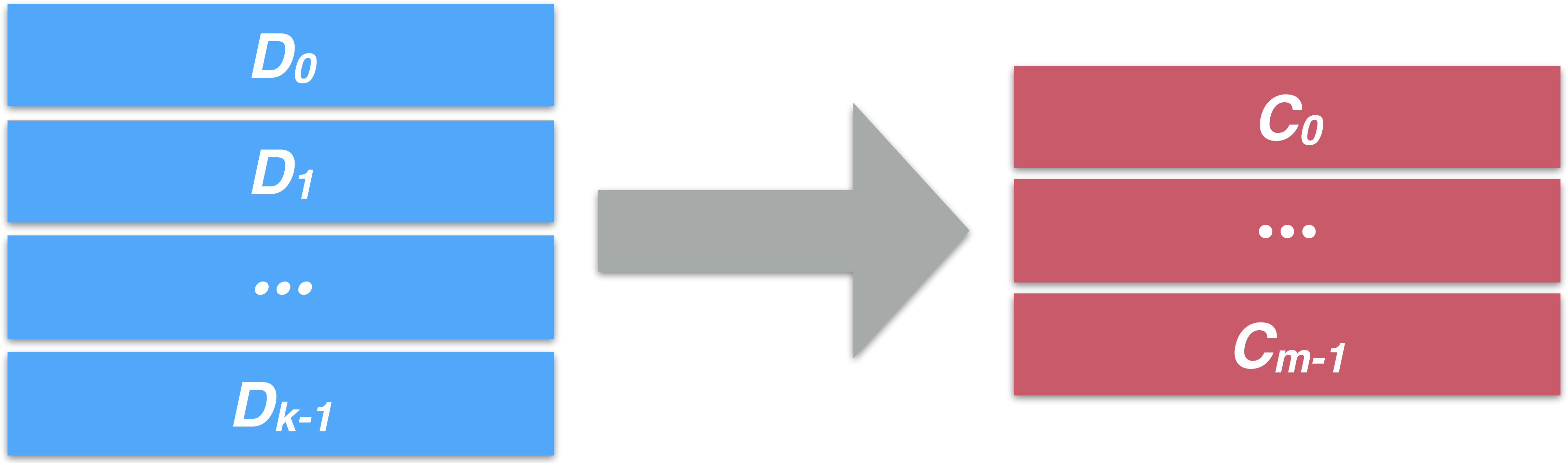
Pyrit: PolYnomial Ring Transforms

How to accelerate MDS Erasure Codes

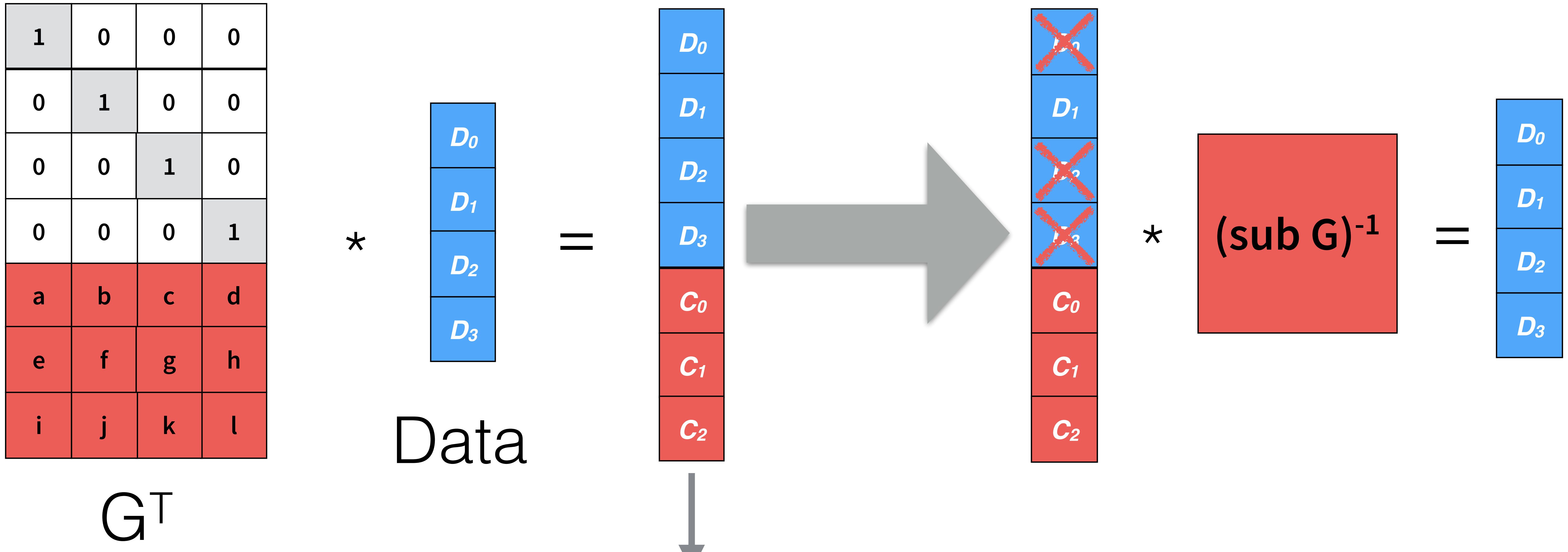
Jonathan DETCHART, Jérôme LACAN
ISAE-SUPAERO
FRANCE

Erasure codes for storage



***k** storage areas* (disks, files,…) are encoded
to generate ***m*** additional storage areas.

<u>reliability</u> : a **(n,k) code** can tolerate *m* losses among $n = k + m$ storage areas



$C_0 = a * D_0 + b * D_1 + c * D_2 + d * D_3$

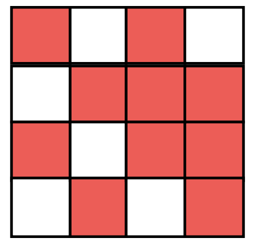
$C_1 = e * D_0 + f * D_1 + g * D_2 + h * D_3$

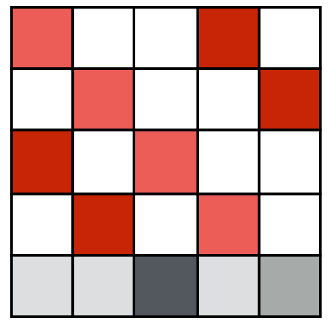
$\boxed{\text{* and + are done in a } \textbf{finite field}}$

The Polynomial Ring Transform

As operations in a field are complex, we do operations into a **ring**

- We transform the elements of a field (as polynomials) into elements of a ring.
- In the field, operations are done modulo an irreducible polynomial.
- In the ring, operations are done modulo ($x^n + 1$) => It is just a cyclic shift ! ($x^n = 1$)

Operations to multiply a storage area by 5 ($1 + x^2$) in GF(16): 

Same operations in Ring(32): 

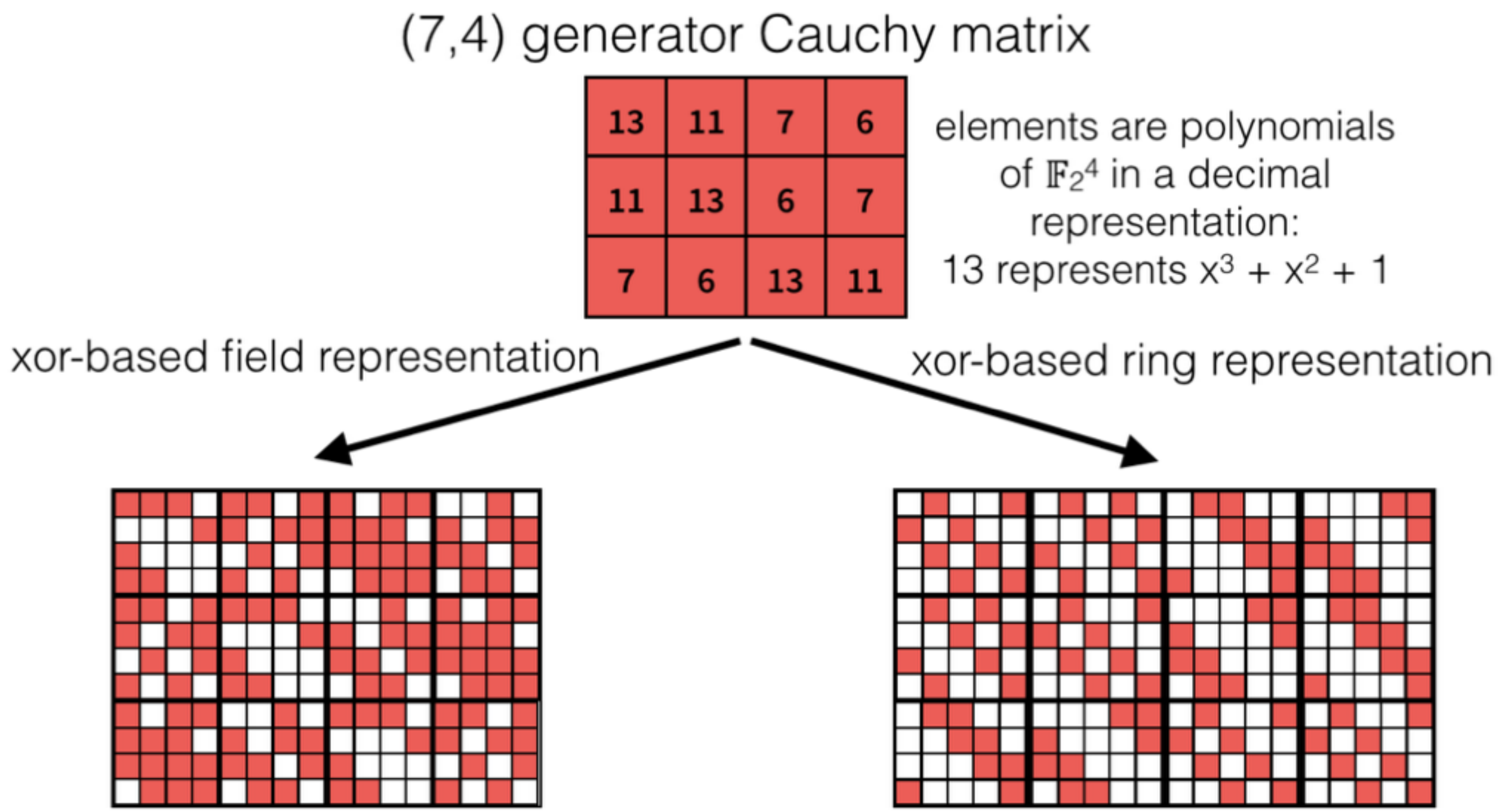
Works when the finite field is defined by a polynomial with the following properties:

- AOP (All-One Polynomials). Ex: $p(x) = x^4 + x^3 + x^2 + x + 1$ is an irreducible factor of ($x^5 + 1$)
  => **GF(16)** becomes **Ring(32)**
- ESP (Equally Spaced Polynomials). Ex: $p(x) = x^6 + x^3 + 1$ is an irreducible factor of ($x^9 + 1$)
  => **GF(64)** becomes **Ring(512)**

We propose 3 methods to make the correspondance between a field and a ring:

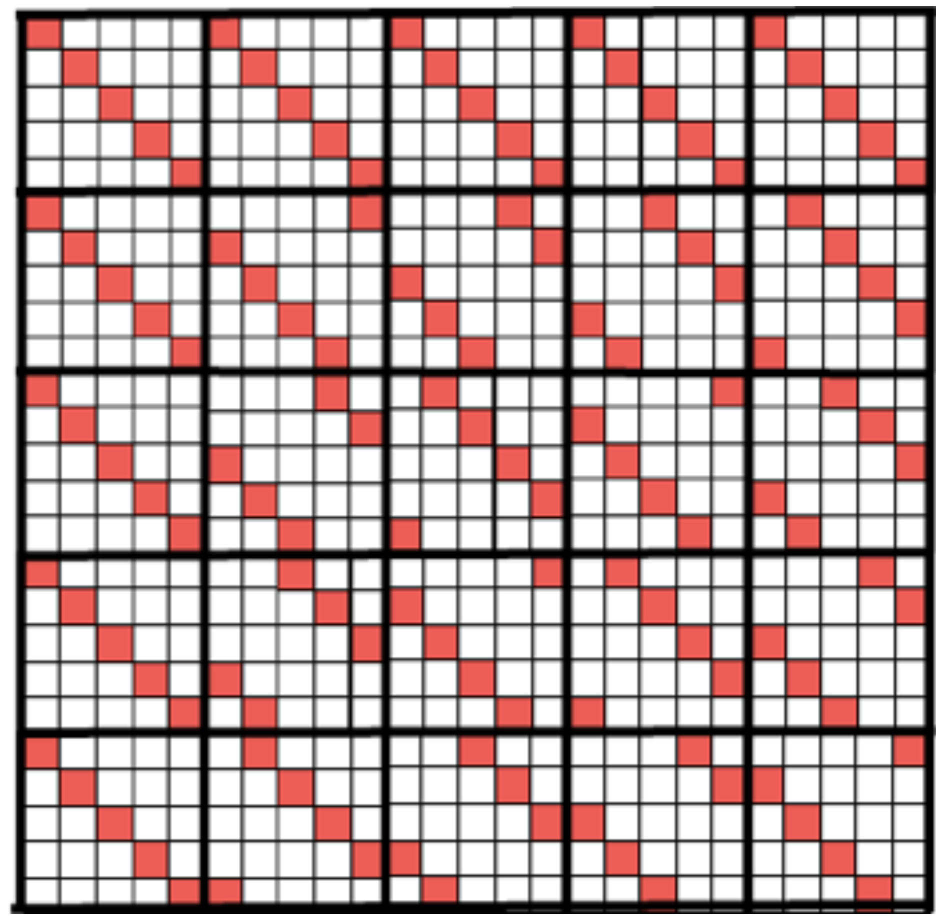
- Embedding: just consider a field element as a ring element (pad with 0)
- Parity: add a parity bit
- Sparsest representation: choose the sparsest ring element

Field VS Ring



Gain of 18% for the complexity for a Cauchy matrix

Minimal density matrices: it is possible to generate matrices with only 1 xor per source !

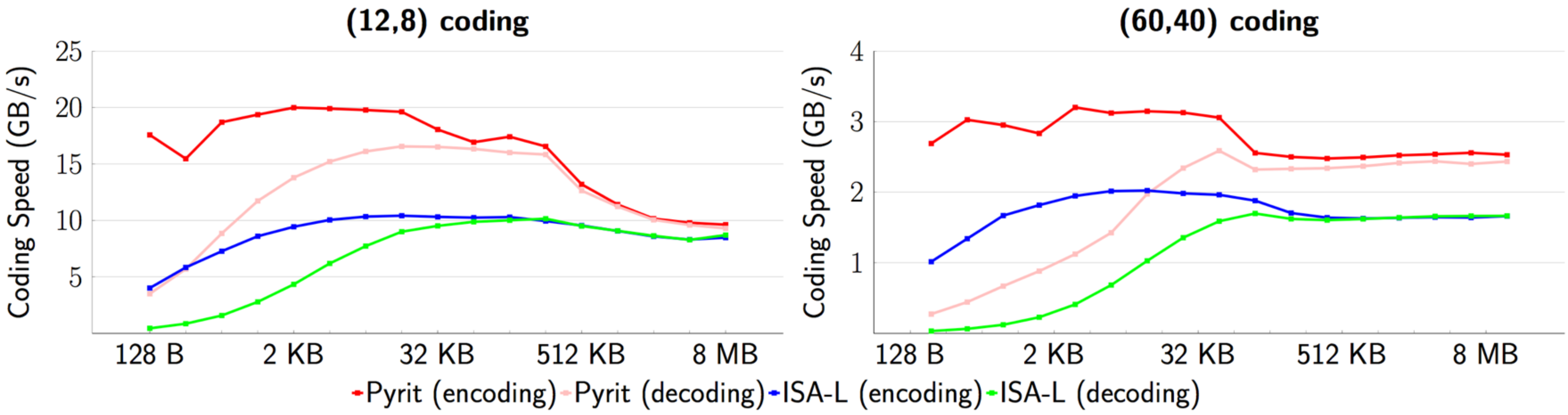


Optimal **(10,5)** Vandermonde generator matrix: from 5 sources, we generate 5 additional repairs with 1 xor by source

The results

Thanks to the ring operations, we have:

- Low-density bit matrices by construction: reduce the number of coding operations
- Data organisation: the modulo is just a cyclic shift: the elements are only composed by cyclic diagonals:
  - smaller representation in memory
  - less branches in the code
  - unrolling is easy
- Easy scheduling: thanks to the cyclic representation of the matrix elements



CPU: Intel i5-6500