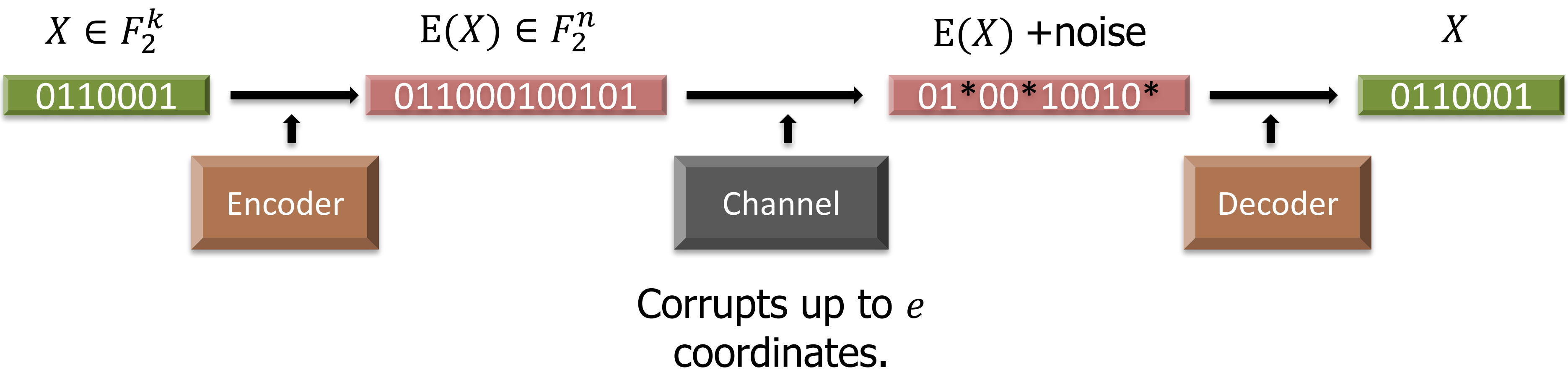
Locally decodable codes:
from computational complexity to cloud computing

Sergey Yekhanin

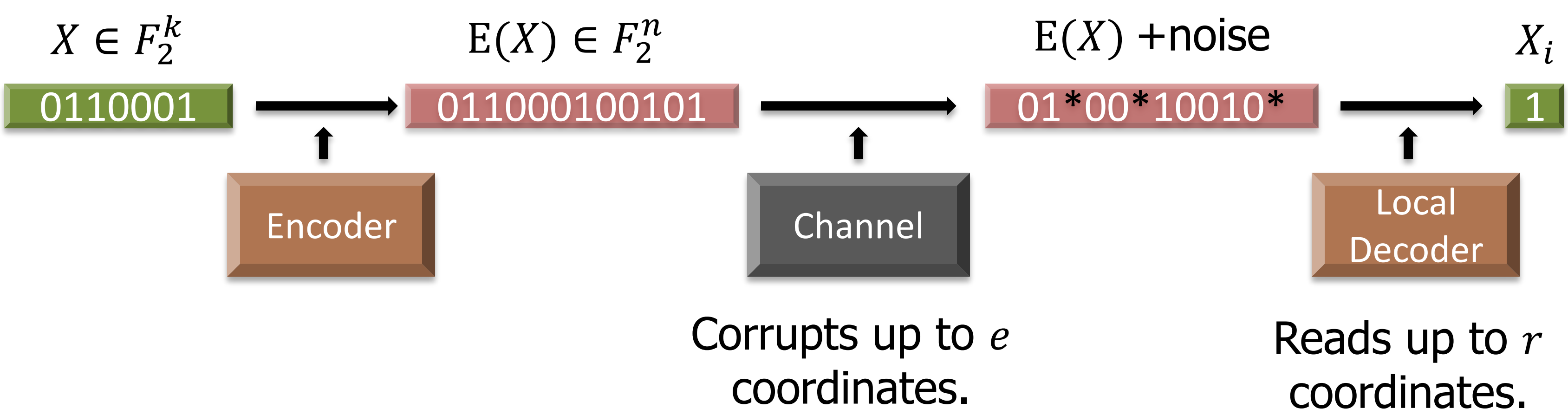
Microsoft Research

Error-correcting codes: paradigm



The paradigm dates back to 1940s (Shannon / Hamming)

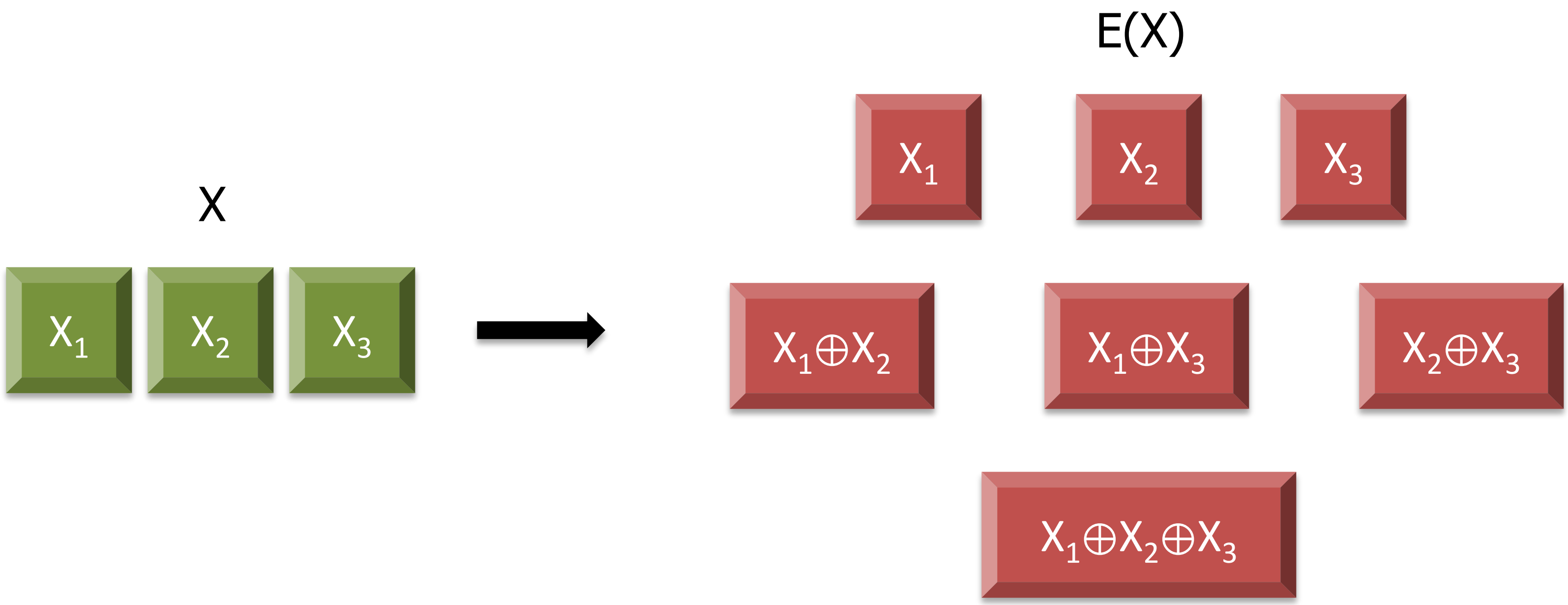
Local decoding: paradigm



Local decoder runs in time much smaller than the message length!

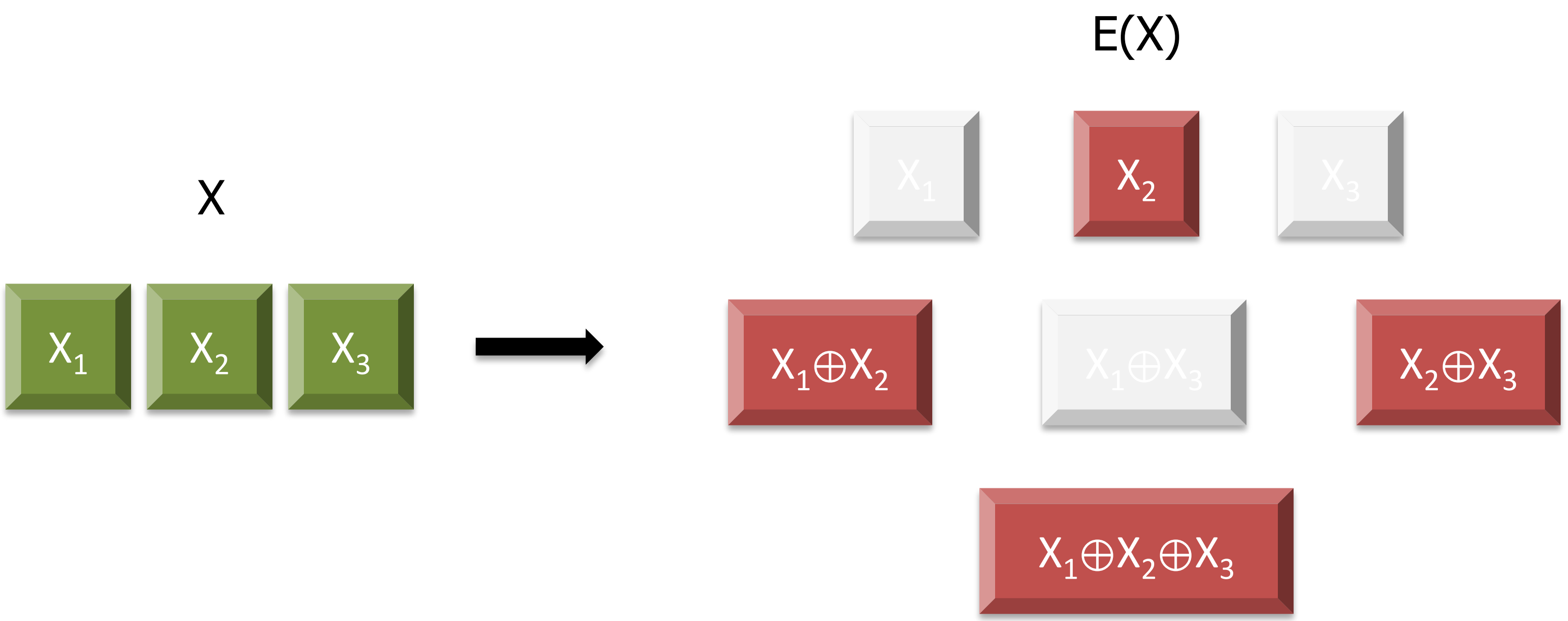
- First account: Reed's decoder for Muller's codes (1954)
- Implicit use: (1950s-1990s)
- Formal definition and systematic study (late 1990s) [Levin'95, STV'98, KT'00]
  - Original applications in computational complexity theory
  - Cryptography
  - Most recently used in practice to provide reliability in distributed storage

Local decoding: example



Message length: $k = 3$
Codeword length: $n = 7$
Corrupted locations: $e = 3$
Locality: $r = 2$

Local decoding: example

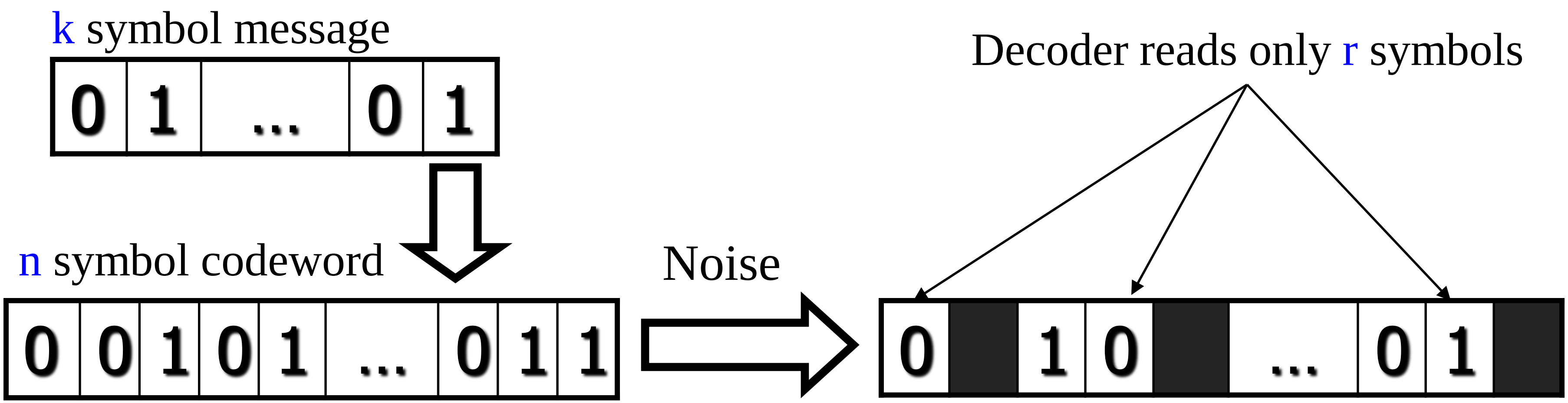


Message length: $\mathrm{k} = 3$
Codeword length: $\mathrm{n} = 7$
Corrupted locations: $e = 3$
Locality: $r = 2$

Locally decodable codes

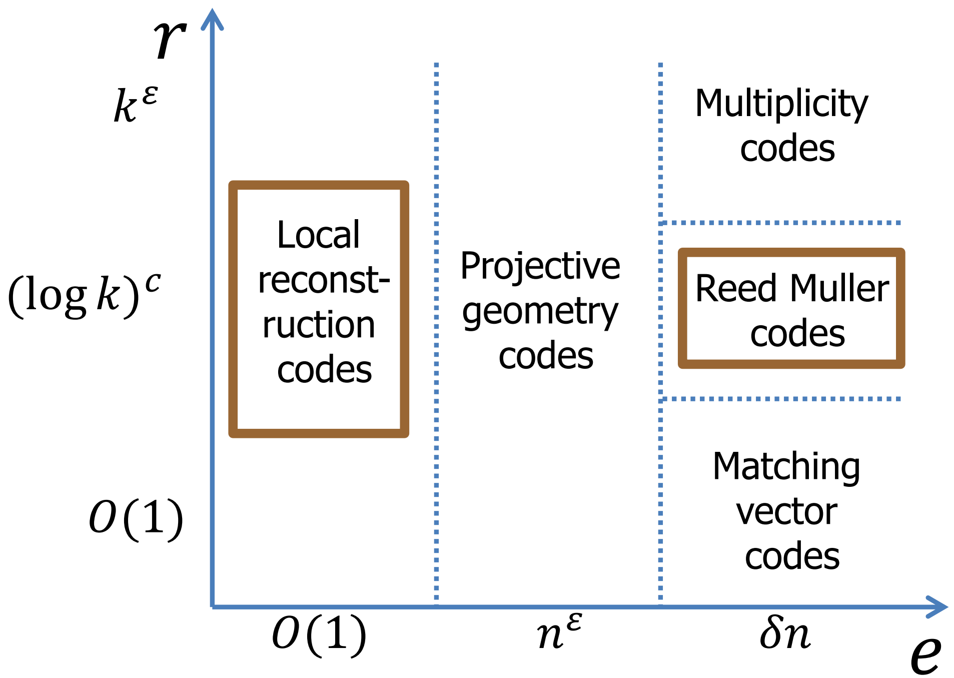
Definition: A code $E: F_q^k \to F_q^n$ is r-locally decodable, if for every message X, each X_i can be recovered from reading some r symbols of $E(X)$, even after up to e coordinates of $E(X)$ are corrupted.

- (Erasures.) Decoder is aware of erased locations. Output is always correct.
- (Errors.) Decoder is randomized. Output is correct with probability 99%.



Locally decodable codes

<u>Goal</u>:
Understand the true shape of the tradeoff between redundancy $n - k$ and locality r, for different settings of e (e.g., $e = \delta n, n^{\epsilon}, O(1)$.)



Taxonomy of known families of LDCs

Plan

- Part I: (Computational complexity)
  - Average case hardness
  - An avg. case hard language in EXP (unless EXP $\subseteq$ BPP)
  - Construction of LDCs
  - Open questions

- Part II: (Distributed data storage)
  - Erasure coding for data storage
  - LDCs for data storage
  - Constructions and limitations
  - Open questions

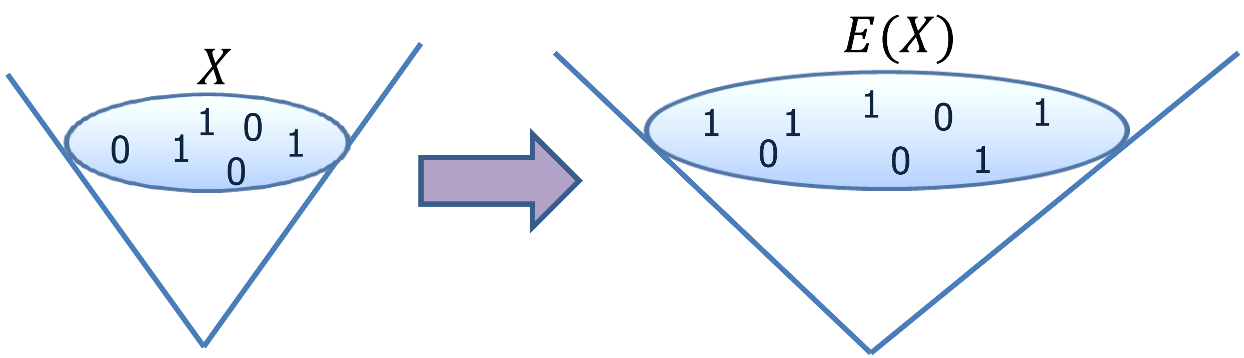
Part I: Computational complexity

Average case complexity

- A problem is hard-on-average if any efficient algorithm errs on 10% of the inputs.
- Establishing hardness-on-average for a problem in NP is a major problem.
- Below we establish hardness-on-average for a problem in EXP, assuming EXP $\not\subseteq$ BPP.

Construction [STV]:

Level k is a string X of length 2^k



L is EXP-complete

L' is in EXP

E: $F_2^k \to F_2^n$
$n = poly(k)$,
$r = (\log k)^c$,
$e = n/10$.

Theorem: If there is an efficient algorithm that errs on <10% of L'; then EXP $\subseteq$ BPP.

Average case complexity

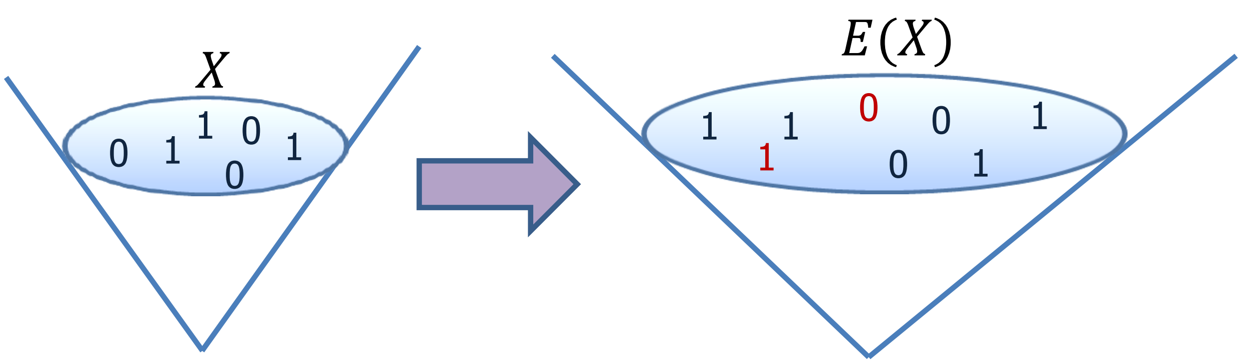
<u>Theorem</u>: If there is an efficient algorithm that errs on <10% of L'; then EXP $\subseteq$ BPP.

<u>Proof</u>: We obtain a BPP algorithm for L:

- Let A be the algorithm that errs on <10% of L';
  A gives us access to the corrupted encoding $E(X)$.

- To decide if X_i invoke the local decoder for $E(X)$.
- Time complexity is $(\log 2^k)^c * poly(k) = poly(k)$.

- Output is correct with probability 99%.



L is EXP-complete

L' is in EXP

E: $F_2^k \to F_2^n$
$n = poly(k)$,
$r = (\log k)^c$,
$e = n/10$.

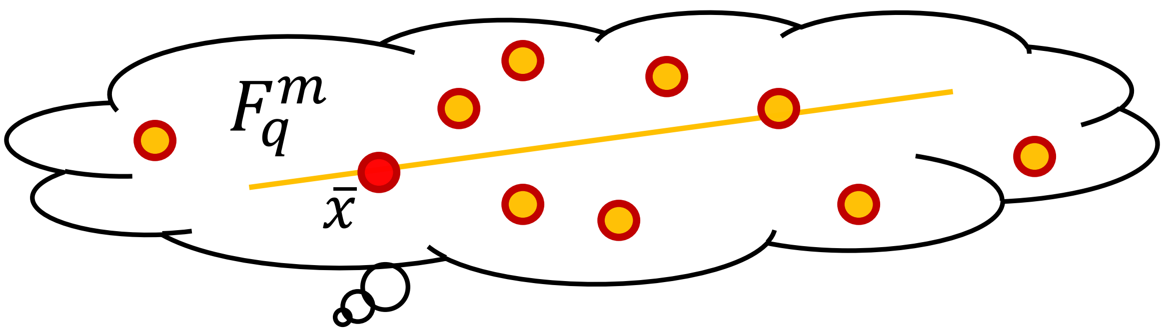
Reed Muller codes

- Parameters: $q, m, d = (1 - 4\delta)q$.
- Codewords: evaluations of degree d polynomials in m variables over F_q.
- Polynomial $f \in F_q[z_1, \dots, z_m]$, $\deg \mathrm{f} < d$ yields a codeword: $\langle f(\bar{x}) \rangle_{\bar{x} \in F_q^m}$
- Parameters: $n = q^m, \quad k = \binom{m+d}{m}, \;\; r = q - 1, \;\; e = \delta n$.

Reed Muller codes: local decoding

- Key observation: Restriction of a codeword to an affine line yields an evaluation of a univariate polynomial $f|_L$ of degree at most d.

- To recover the value at $\bar{x}$:
  - Pick a random affine line through $\bar{x}$.
  - Do noisy polynomial interpolation.



- Locally decodable code: Decoder reads $q-1$ random locations.

Reed Muller codes: parameters

$$n = q^m, \qquad k = \binom{m+d}{m}, \qquad d = (1 - 4\delta)q, \qquad r = q - 1, \qquad e = \delta n.$$

<u>Setting parameters</u>:

- q $= O(1),\ m \to \infty$: $\quad r = O(1),\ n = \exp\left(k^{\frac{1}{r-1}}\right).$
- q $= m^2$: $\quad r = (\log k)^2,\ n = poly(k).$
- q $\to \infty,\ m = O(1)$: $\quad r = k^{\epsilon},\ n = O(k).$

Better codes are known

Reducing codeword length is a major open question.

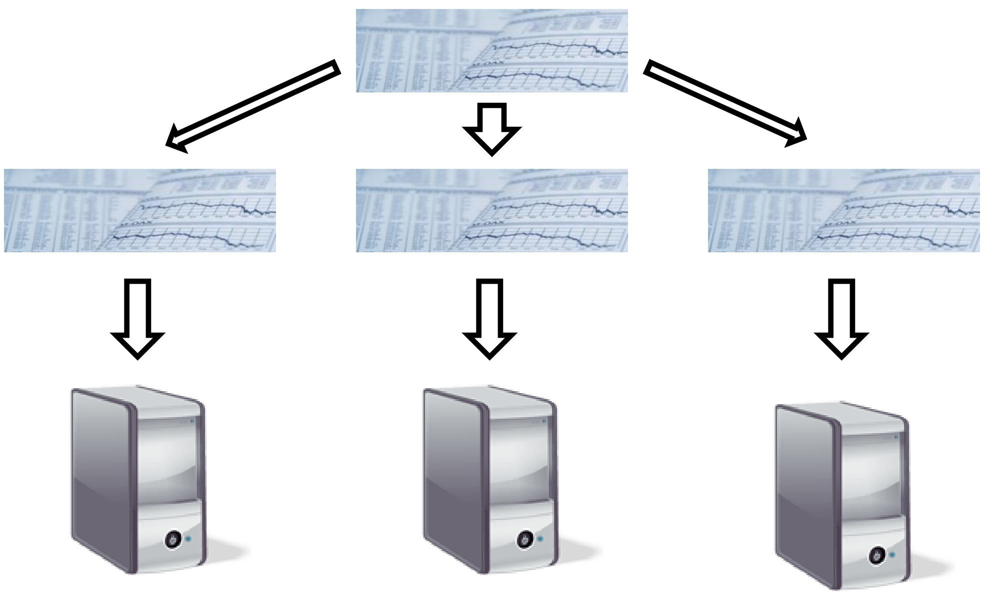
Part II: Distributed storage

Data storage

- Store data reliably
- Keep it readily available for users

Data storage: Replication

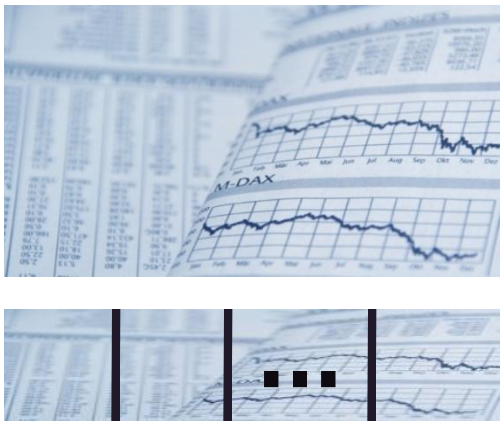
- Store data reliably
- Keep it readily available for users



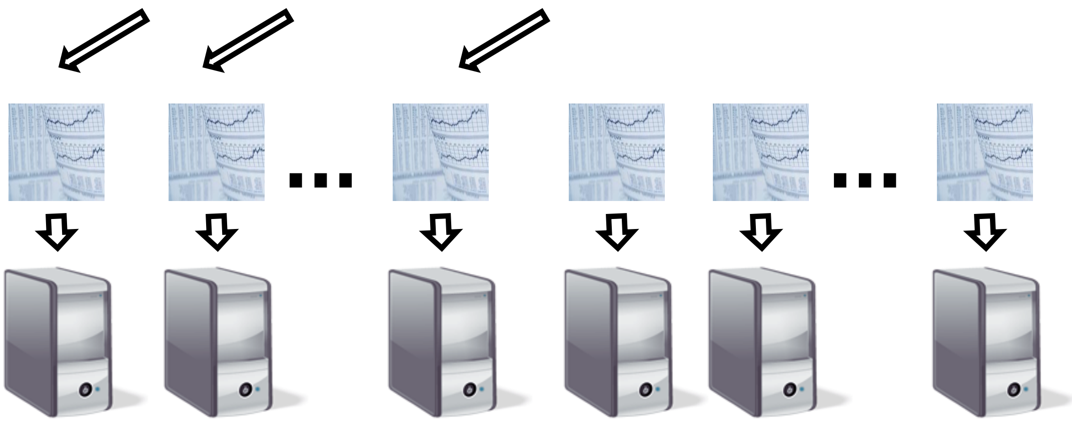
- Very large overhead
- Moderate reliability

- <u>Local recovery</u>:
  Lose one machine, access one

Data storage: Erasure coding



- Store data reliably
- Keep it readily available for users



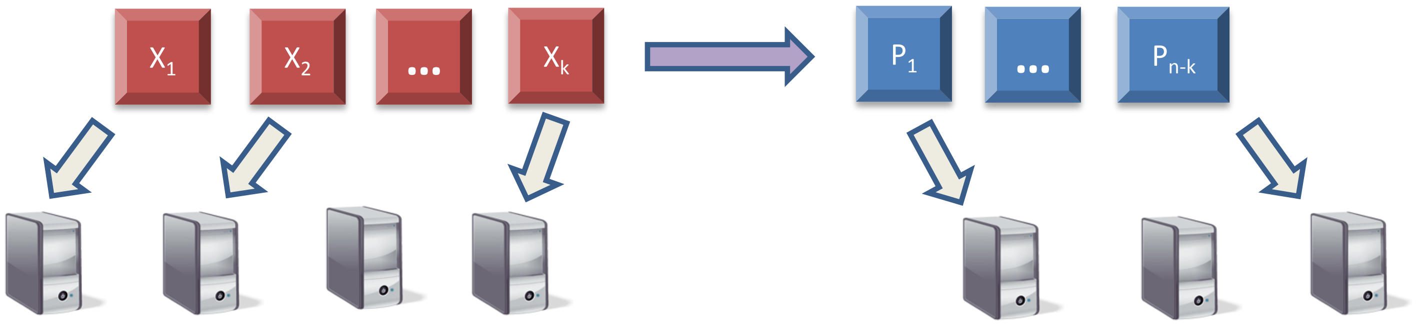
- Low overhead
- High reliability

- No local recovery:
  Loose one machine, access k

k data chunks n-k parity chunks

Need: Erasure codes with local decoding

Codes for data storage



- Goals:
  - (Cost) minimize the number of parities.
  - (Reliability) tolerate any pattern of $h+1$ simultaneous failures.
  - (Availability) recover any data symbol by accessing at most r other symbols
  - (Computational efficiency) use a small finite field to define parities.

Local reconstruction codes

- Def: An (r,h) – Local Reconstruction Code (LRC) encodes k symbols to n symbols, and
    - Corrects any pattern of h+1 simultaneous failures;
    - Recovers any single erased data symbol by accessing at most r other symbols.

Local reconstruction codes

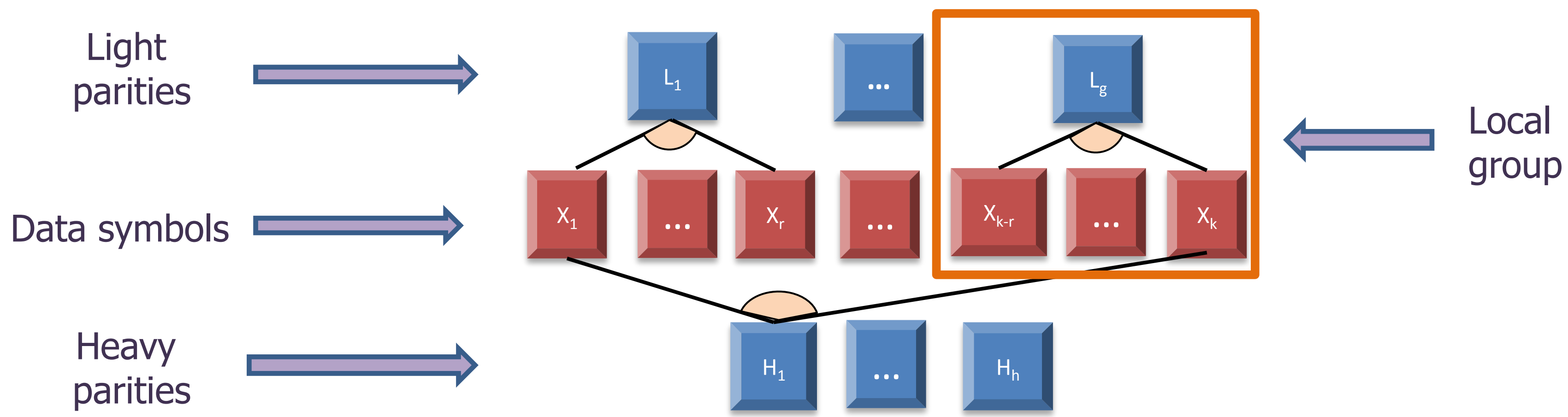
- Def: An (r,h) – Local Reconstruction Code (LRC) encodes k symbols to n symbols, and
  - Corrects any pattern of h+1 simultaneous failures;
  - Recovers any single erased data symbol by accessing at most r other symbols.
- Theorem[GHSY]: In any (r,h) – (LRC), redundancy n-k satisfies $n - k \geq \left\lceil \frac{k}{r} \right\rceil + h.$

Local reconstruction codes

- Def: An (r,h) – Local Reconstruction Code (LRC) encodes k symbols to n symbols, and
  - Corrects any pattern of h+1 simultaneous failures;
  - Recovers any single erased data symbol by accessing at most r other symbols.

- Theorem[GHSY]: In any (r,h) – (LRC), redundancy n-k satisfies $n - k \geq \left\lceil \frac{k}{r} \right\rceil + h.$

- Theorem[GHSY]: If r | k and h<r+1; then any (r,h) – LRC has the following topology:

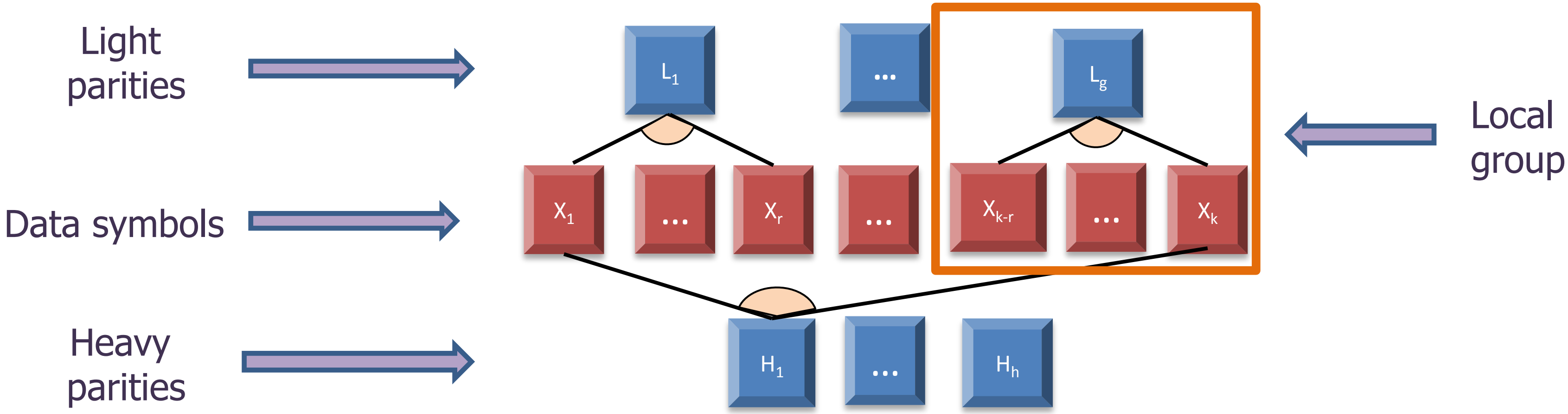


Local reconstruction codes

- Def: An (r,h) – Local Reconstruction Code (LRC) encodes k symbols to n symbols, and
  - Corrects any pattern of h+1 simultaneous failures;
  - Recovers any single erased data symbol by accessing at most r other symbols.

- Theorem[GHSY]: In any (r,h) – (LRC), redundancy n-k satisfies $n - k \geq \left\lceil \frac{k}{r} \right\rceil + h.$

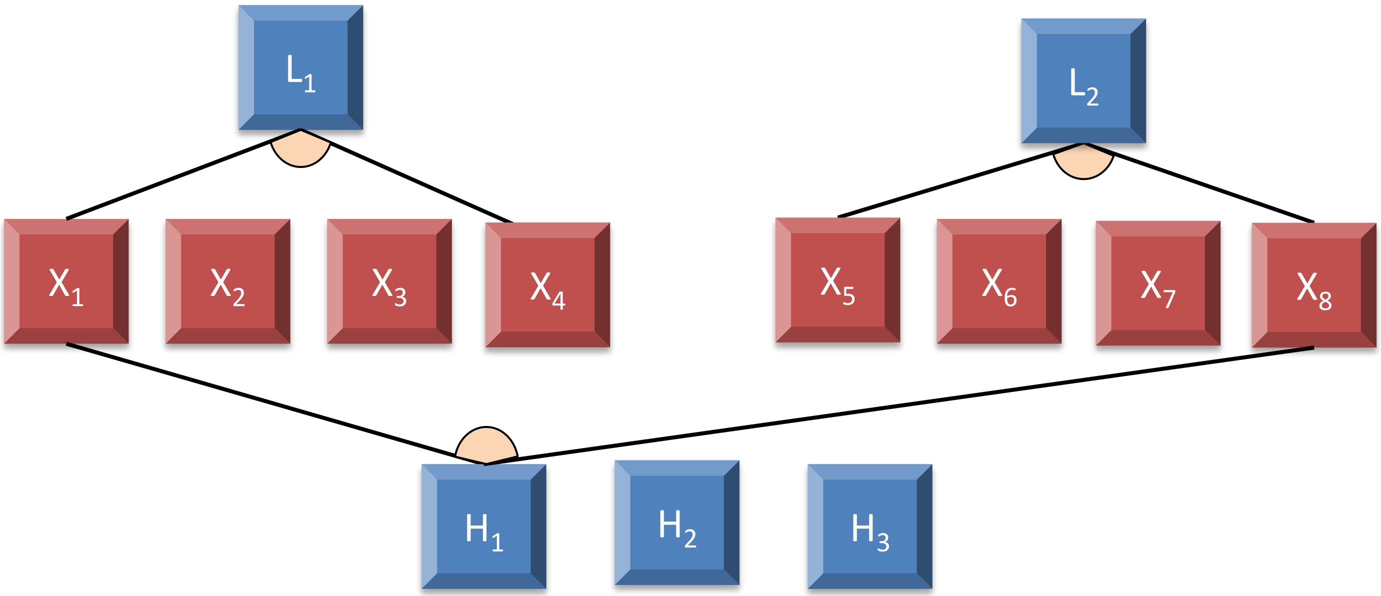
- Theorem[GHSY]: If r | k and h<r+1; then any (r,h) – LRC has the following topology:



- Fact: There exist (r,h) – LRCs with optimal redundancy over a field of size k+h.

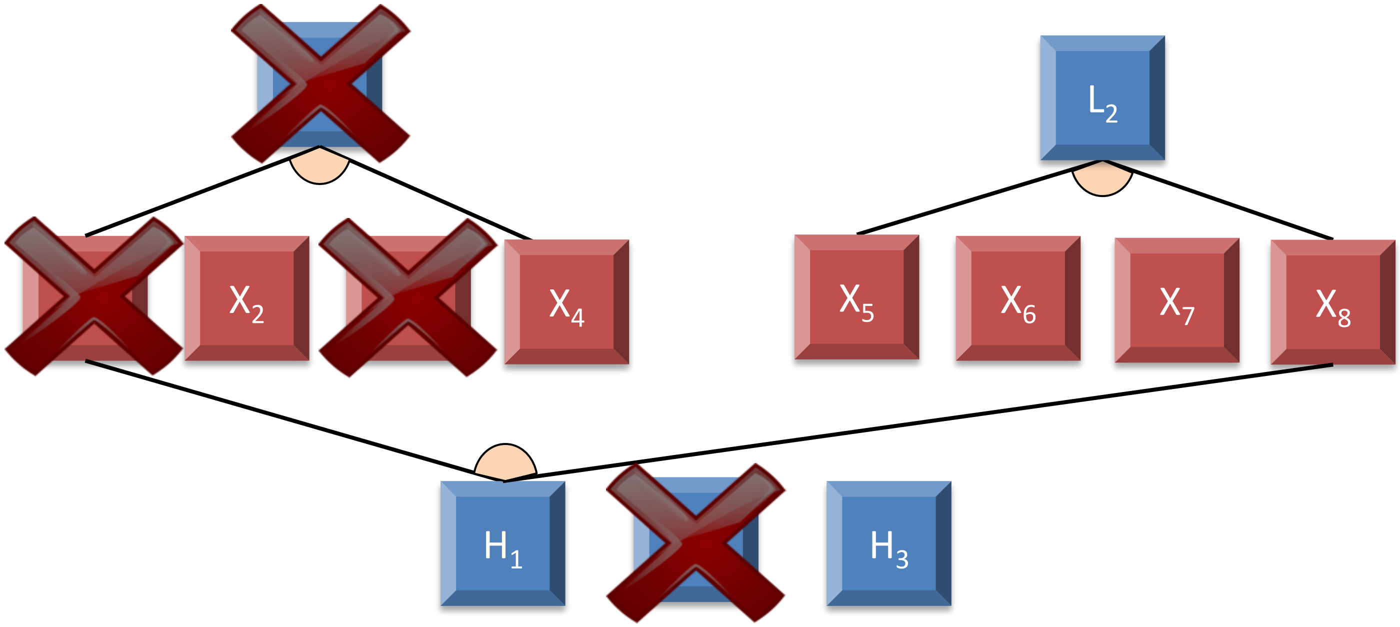
Reliability

Set k=8, r=4, and h=3.



Reliability

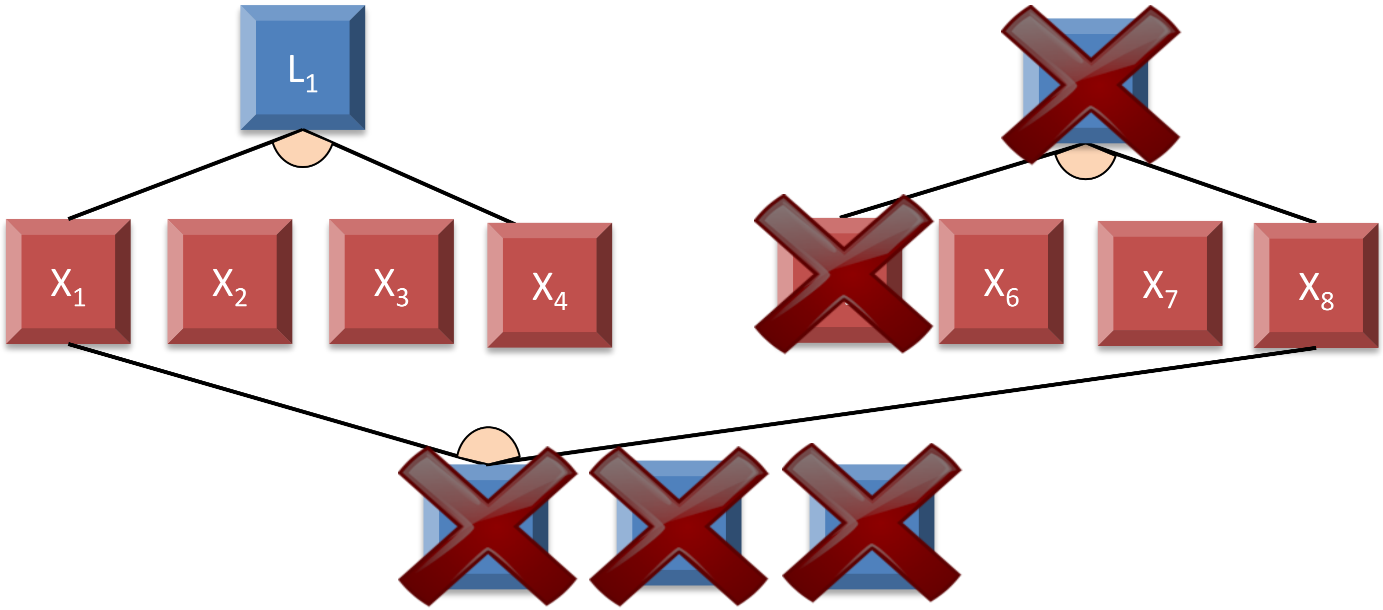
Set k=8, r=4, and h=3.



- All 4-failure patterns are correctable.

Reliability

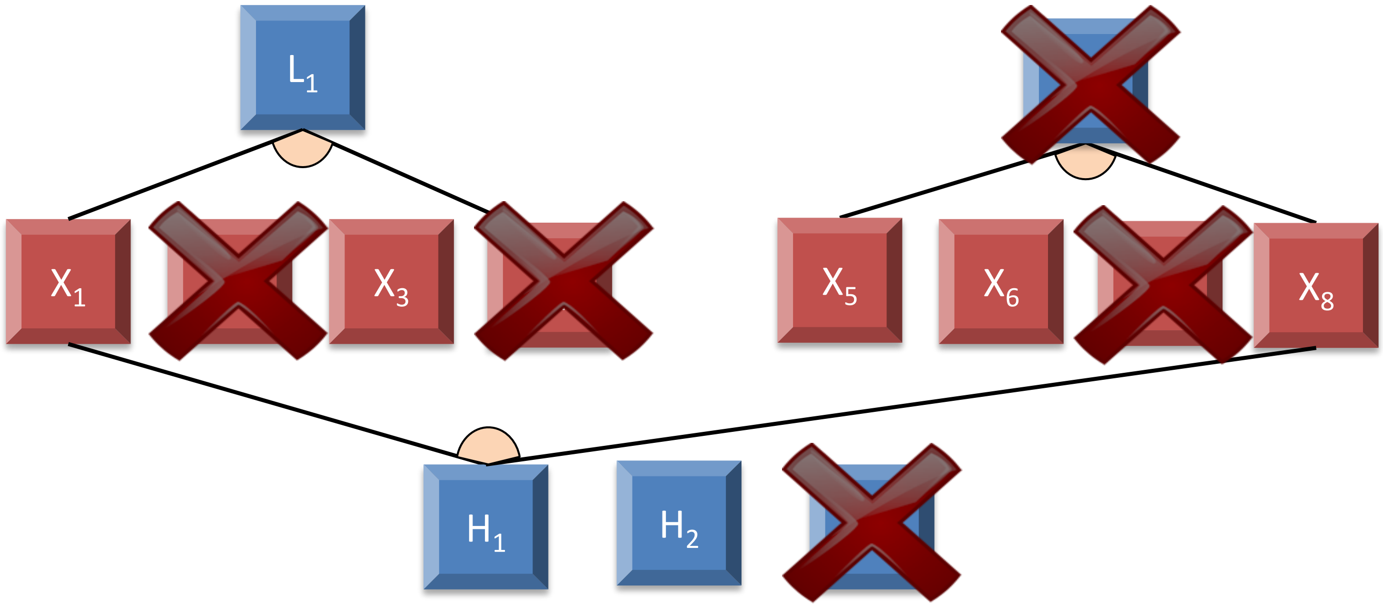
Set k=8, r=4, and h=3.



- All 4-failure patterns are correctable.
- Some 5-failure patterns are not correctable.

Reliability

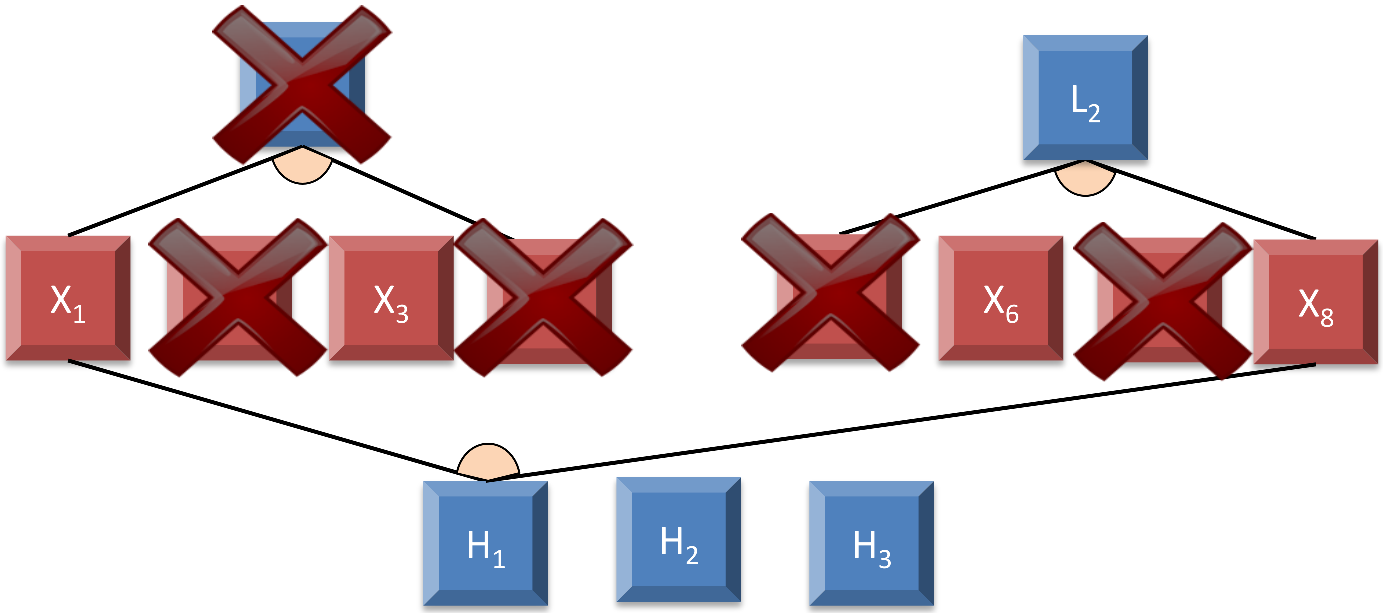
Set k=8, r=4, and h=3.



- All 4-failure patterns are correctable.
- Some 5-failure patterns are not correctable.
- Other 5-failure patterns might be correctable.

Reliability

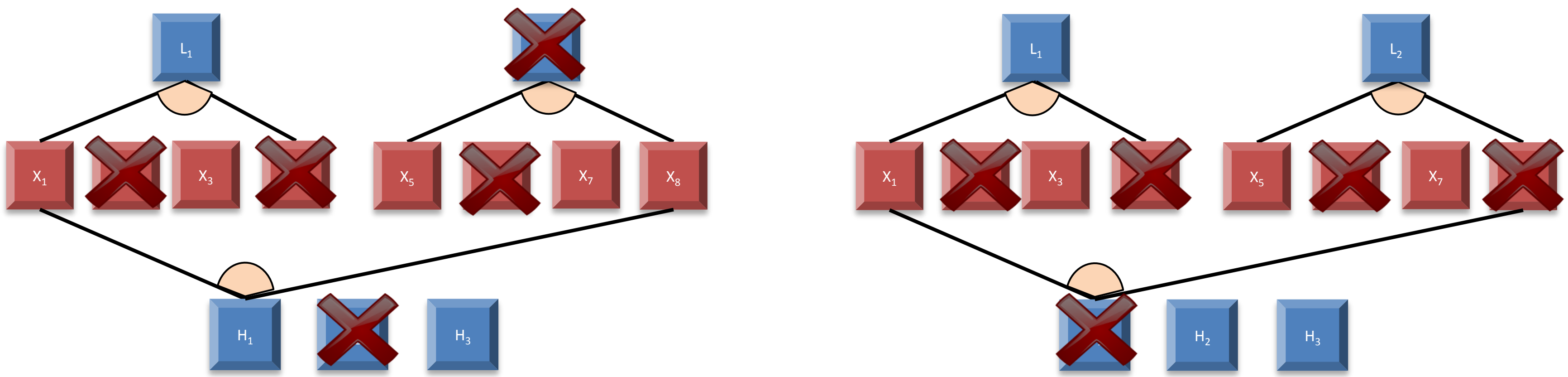
Set k=8, r=4, and h=3.



- All 4-failure patterns are correctable.
- Some 5-failure patterns are not correctable.
- Other 5-failure patterns might be correctable.

Combinatorics of correctable failure patterns

<u>Def</u>: A regular failure pattern for a (r,h)-LRC is a pattern that can be obtained by failing one symbol in each local group and h extra symbols.



<u>Theorem</u>:

- Every failure pattern that is not dominated by a regular failure pattern is not correctable by any LRC.
- There exist LRCs that correct all regular failure patterns.

Maximally recoverable codes

Def: An (r,h)-LRC is maximally recoverable if it corrects all regular failure patterns.

Theorem: Maximally reliable (r,h)-LRCs exist.

Proof sketch: Pick the coefficients in heavy parities at random from a large finite field.

Asymptotic setting: $h = O(1),\ r = O(1),\ k \to \infty$.

Random choice needs a field of size at least: $\Omega\left(k^{h-1}\right)$.

The tradeoff: Larger fields allow for more reliable codes up to maximal recoverability.

We want both: small field size (efficiency) and maximal recoverability.

Explicit maximally recoverable codes

Theorem[GHJY]: There exist maximally recoverable (r,h)-LRC over a field of size

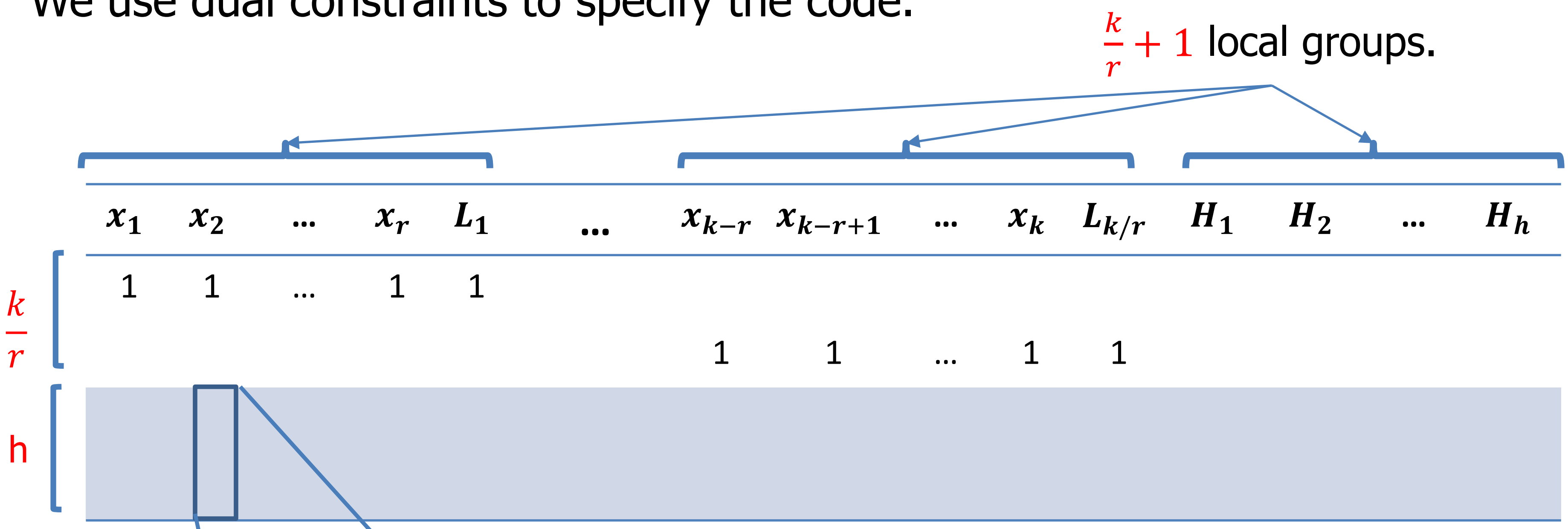
$$ck^{\left\lceil (h-1)\left(1-\frac{1}{2^r}\right)\right\rceil}.$$

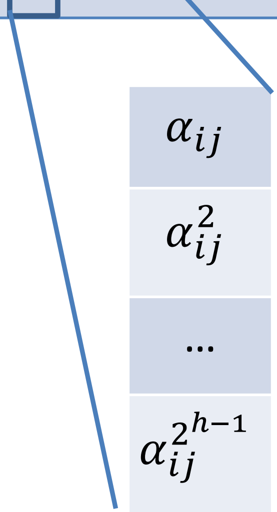
Comparison:

- Our alphabet grows as $O\left(k^{h-1}\right)$ or slower.
- Beats random codes for small h and large h.
- Our only lower bound for the alphabet size thus far is k+1 independent of h.

Code construction

We use dual constraints to specify the code.





Element α_{ij} appears in the j-th column of the i-th group.

We consider a sequence field extensions $F_2 \subseteq F_{2^a} \subseteq F_{2^b}$.

$\{\xi_j\} \subseteq F_{2^a}$ form a basis over F_2.

$\{\lambda_i\} \subseteq F_{2^b}$ are h-independent over F_{2^a}.

$\alpha_{ij} = \xi_j \times \lambda_i$.

Erasure correction

k=8, r=4, h=2.

x_1 (erased)	x_2 (erased)	x_3	x_4	L_1	x_5 (erased)	x_6 (erased)	x_7	x_8	L_2	H_1 (erased)	H_2	H_3
1	1	1	1	1								
					1	1	1	1	1			
α_{11}	α_{12}				α_{21}	α_{22}				α_{31}		
α_{11}^2	α_{12}^2				α_{21}^2	α_{22}^2				α_{31}^2		
α_{11}^4	α_{12}^4				α_{21}^4	α_{22}^4				α_{31}^4		

→

1	1			
		1	1	
α_{11}	α_{12}	α_{21}	α_{22}	α_{31}
α_{11}^2	α_{12}^2	α_{21}^2	α_{22}^2	α_{31}^2
α_{11}^4	α_{12}^4	α_{21}^4	α_{22}^4	α_{31}^4

→

$\alpha_{11}+\alpha_{12}$	$\alpha_{21}+\alpha_{22}$	α_{31}
$\alpha_{11}^2+\alpha_{12}^2$	$\alpha_{21}^2+\alpha_{22}^2$	α_{31}^2
$\alpha_{11}^4+\alpha_{12}^4$	$\alpha_{21}^4+\alpha_{22}^4$	α_{31}^4

→

$(\alpha_{11}+\alpha_{12})$	$(\alpha_{21}+\alpha_{22})$	α_{31}
$(\alpha_{11}+\alpha_{12})^2$	$(\alpha_{21}+\alpha_{22})^2$	α_{31}^2
$(\alpha_{11}+\alpha_{12})^4$	$(\alpha_{21}+\alpha_{22})^4$	α_{31}^4

→

$(\alpha_{11}+\alpha_{12})$	$(\alpha_{21}+\alpha_{22})$	α_{31}

→

$(\xi_1+\xi_2)\times\lambda_1$	$(\xi_1+\xi_2)\times\lambda_2$	$\xi_1\times\lambda_3$

Looking forward

The main challenge in LRC design is to obtain maximally reliable codes over small finite fields. Empirical evidence suggests that there is a tradeoff between reliability and computational efficiency.

Open questions:

- Study the tradeoff between redundancy and locality.
- Develop tight bounds for redundancy when e is a constant larger than 1.