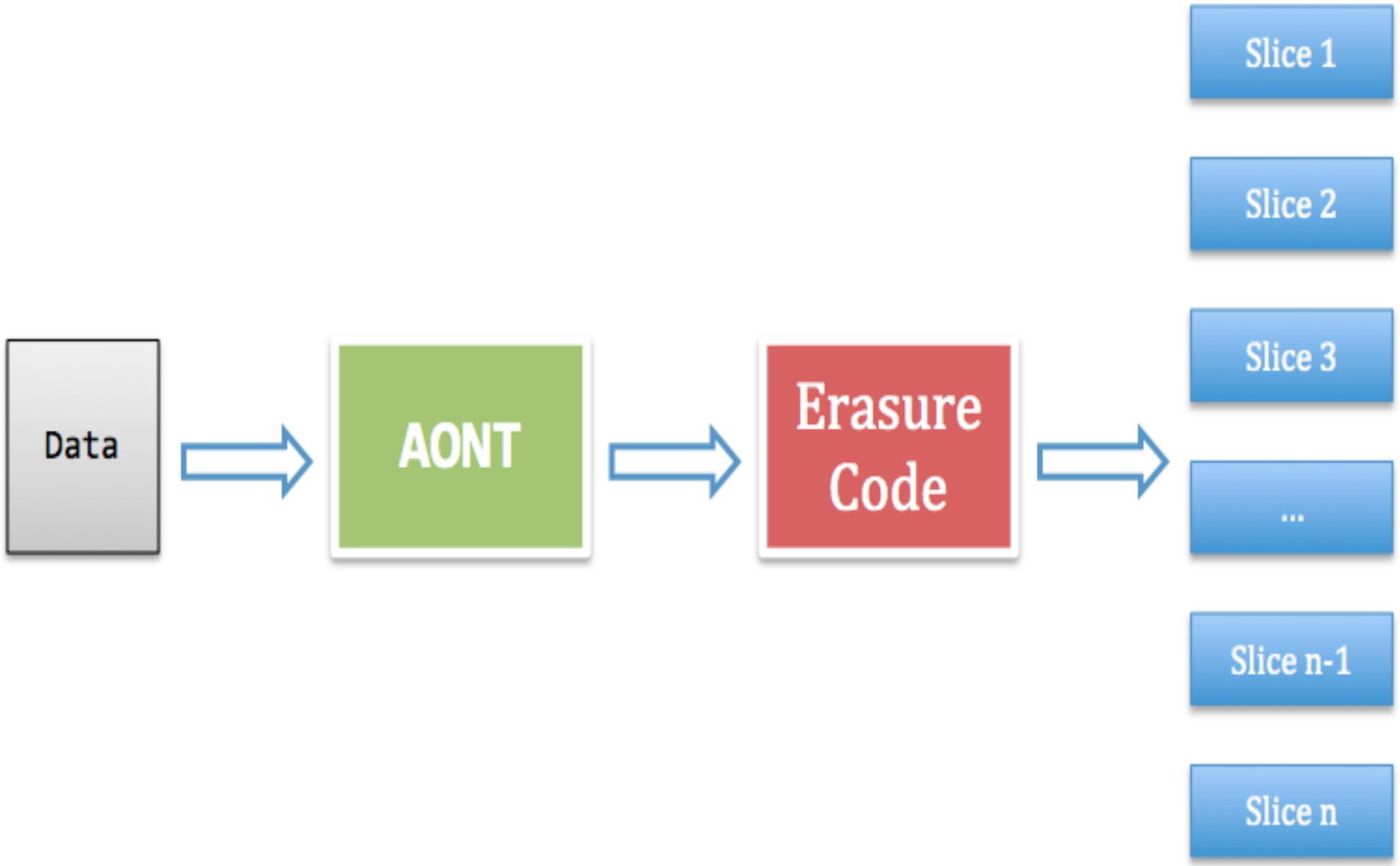
Erasure coding and AONT algorithm selection for Secure Distributed Storage

Alem Abreha
Sowmya Shetty

Secure Distributed Storage



AONT(All-Or-Nothing Transform)

- unkeyed transformation ϕ mapping a sequence of input blocks (x1,x2,x3 … xs) to a sequence of output blocks (y1,y2,y3 … ys‘)
- AONT Properties:
  i. given all (y1,y2,y3 … ys‘) it is easy to compute (x1,x2,x3 … xs)
  ii. if any one of the yi is missing then it is computationally infeasible to obtain any information about any xi.
- 3 - Algorithms:
  - Rivest's AONT : Ronald L. Rivest in 1997
  - Boyko's AONT : Victor Boyko in 1999
  - Stinson's AONT: D.R. Stinson in 2001

Rivest's AONT

- **Input:**

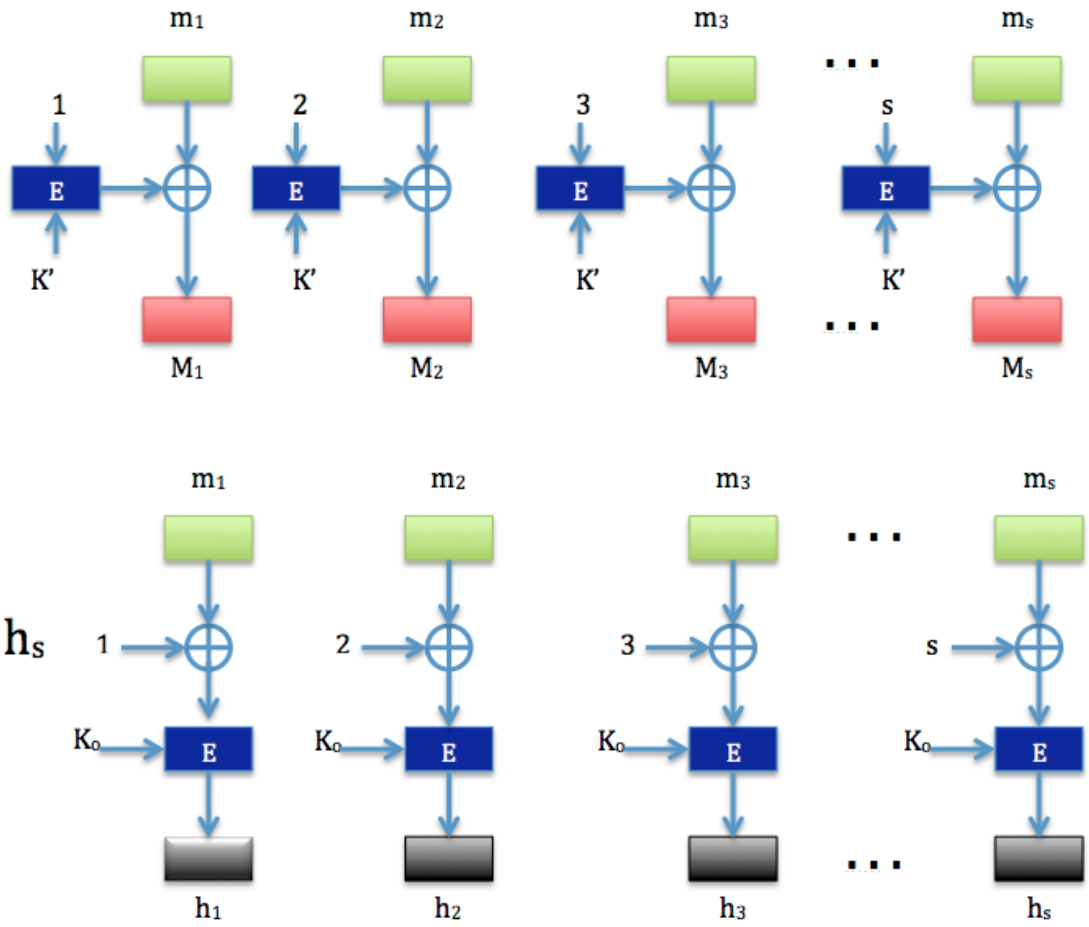
  $m_1, m_2, m_3 \ldots m_s$
- Random key K'
- Fixed, public key Ko
- **Output:**

  $M_1, M_2, M_3 \ldots M_s, M_{s'}$
- **Where:**

  $s' = s + 1$

  and

$$M_{s'} = K' \oplus h_1 \oplus h_2 \oplus \ldots \oplus h_s$$



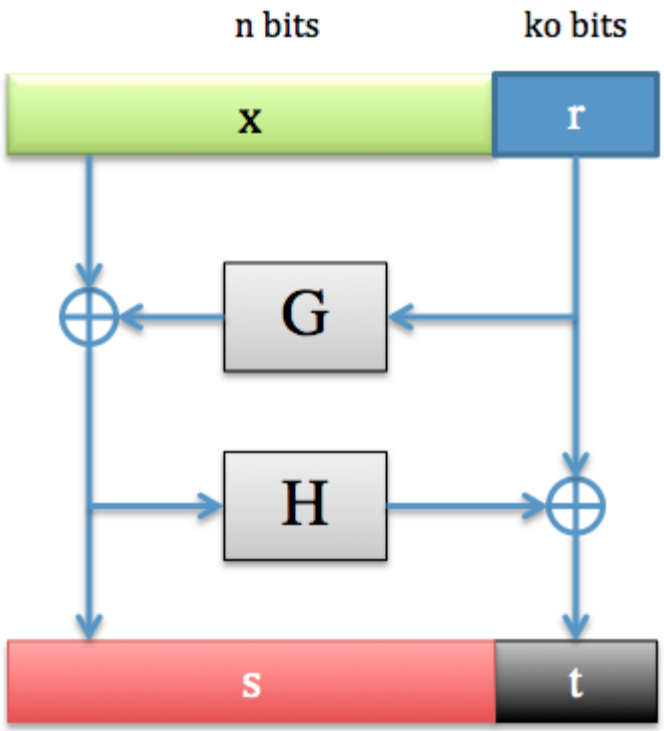
Boyko's AONT

Based on OAEP(Optimal Asymmetric Encryption Padding)

- **Input :** message x , random string r

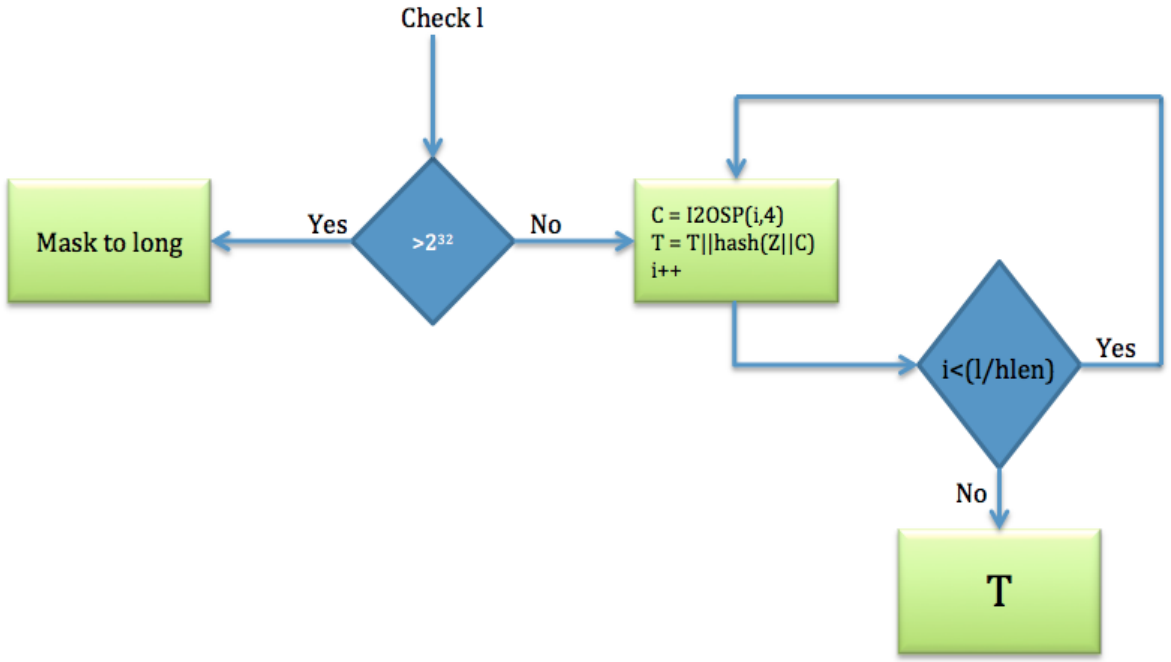
  $x = m_1||m_2||m_3|| \ldots||m_s$
- **Output:** s||t
- **Where:** G and H are random oracles
- |s| = n and |t| = ko

$$\mathrm{OAEP}^{G,H}(x,r) = x \oplus G(r) \,\|\, r \oplus H(x \oplus G(r))$$



MGF1 modeling of G and H

MGF1(Z,l)



- **Where :**
  hash = hash function
  hlen = hash function output length (in octets)
  Z = seed used for mask generation (octet string)
  l = intended length of mask in octets (max 2^32)
  T= initialized as empty octet string
  i = for loop counter starting from 0
  I2OSP = Integer-to-Octet-String

Stinson's AONT

- **Input :** x
- **Output :** y
- **Where :**
  - M is an invertible s by s matrix, and no entry of M is equal to 0
  - $y = xM^{-1}$, $x = yM$

Special case of M for speed optimization :

1. For $1 \le i \le s-1$, compute $y_i = x_i + x_s$
2. Compute $x_s = x_1 + \cdots + x_{s-1} + \lambda x_s$

$AONT^{-1}$:

1. Compute $y_s = \gamma(x_1 + \cdots + x_{s-1} - y_s)$
2. For $1 \le i \le s-1$, compute $x_i = y_i - x_s$

$$M = \begin{pmatrix} 1-\gamma & -\gamma & -\gamma & \dots & -\gamma & \gamma \\ -\gamma & 1-\gamma & -\gamma & \dots & -\gamma & \gamma \\ -\gamma & -\gamma & 1-\gamma & \dots & -\gamma & \gamma \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\gamma & -\gamma & -\gamma & \dots & 1-\gamma & \gamma \\ \gamma & \gamma & \gamma & \dots & \gamma & -\gamma \end{pmatrix}$$

$$M^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & \lambda \end{pmatrix}$$

AONT Algorithm comparisons

Rivest's AONT:

- Very easy to implement using standard encryption and hash functions
- Open-source C libraries are available
- Encoding speed bench marked on 4-core Intel Xeon W3530 at 2.80 GHz with 6GB of memory at 1066 MHz running Linux kernel 2.6.32 platform
- High speed AONT : combining RC4-128 and MD5, we obtain encoding speed of 237.99MB/s
- High security AONT: combining AES-256 and SHA-256, we obtain encoding speed of 75.60MB/s
- Down sides of Rivest's AONT:
  - Encryption is done twice during encoding, hence slower encoding speed

AONT Algorithm comparisons

Boyko's AONT:

- based on ideal random oracles, need to modify OAEP by replacing G or H by a deterministic functions
- replacing G and H by deterministic variable length hash functions using mask generation function
- MGF1 is a mask generation function based on a hash function and can be used to implement variable length output hash functions.
- MGF1 libraries:
  - java : available at http://javadoc.iaik.tugraz.at/iaik_jce/3.13/iaik/pkcs/pkcs1/MGF1.html
  - C++: from Botan available at http://fossies.org/dox/Botan-1.10.3/mgf1_8cpp_source.html
  - C: from OpenSSL implementation, http://www.openssl.org/source/
- Can achieve faster encoding speed than Rivest's AONT

AONT Algorithm comparisons

Stinson's AONT:

- No existing implementation, no Libraries known to be available for implementation
- Ideally adversary can get enough parameters to compute/guess missing cipher blocks using the available compromised ciphertext blocks

Feature	Rivest's AONT	Boyko's AONT	Stinson's AONT
Ease of Implementation	Easy	Medium	Hard
Speed	Low - High	High	N/A
AONT Security	Yes	Yes*	N/A

yes* = provided that G and H are collision resistant.

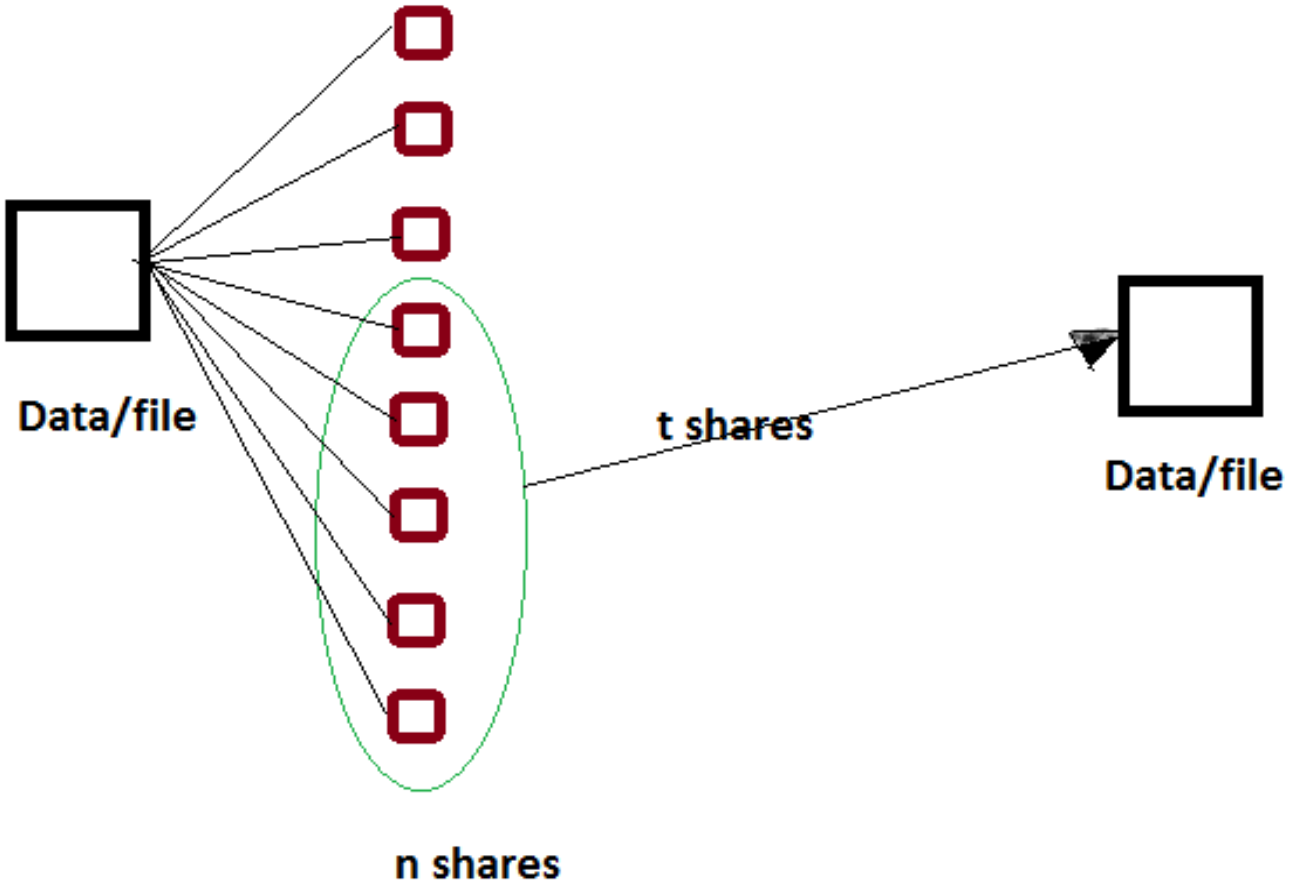
Erasure codes

Method of distributing data 'D' into 'n' shares such that data can be recovered from the available 't' shares(where the remaining (n-t) shares out of reach)

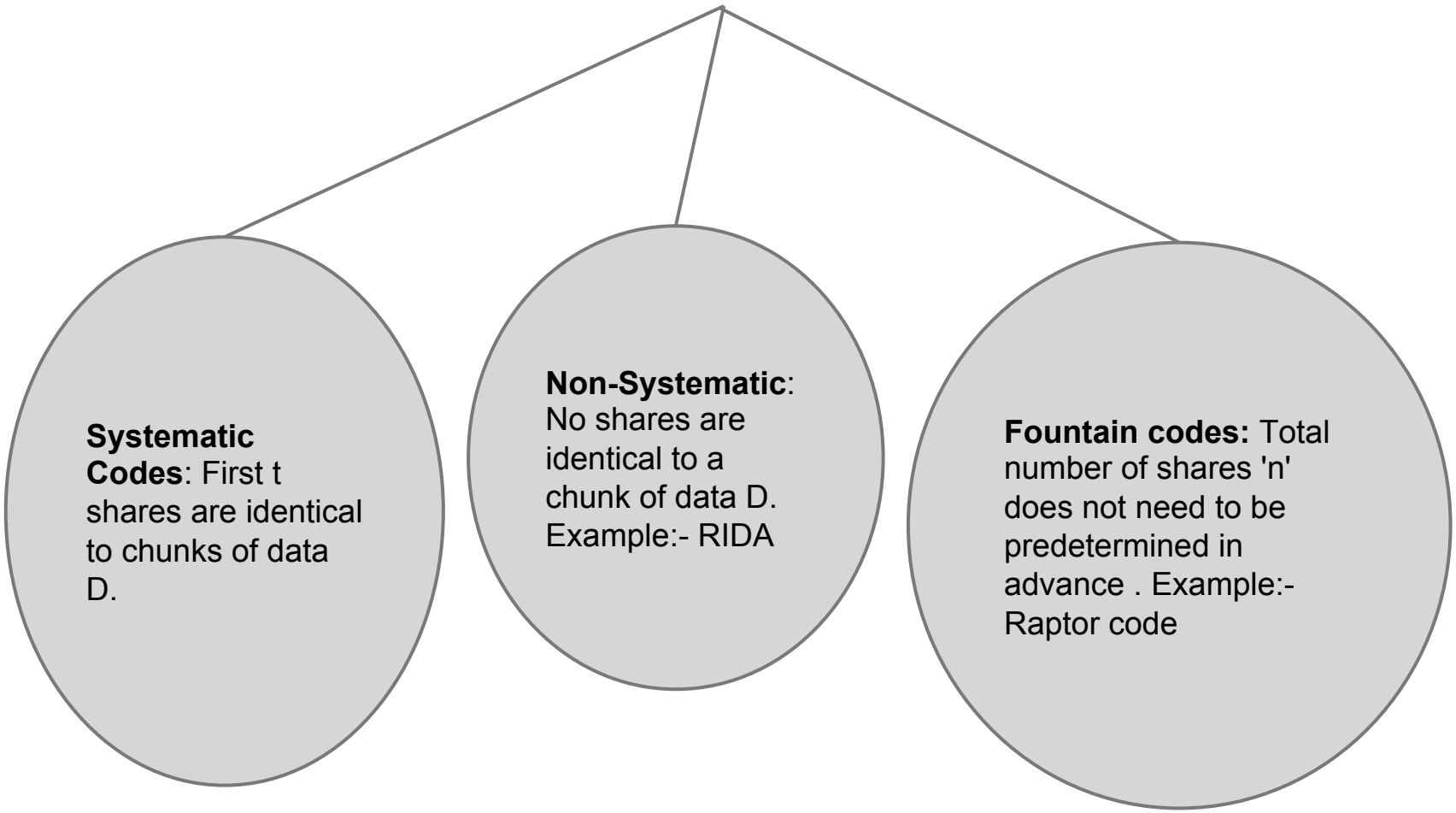
Parameters :

t - threshold(t<n)

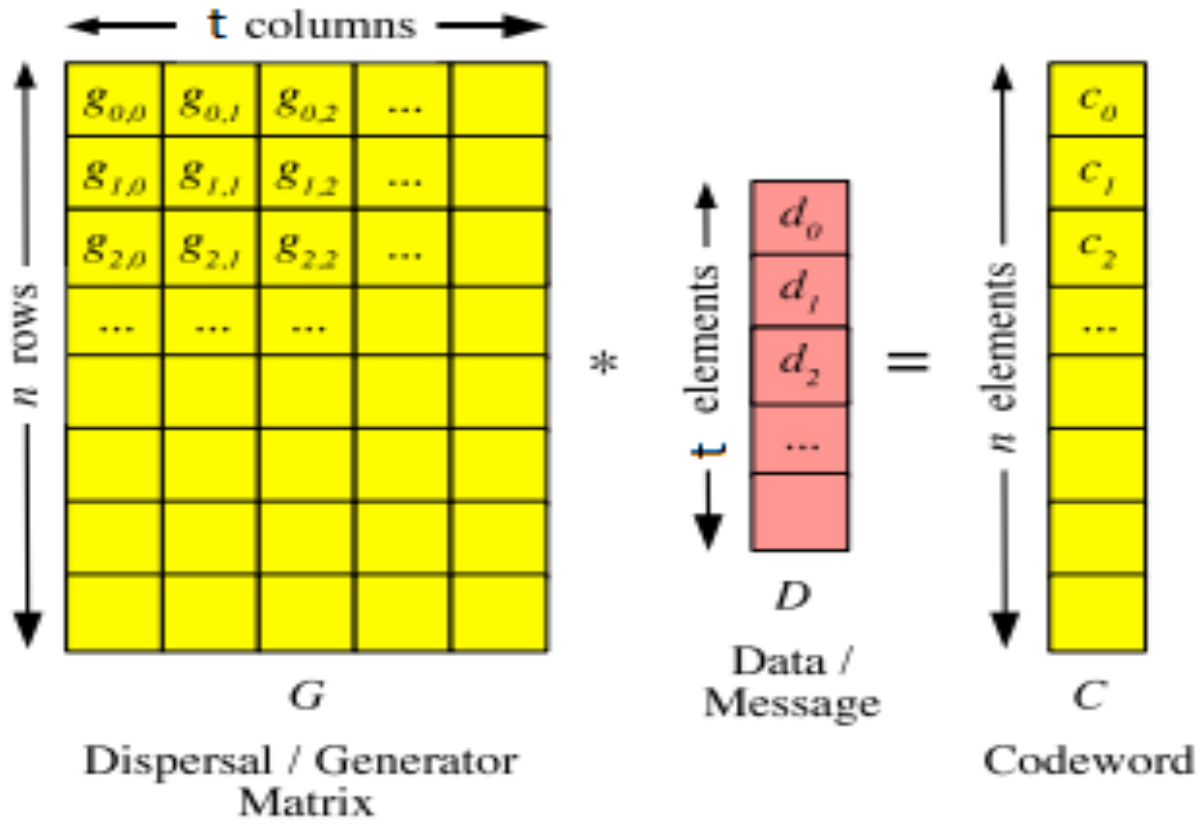
n - number of shares



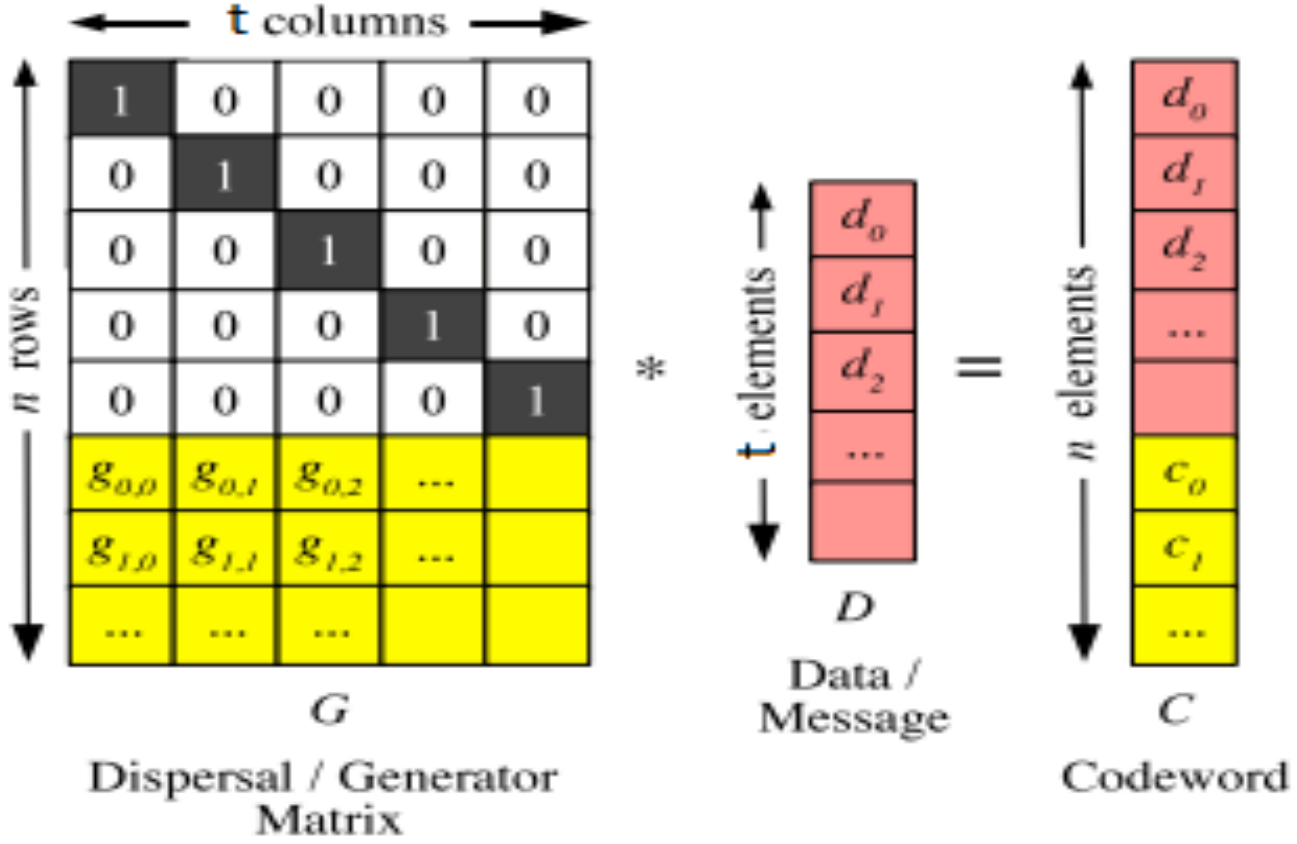
Types of erasure codes



Erasure code

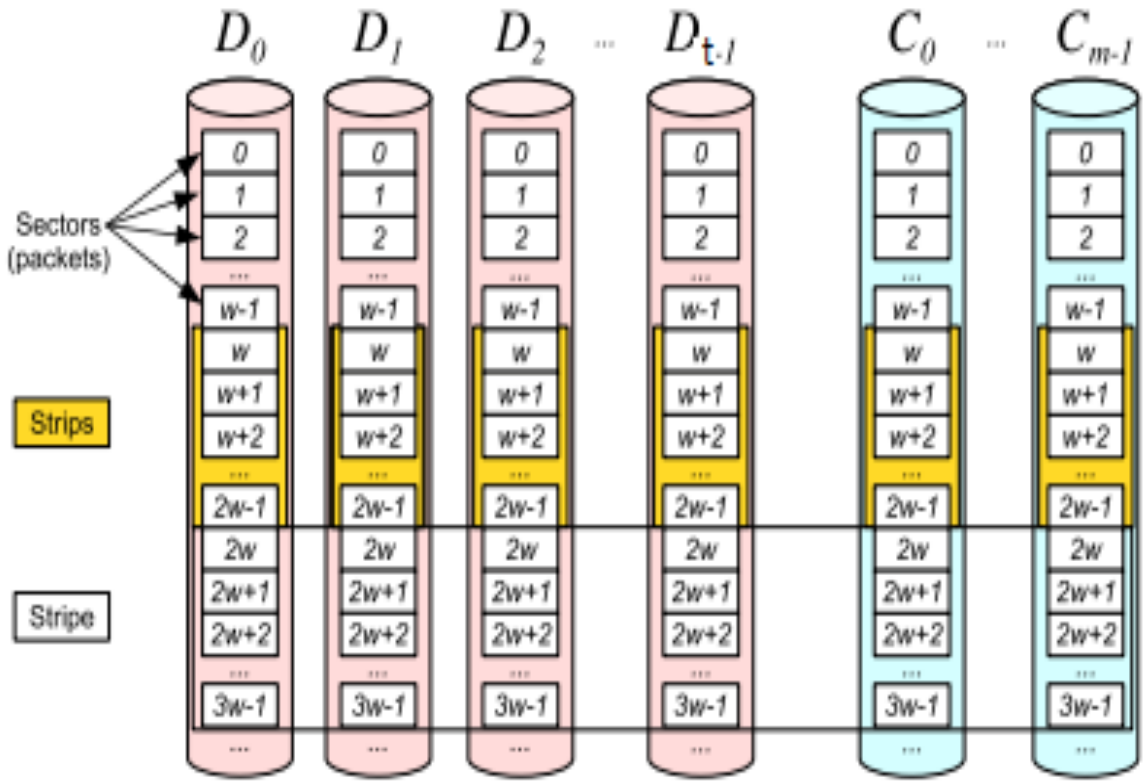


Systematic erasure code



Example: Reed Solomon code, Cauchy Reed Solomon code, EvenOdd code etc

Erasure code..



Erasure code

*Standard Reed-Solomon Code(RS):

The most general technique for tolerating simultaneous failures with exactly checksum/ coding devices and can take arbitrary values.

*Cauchy Reed-Solomon code(CRS):

Modification are done on RS to improve efficiency

*Rabin's Information dispersal Algorithm:

Michael O. Rabin in his paper “Efficient dispersal of information for security, load balancing, and fault tolerance”

Standard Reed Solomon Code

1.Encoding:

$c_i = F_i(d_1, d_2, \ldots, d_t)$

define F to be the $m \times t$ Vandermonde matrix: $f_{i,j} = j^{i-1}$, and thus the above equation becomes:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & t \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \cdots & t^{m-1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_t \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_t \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}.$$

Generator matrix

2.Decoding

-If there are any ‘t’ devices available, then first any missing data d1.. dt are calculated using Gaussian elimination method.

-Then checksum/coding data are Calculated

Problem :

Multiplication and division are complex and use Galois field, which slows down the operation.

Cauchy Reed Solomon Code

1. Instead of using a Vandermonde matrix, CRS coding employs an t x n Cauchy matrix, over GF (2^w), where n + t ≤ 2w, which improves the performance of matrix inversion for decoding.
2. Eliminate the expensive multiplications of RS codes by converting them to extra XOR operations, so that encoding takes $O(n\log_2(t+n))$ ↄRs per coding block.

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

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

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

The matrix

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

1. Every square matrices of Cauchy matrix is invertible
2. Encoding is a linear function of message and the un-encoded message
3. Decoding is inversely proportional to unencoded message received

Rabin's Information dispersal algorithm

Dispersal(F,t,n):

Splitting the data M into *n* shares Si (1 ≤ i ≤ *n).*

|Si| = |M|/t. Thus the size of M, m should be a multiple of *t* (after possible padding).

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{t-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{t-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{t-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{t-1} \\ 1 & x_n & x_n^2 & & x_n^{t-1} \end{bmatrix}, \quad S = A \bullet B = \begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_n \end{bmatrix} \begin{bmatrix} B_1 & B_2 & \dots & B_{m/t} \end{bmatrix}$$

A

Recovery:

Reconstructing the original data M from any *t* shares selected from among *n* shares (Si (1 ≤ i ≤ *n)).*

$$\begin{bmatrix} A_{i_1} \\ A_{i_2} \\ \dots \\ A_{i_t} \end{bmatrix}^{-1} \begin{bmatrix} S_{i_1} \\ S_{i_2} \\ \dots \\ S_{i_t} \end{bmatrix} = B$$

Comparison of erasure codes

Reed Solomon Code:

Pros:RS-raid coding is the only general solution for all values of t and n.

Cons:Multiplication and division are complex and use Galois field, slows down the operation.

Cauchy Reed Solomon Code:

Pros: Compared to RS, code performance has been improved.

Cons: Slow compared to other erasure codes, since the encoding is not constant number of XOR's per coding block instead it is O(nlog(base 2)(t+n)) XOR's per coding block.

Rabins IDA:

Pros: offer sufficient security against reconstructing consecutive portions of data (ciphertext) based on the information stored on t-1 or fewer compromised servers.

Cons: Shares are typically a complex function of a larger subset of data which in turn affects efficiency

Comparison

Feature	RS	CRS	Rabin's IDA
Speed	Low	High	Medium
Capacity to reconstruct partial information	More	More	Less
Size of each share	D/t 8k/w (Words)	D/t >=Max(log(t),log(n-t))	D/t Equal Size

Conclusion

AONT:

- Rivest's AONT uses standard encryption and hash functions which makes it the easiest candidate to implement and gives the options to tune on speed or security.
- Very fast encoding speed can be achieved by implementing Boyko's AONT based on MGF1 for variable output hash function.
- Stinson's AONT currently doesn't have any implementation or available libraries that can be used to implement it, needs more research.

Erasure code:

- Cauchy Reed-Solomon coding outperforms classic Reed-Solomon coding significantly, as long as attention is paid to generating good encoding matrices.
- Parameter selection can have a huge impact on how well an implementation performs. Not only must the number of computational operations be considered, but also how the code interacts with the memory hierarchy, especially the caches.

Thank you !

Q&A