Figure 7: Bit matrix representation of RAID-6 coding when $k = 4$ and $w = 4$.

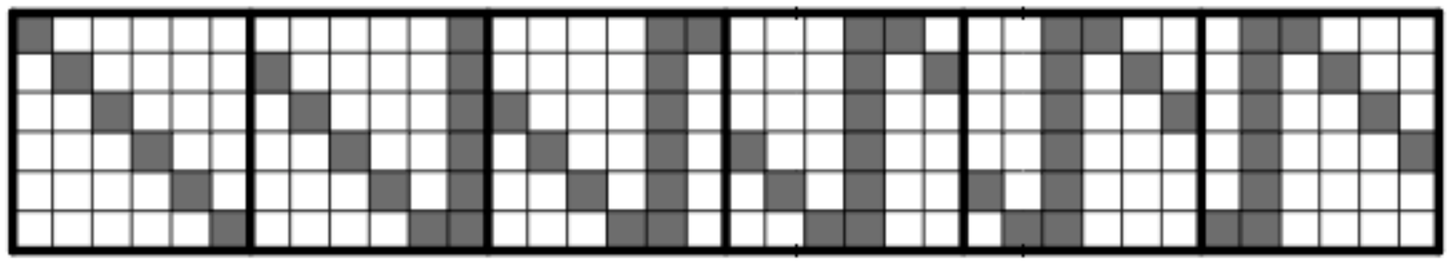
The RAID-6 Liberation Codes
James S. Plank

RDP

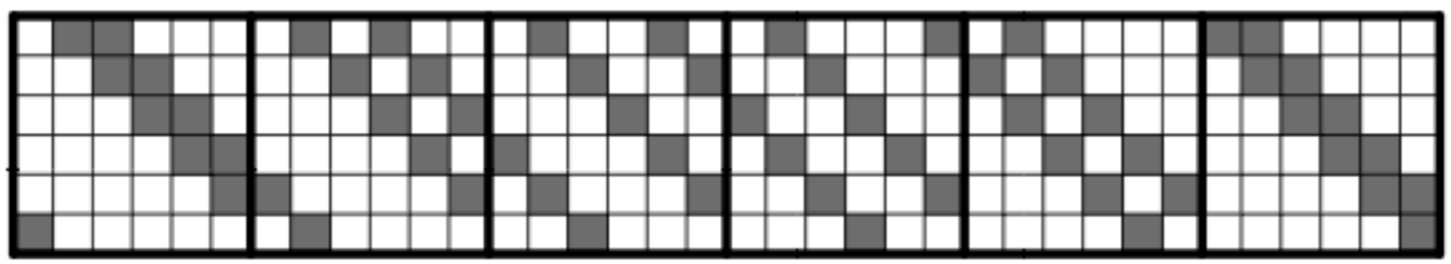


The RAID-6 Liberation Codes
James S. Plank

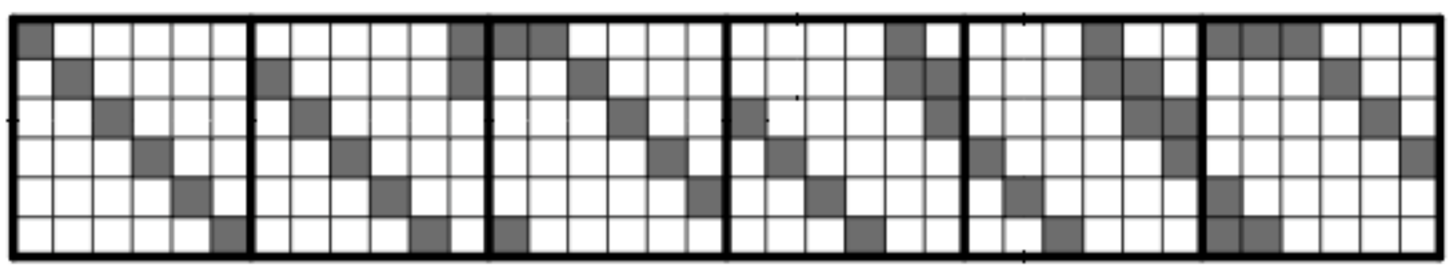
Liberation Codes



(a) EVENODD.



(b) RDP.

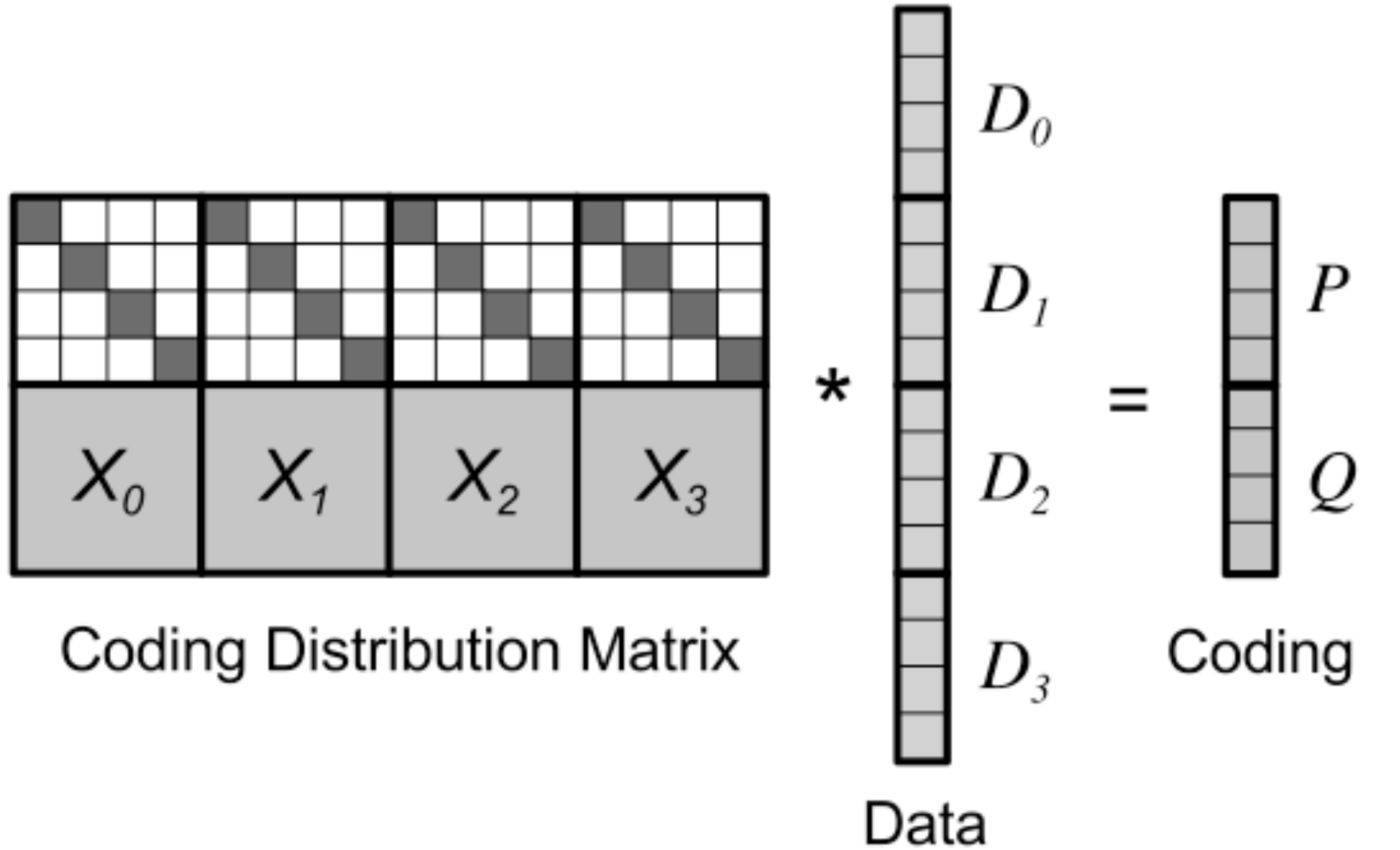


(c) Cauchy Reed-Solomon coding.

Figure 8: The X_i matrices defining the BDM's for various RAID-6 coding techniques, $k = 6$ and $w = 6$.

The RAID-6 Liberation Codes
James S. Plank

Liberation Codes



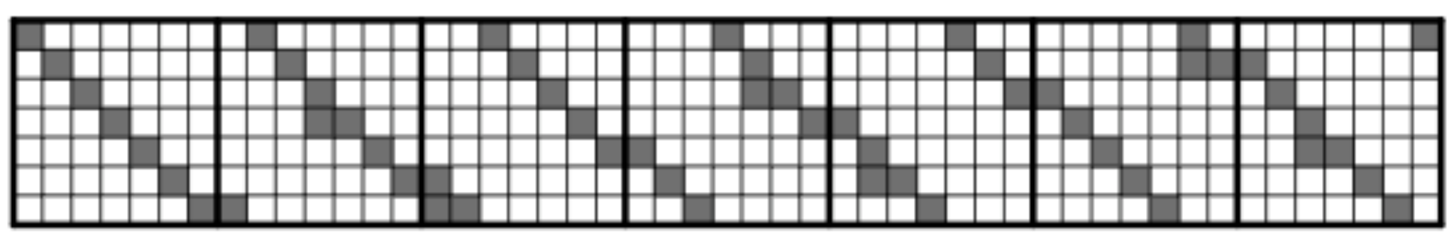
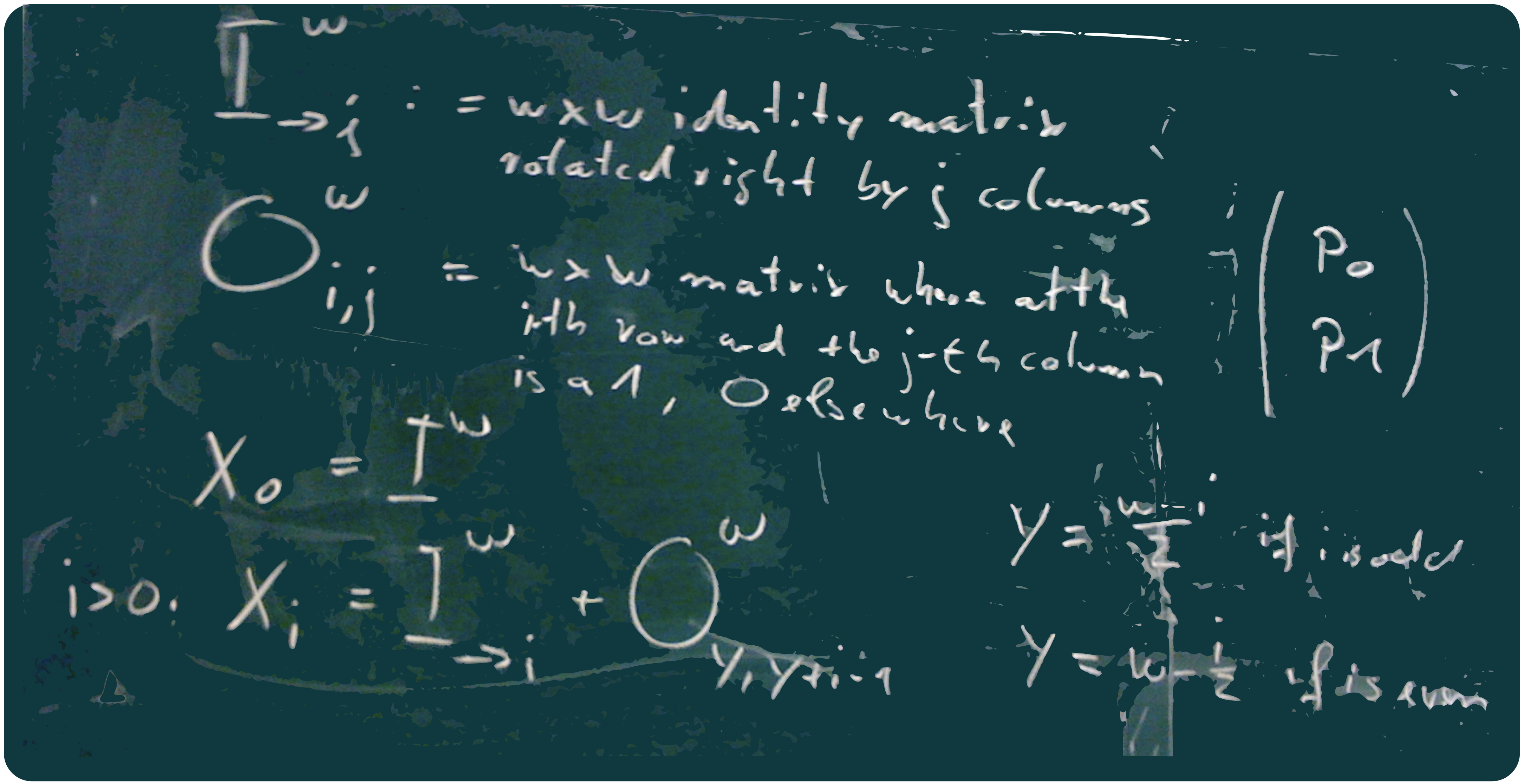


Figure 9: The X_i matrices for the Liberation Code when $k = 7$ and $w = 7$.

The RAID-6 Liberation Codes
James S. Plank

Liberation Codes



The RAID-6 Liberation Codes
James S. Plank

Liberation Codes

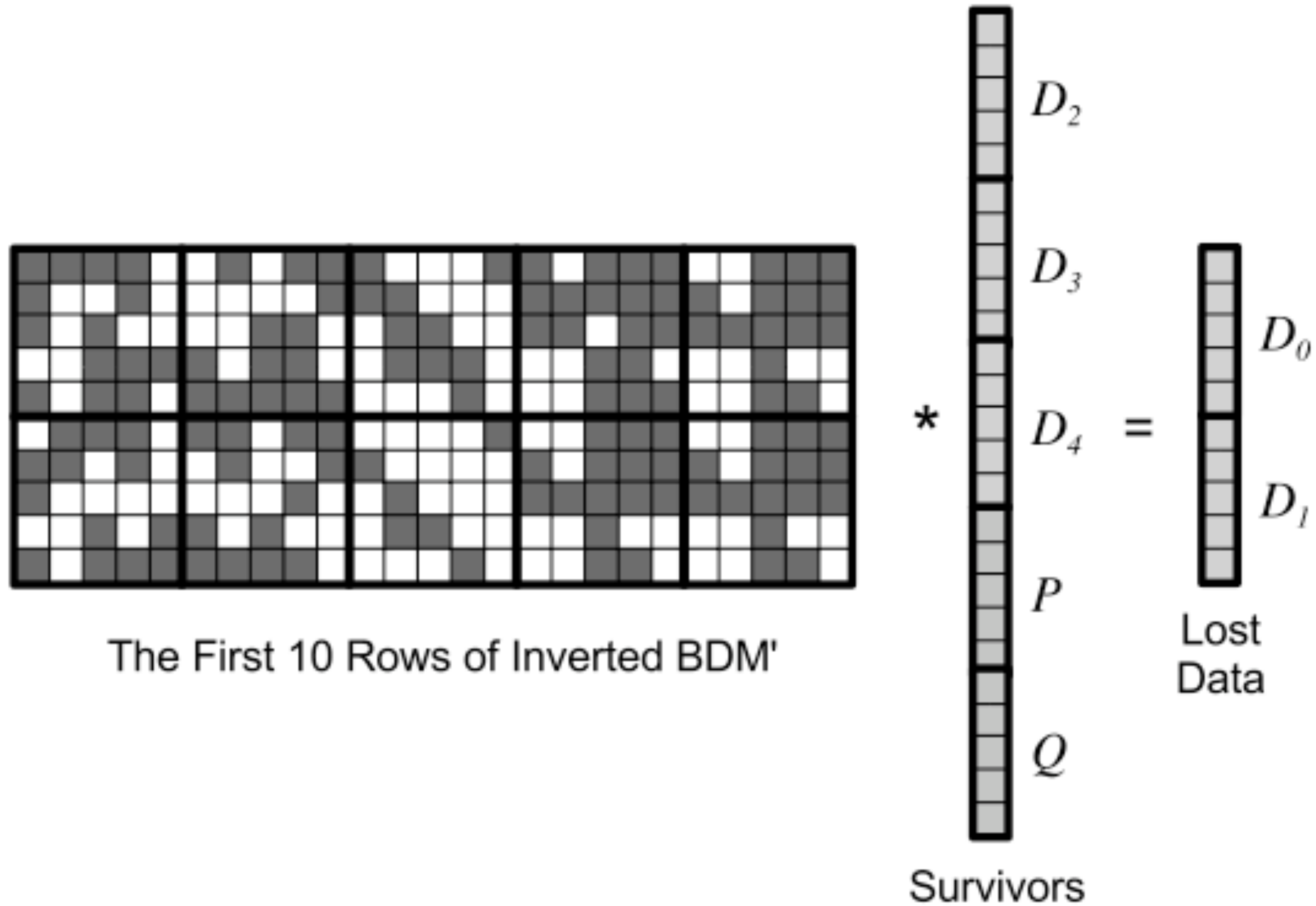


Figure 10: Decoding D_0 and D_1 from the Liberation Codes when $k = 5$ and $w = 5$.

The RAID-6 Liberation Codes
James S. Plank

Performance

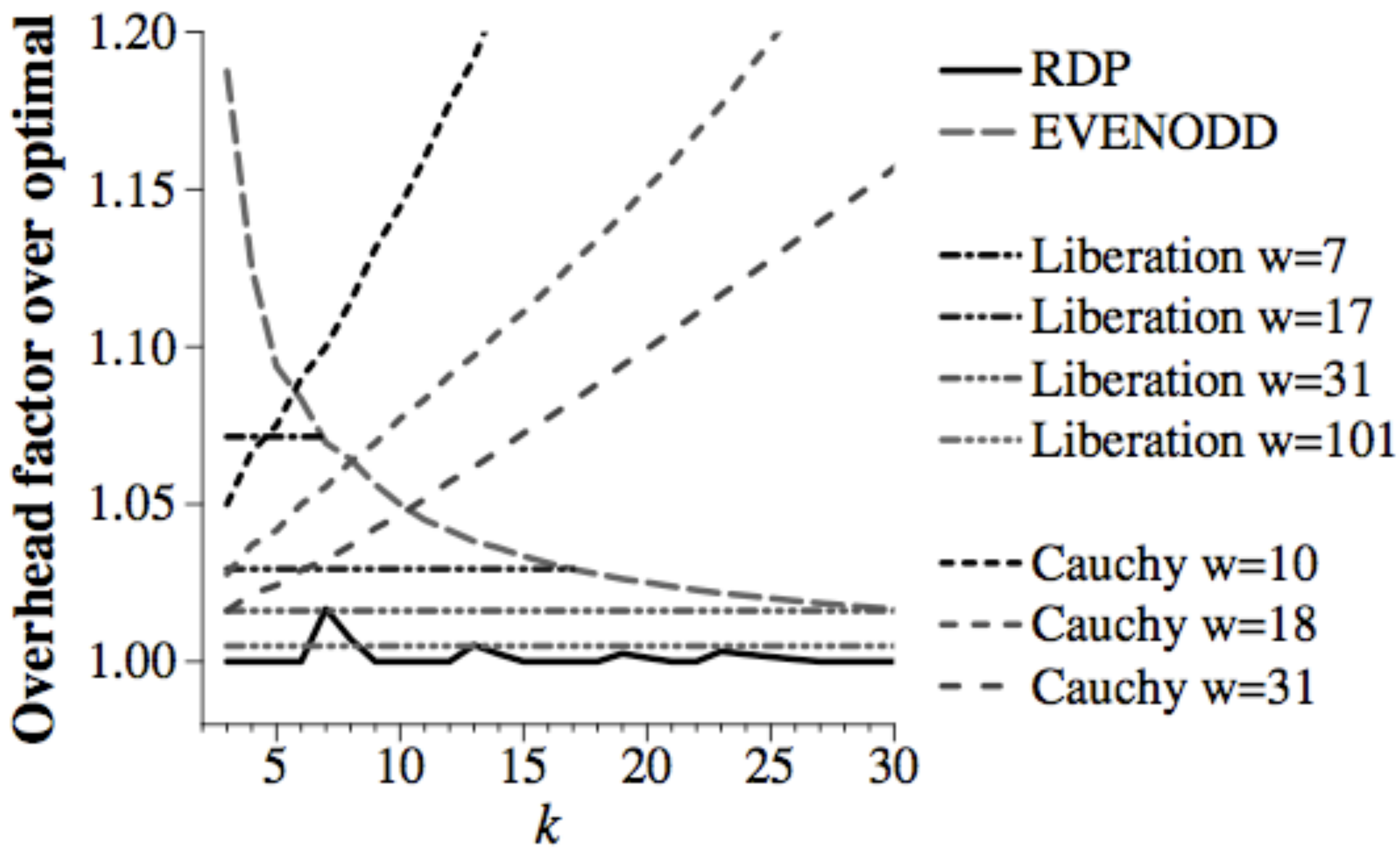


Figure 11: Encoding performance of various XOR-based RAID-6 techniques. Optimal encoding is $k-1$ XORs per coding word.

The RAID-6 Liberation Codes
James S. Plank

Performance

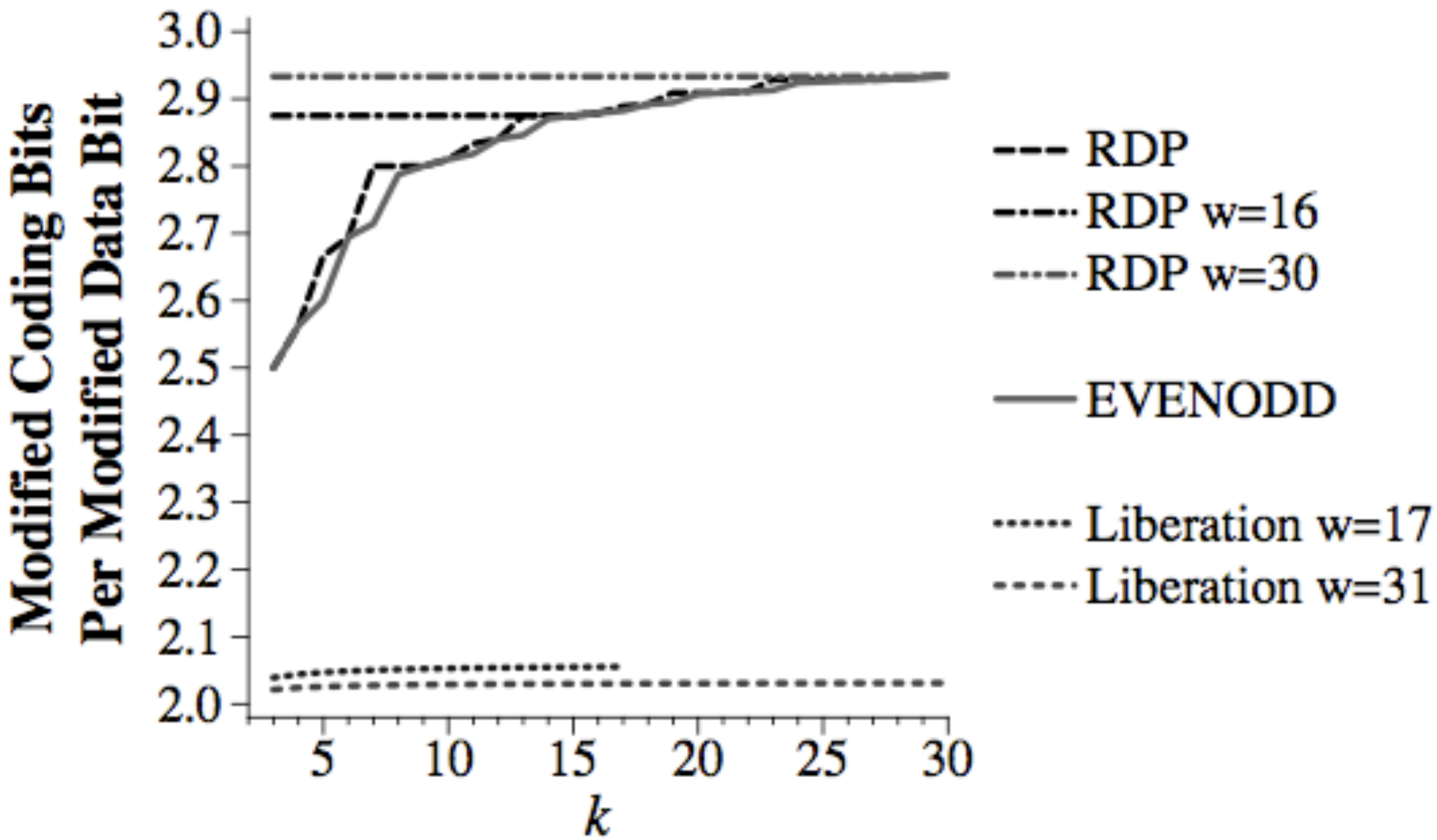


Figure 13: Modification performance of RDP, EVEN-ODD and Liberation codes.

The RAID-6 Liberation Codes
James S. Plank

Performance

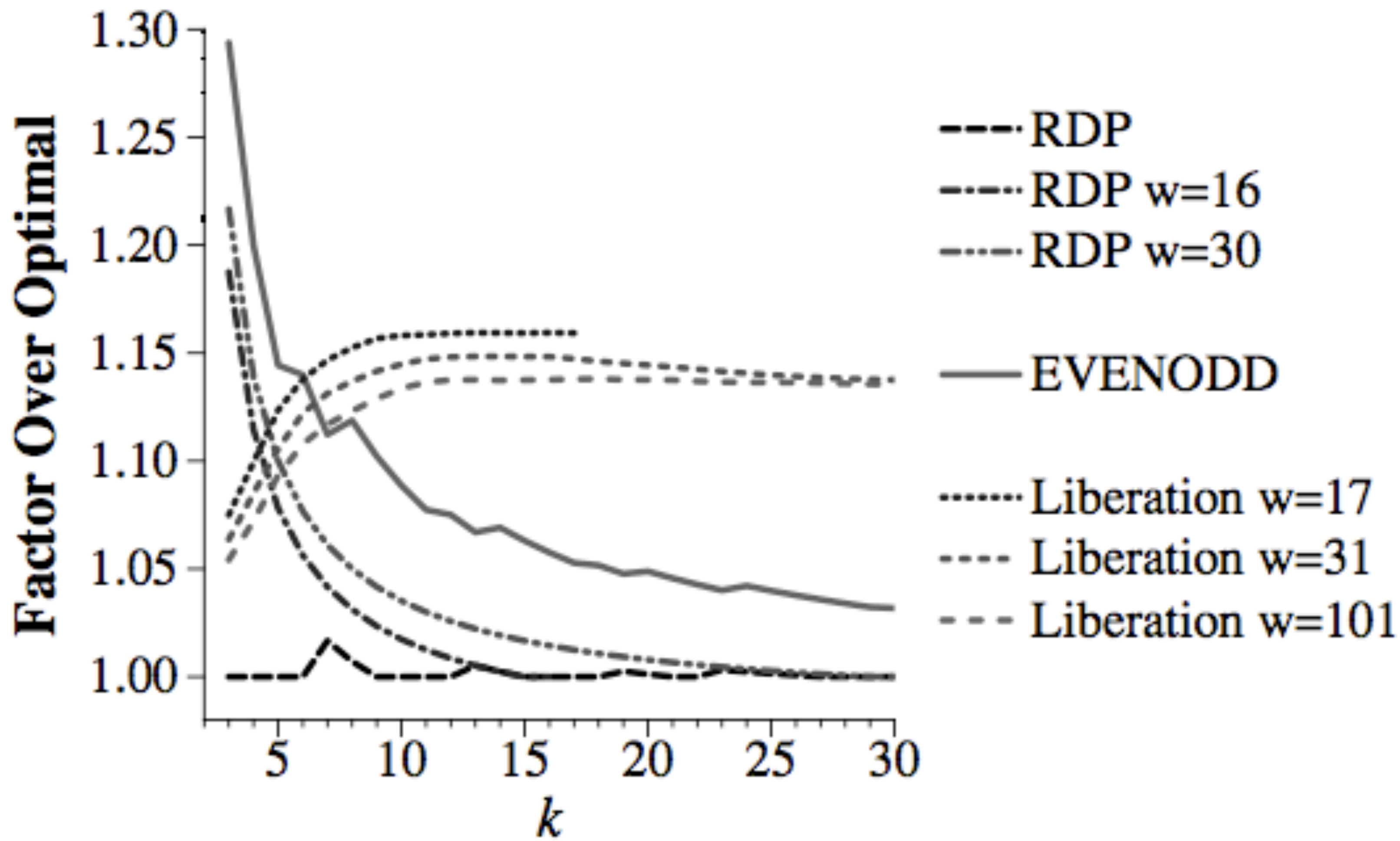


Figure 14: Decoding performance of RDP, EVENODD and Liberation codes.

The RAID-6 Liberation Codes
James S. Plank

ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Distributed Storage Networks and Computer Forensics

5 Raid-6 Encoding

Christian Schindelhauer

University of Freiburg
Technical Faculty
Computer Networks and Telematics
Winter Semester 2011/12

CoNe Freiburg

IIF
INSTITUT FÜR INFORMATIK FREIBURG