# Algorithms for Noisy Broadcast Under Erasures

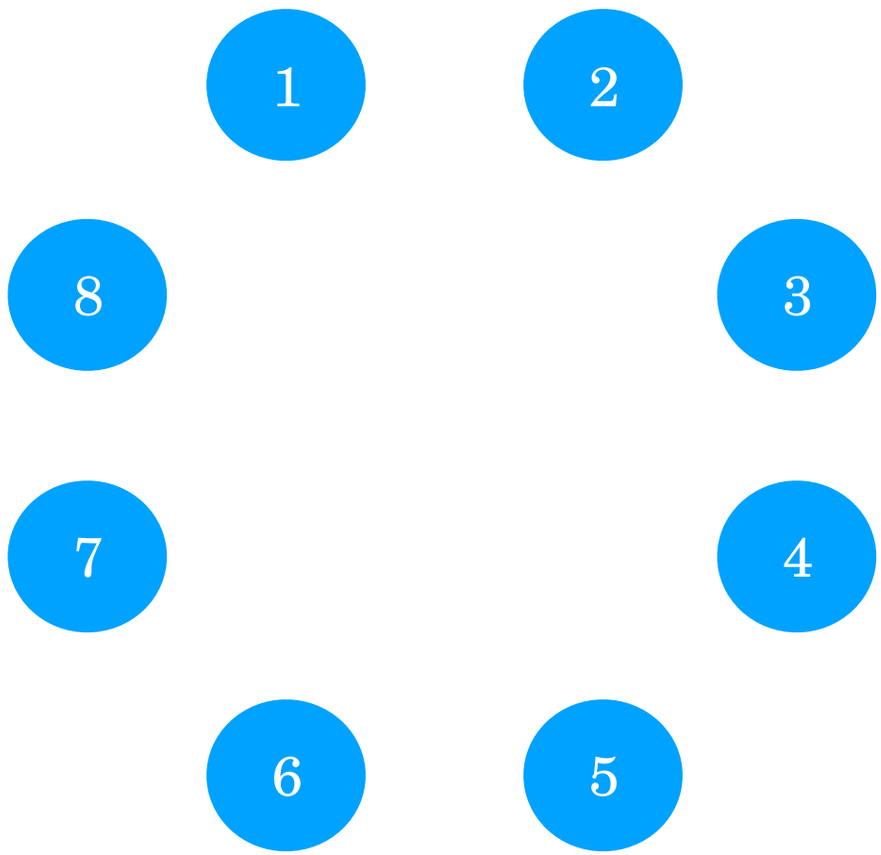
Sidhanth Mohanty
Carnegie Mellon University

Joint work with

Ofer Grossman
MIT

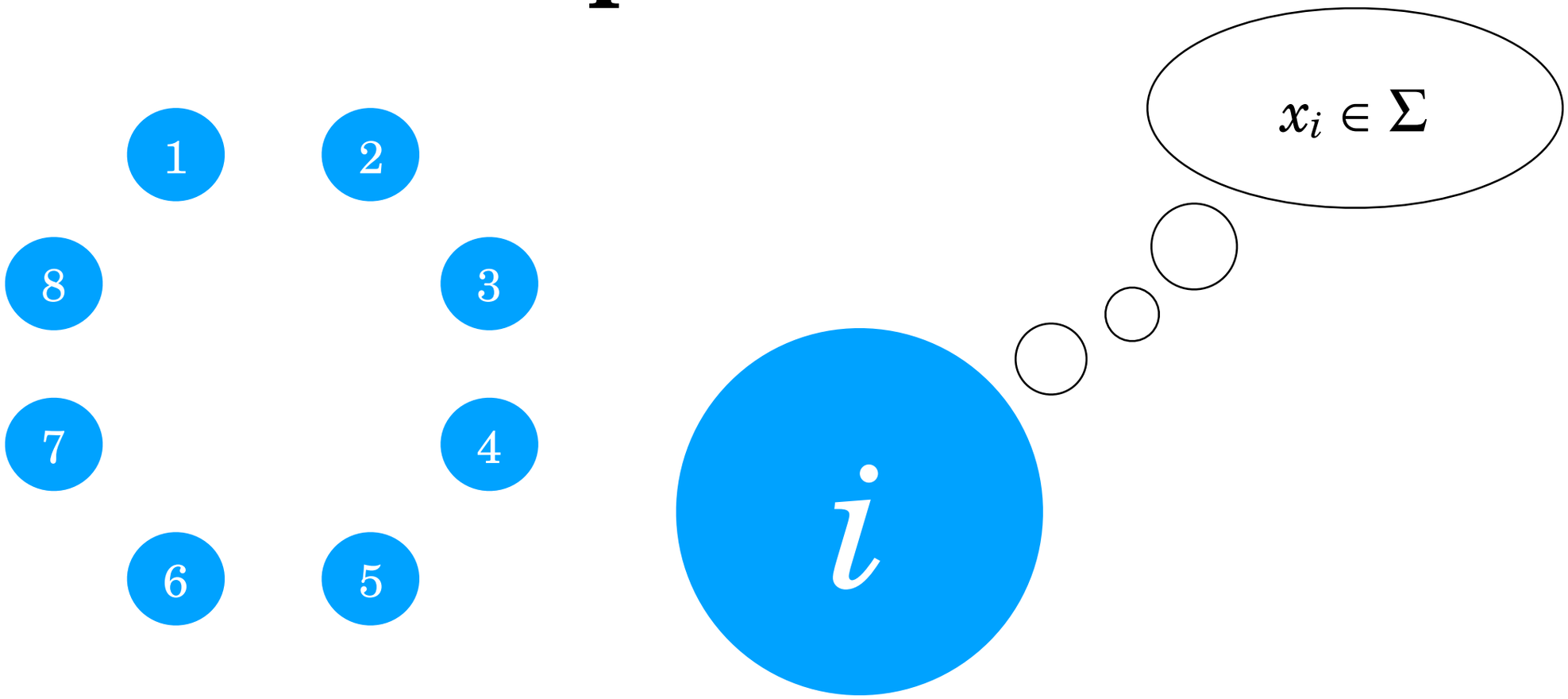
Bernhard Haeupler
Carnegie Mellon University

# $n$ processors



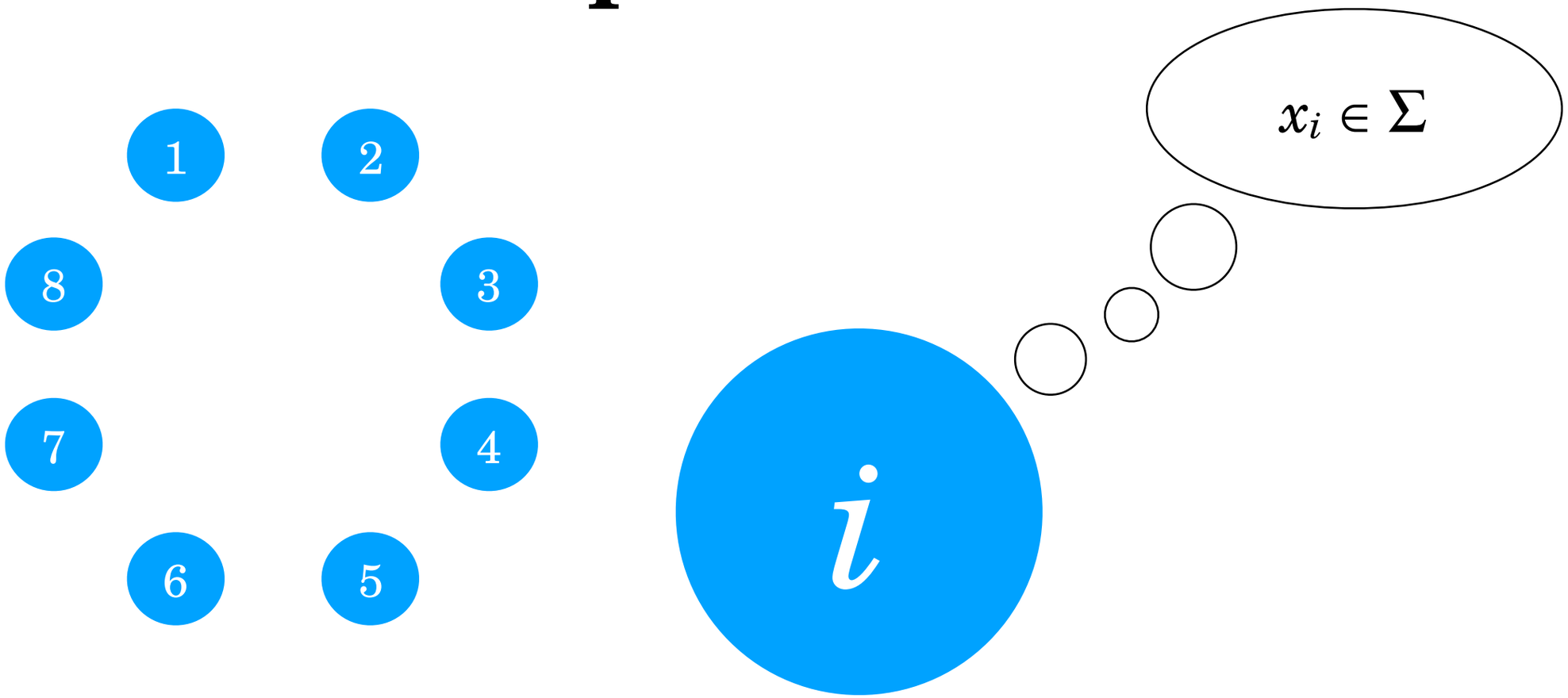

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# $n$ processors




**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

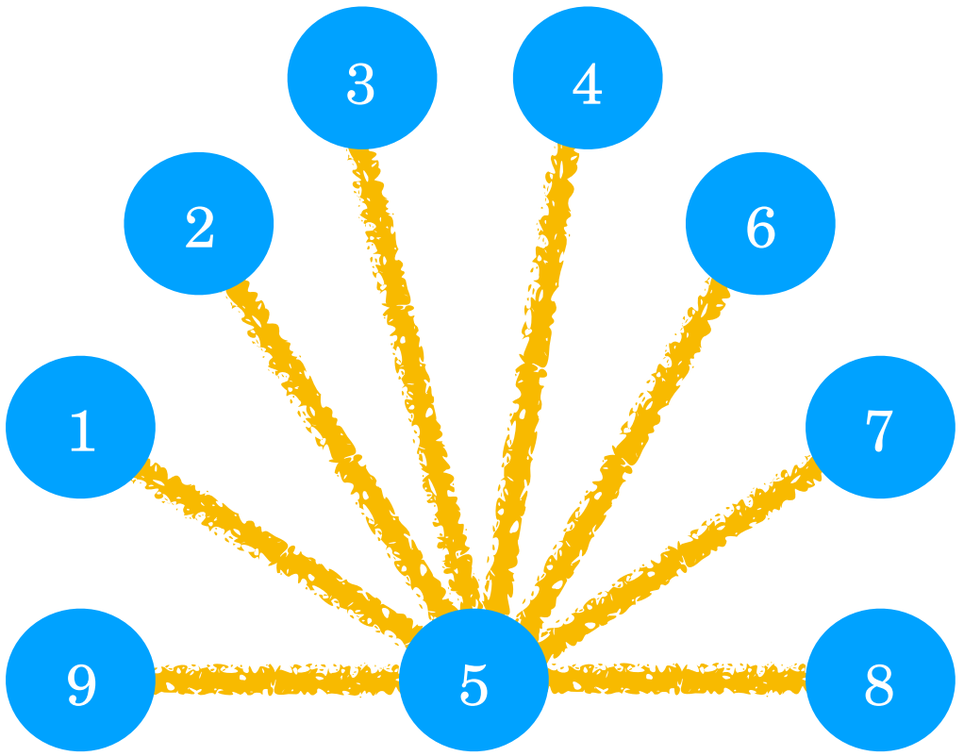
# $n$ processors




**Goal:** processors 1, …, $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Noisy Broadcast Model

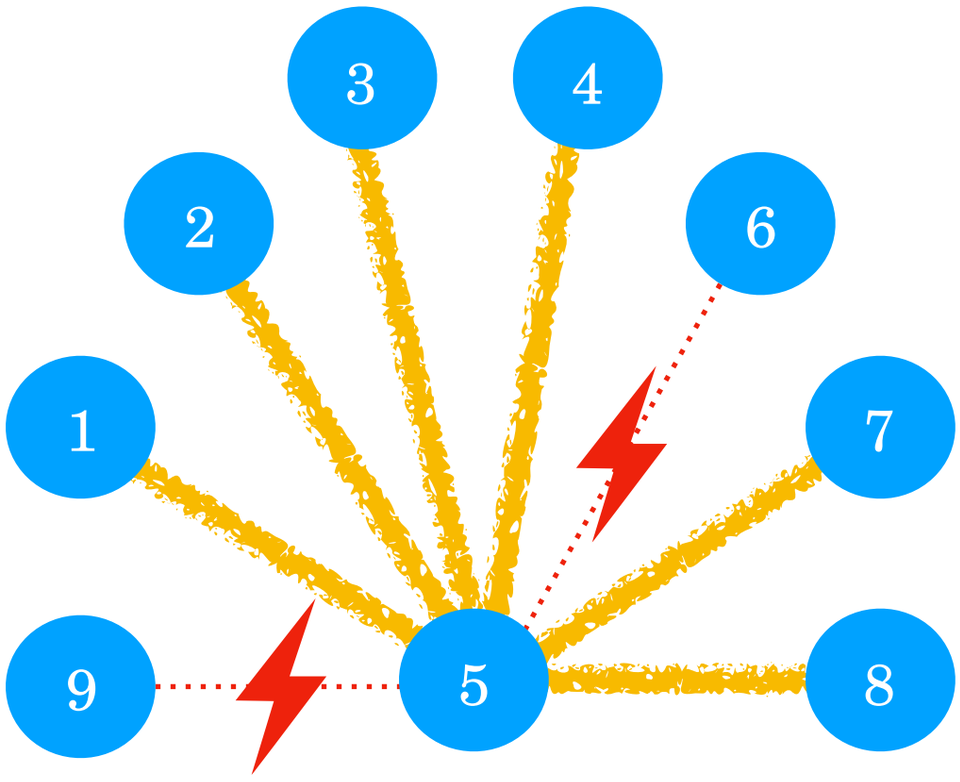



**View from processor 5**

- Each processor $i$ for $1 \leq i \leq n$ broadcasts a character in $\Sigma$ to all $j$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Noisy Broadcast Model

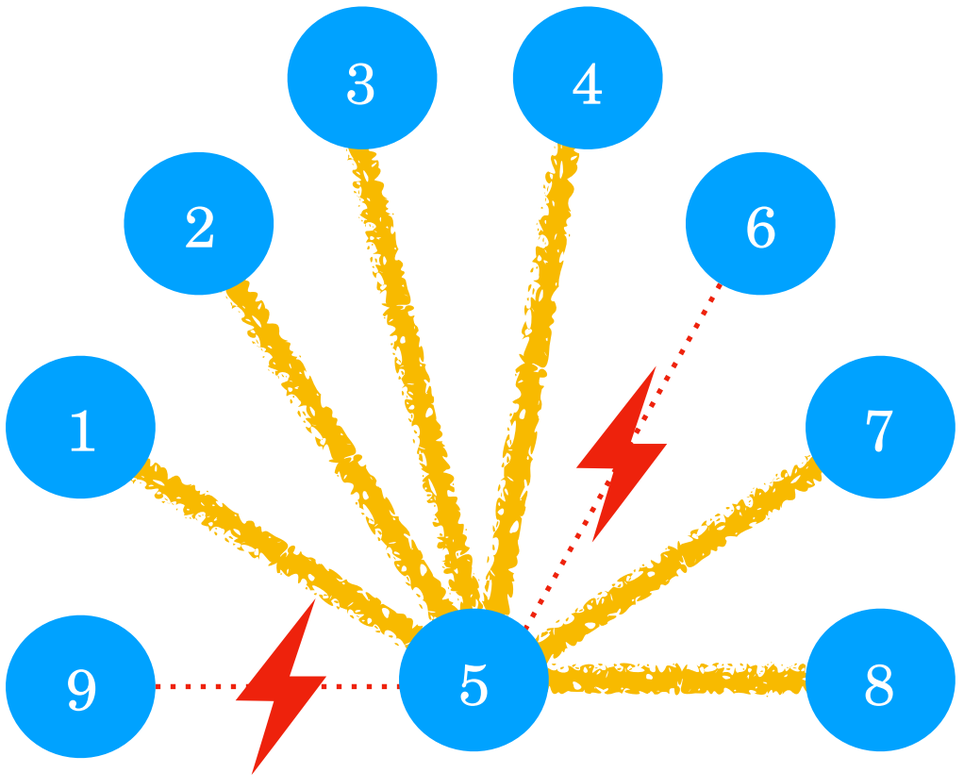



**View from processor 5**

- Each processor $i$ for $1 \le i \le n$ broadcasts a character in $\Sigma$ to all $j$
- For each $i$ and $j$ transmission is erased with probability *0.1*

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Noisy Broadcast Model



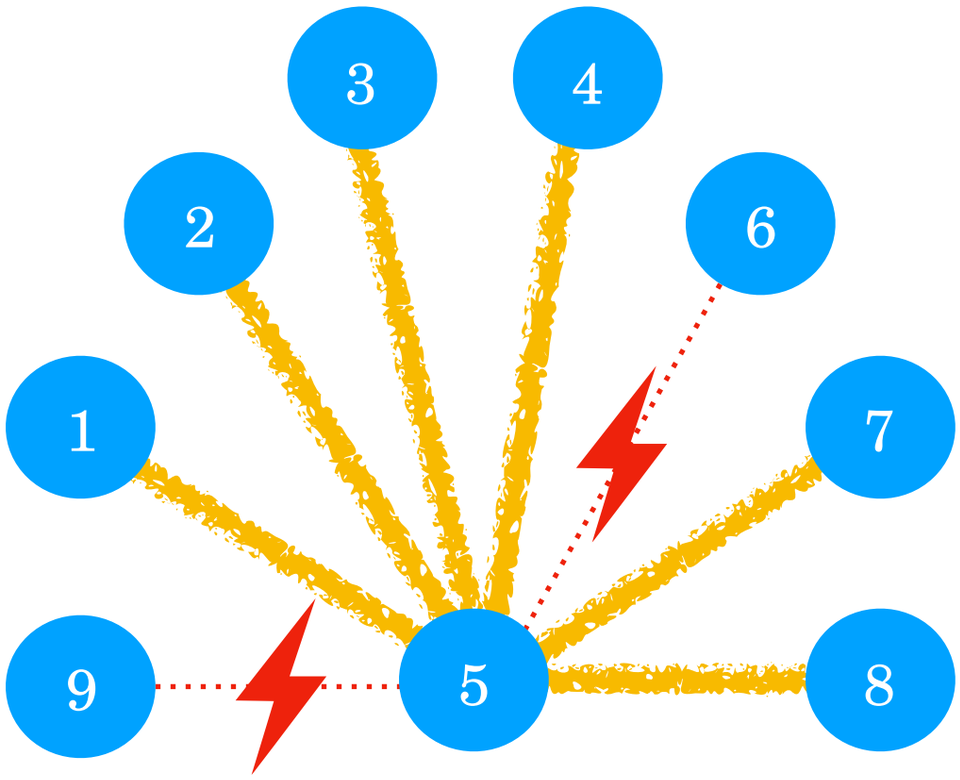

**View from processor 5**

- Each processor $i$ for $1 \le i \le n$ broadcasts a character in $\Sigma$ to all $j$
- For each $i$ and $j$ transmission is erased with probability *0.1*

***Erasure*** means transmission is replaced with ***?*** at random

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Noisy Broadcast Model




**View from processor 5**

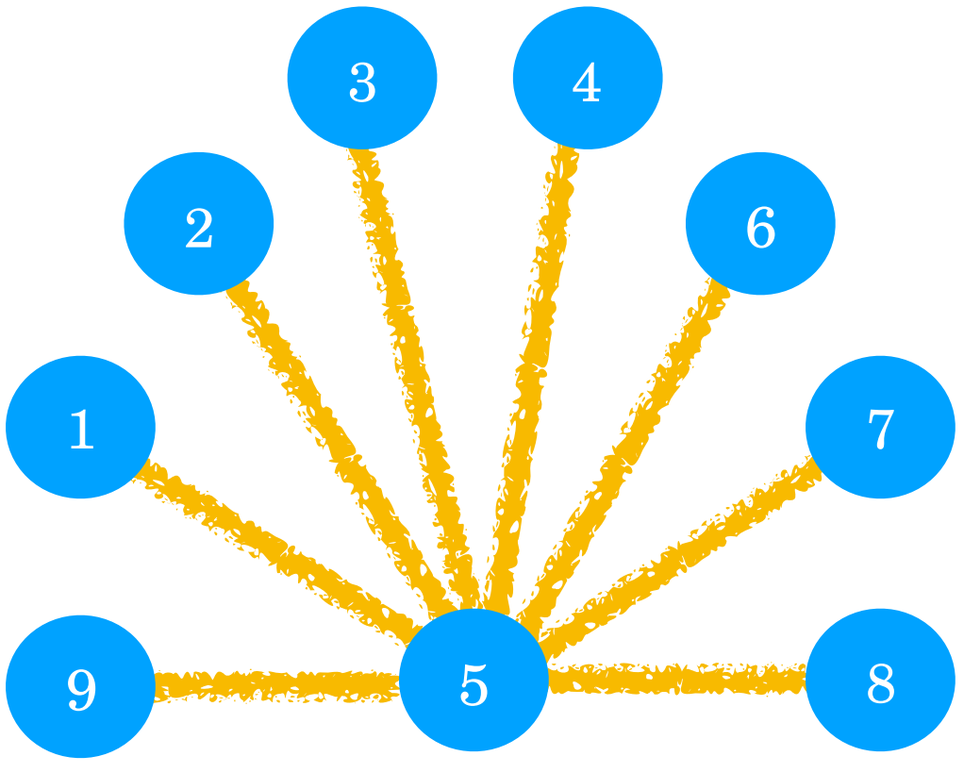
- Each processor $i$ for $1 \le i \le n$ broadcasts a character in $\Sigma$ to all $j$
- For each $i$ and $j$ transmission is erased with probability *0.1*

***Erasure*** means transmission is replaced with ***?*** at random

**Note:** less harsh than previously studied ***substitutions*** where transmission is replaced with a random character from $\Sigma$.

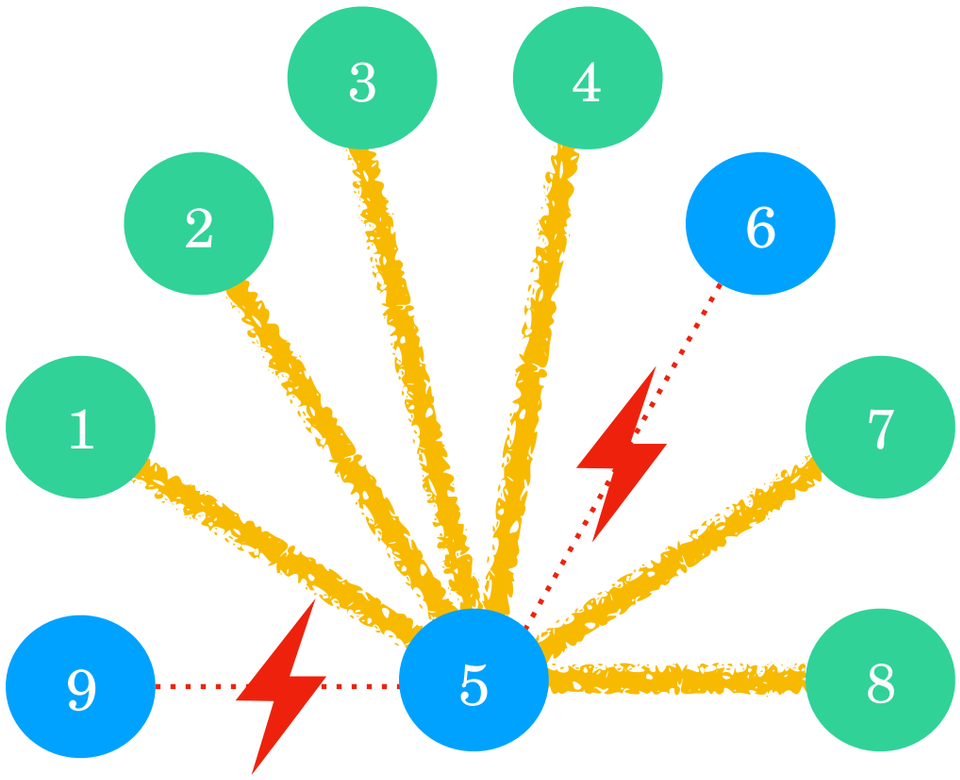
**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Trivial Attempt




- Processor $i$ repeatedly broadcasts $x_i$ for $2\log n$ rounds

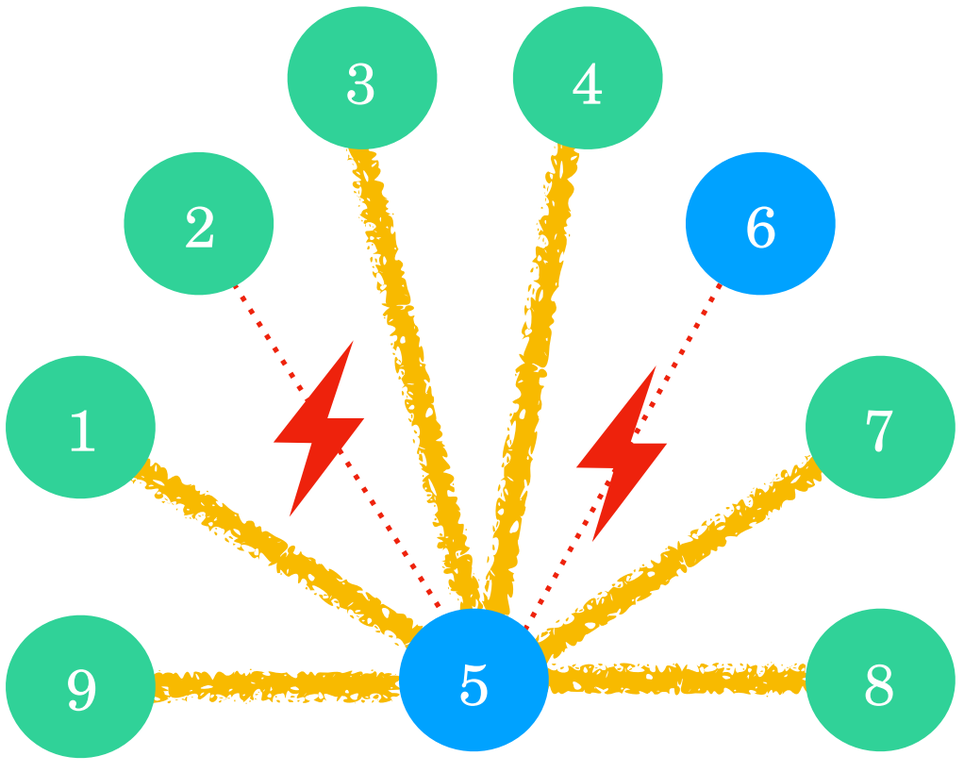
# Trivial Attempt




**Round 1**

- Processor $i$ repeatedly broadcasts $x_i$ for $2\log n$ rounds

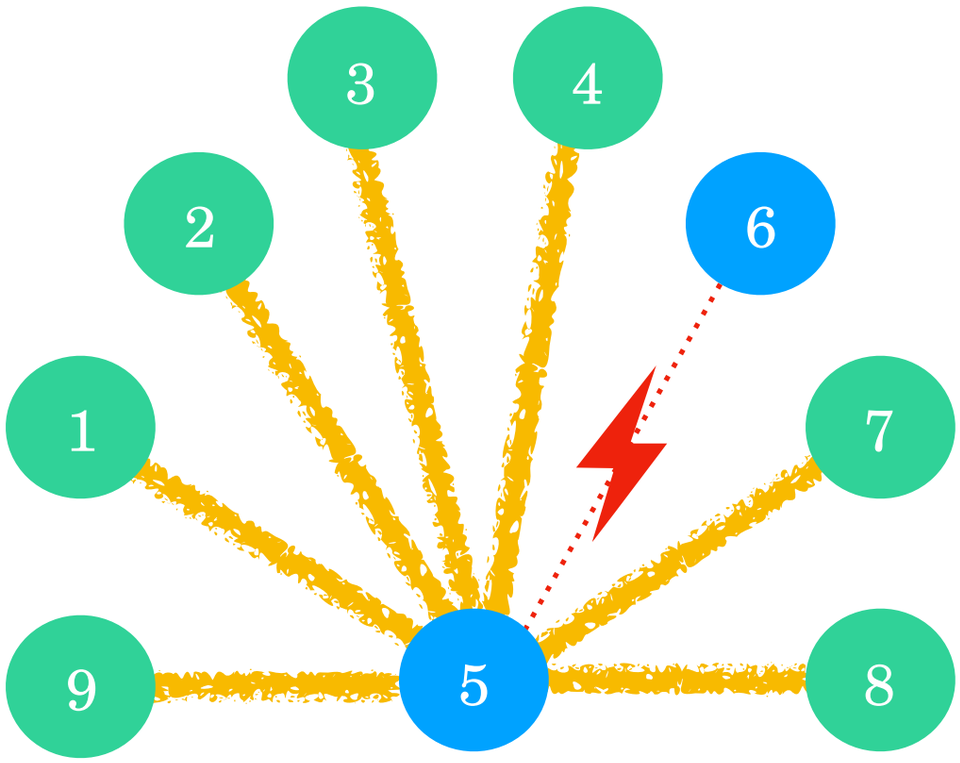
# Trivial Attempt




**Round 2**

- Processor $i$ repeatedly broadcasts $x_i$ for $2\log n$ rounds

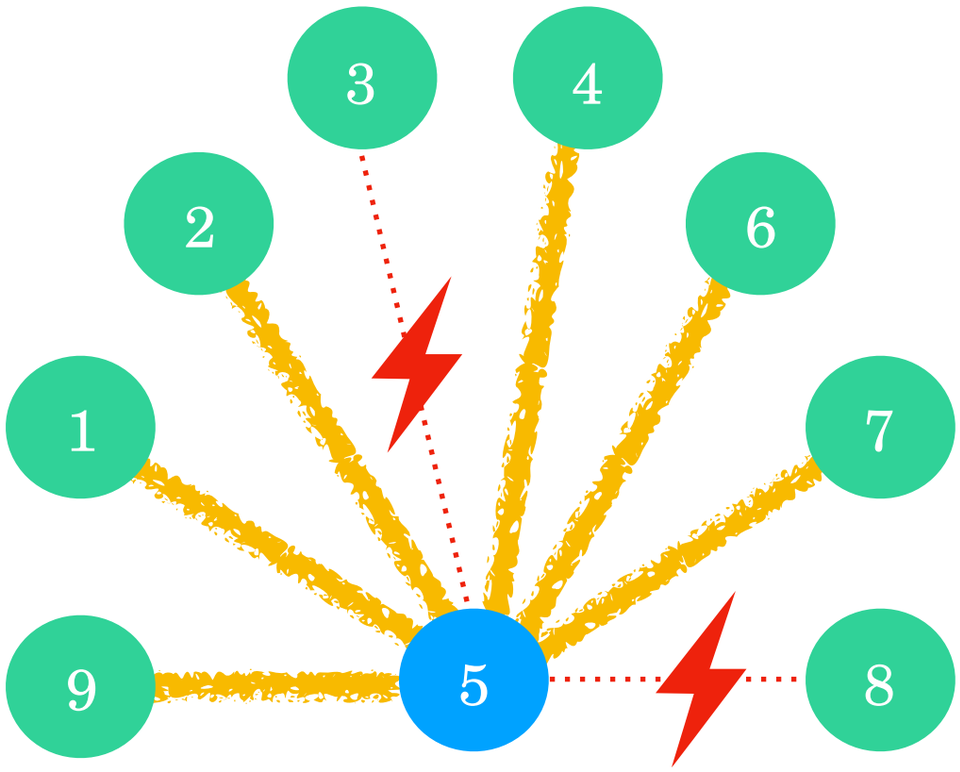
# Trivial Attempt




**Round 3**

- Processor $i$ repeatedly broadcasts $x_i$ for $2\log n$ rounds

# Trivial Attempt




**Round 4**

- Processor $i$ repeatedly broadcasts $x_i$ for $2\log n$ rounds

# Can we beat $O(\log n)$?

- Yes! [Gallagher'88] shows $O(\log \log n)$ rounds, even under substitutions

# Can we beat $O(\log \log n)$?

- Under ***substitutions***, no! [Goyal, Kindler, Saks'08] shows $\Omega(\log \log n)$ rounds, even under substitutions
- Under ***erasures***, yes! ***[this work]***

# Main Result

**Recap goal:** processors 1, ..., $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

$O(\log^* n)$ round algorithm for goal

$O(1)$ round algorithm when $|\Sigma| = \Omega(\text{poly}(n))$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Main Result

**Recap goal:** processors 1, ..., $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

$O(\log^* n)$ round algorithm for goal

**Focus of this talk!**

$O(1)$ round algorithm when $|\Sigma| = \Omega(\text{poly}(n))$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Main Result

**Recap goal:** processors 1, ..., $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

## $O(\log^* n)$ round algorithm guarantee:

- **SUCCESS** All processors output $x_1 \ldots x_n$ with probability $\geq 1 - n^{-5}$
- **FAILURE WITHOUT KNOWLEDGE** Some processor outputs string $X \neq x_1 \ldots x_n$ with probability $\leq 2^{-7n}$
- **FAILURE WITH KNOWLEDGE** With remaining probability all processors output *'failed'*

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Main Result

**Recap goal:** processors 1, ..., $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

## $O(\log^* n)$ round algorithm guarantee:

- **SUCCESS** All processors output $x_1 \ldots x_n$ with probability $\geq 1 - n^{-5}$
- **FAILURE WITHOUT KNOWLEDGE** Some processor outputs string $X \neq x_1 \ldots x_n$ with probability $\leq 2^{-7n}$
- **FAILURE WITH KNOWLEDGE** With remaining probability all processors output *'failed'*

## *Failure without knowledge* is catastrophic

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Main Algorithm Outline

**Recap goal:** processors 1, ..., $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

## $O(\log^* n)$ round algorithm outline:

A. **Learning Phase:** success probability of $1 - n^{-5}$ of all processors to learn string

B. **Validation Phase:** if A. failed, probability $1 - 2^{-7n}$ for all processors to detect failure

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Main Algorithm Outline

**Recap goal:** processors 1, ..., $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

## $O(\log^* n)$ round algorithm outline:

A. **Learning Phase:** success probability of $1 - n^{-5}$ of all processors to learn string

B. **Validation Phase:** if A. failed, probability $1 - 2^{-7n}$ for all processors to detect failure

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Learning Phase

## Base case:

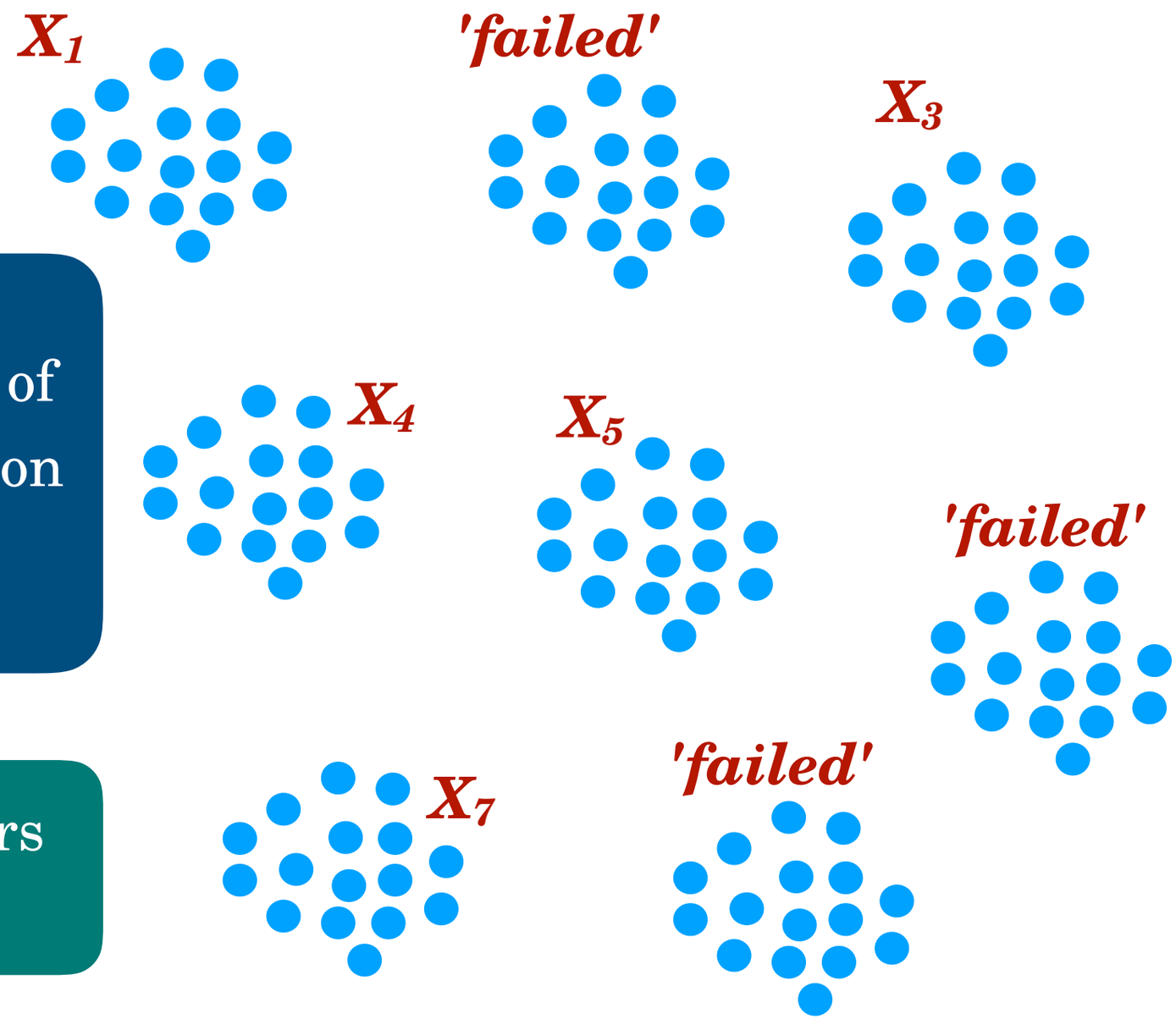
If $n \leq 10$, each processor repeatedly broadcasts 10 times

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Learning Phase: Recursion

Divide into $n / \log n$ groups of size $\log n$ each and recurse on each group

Roughly $n / \log^5 n$ processors part of failed group



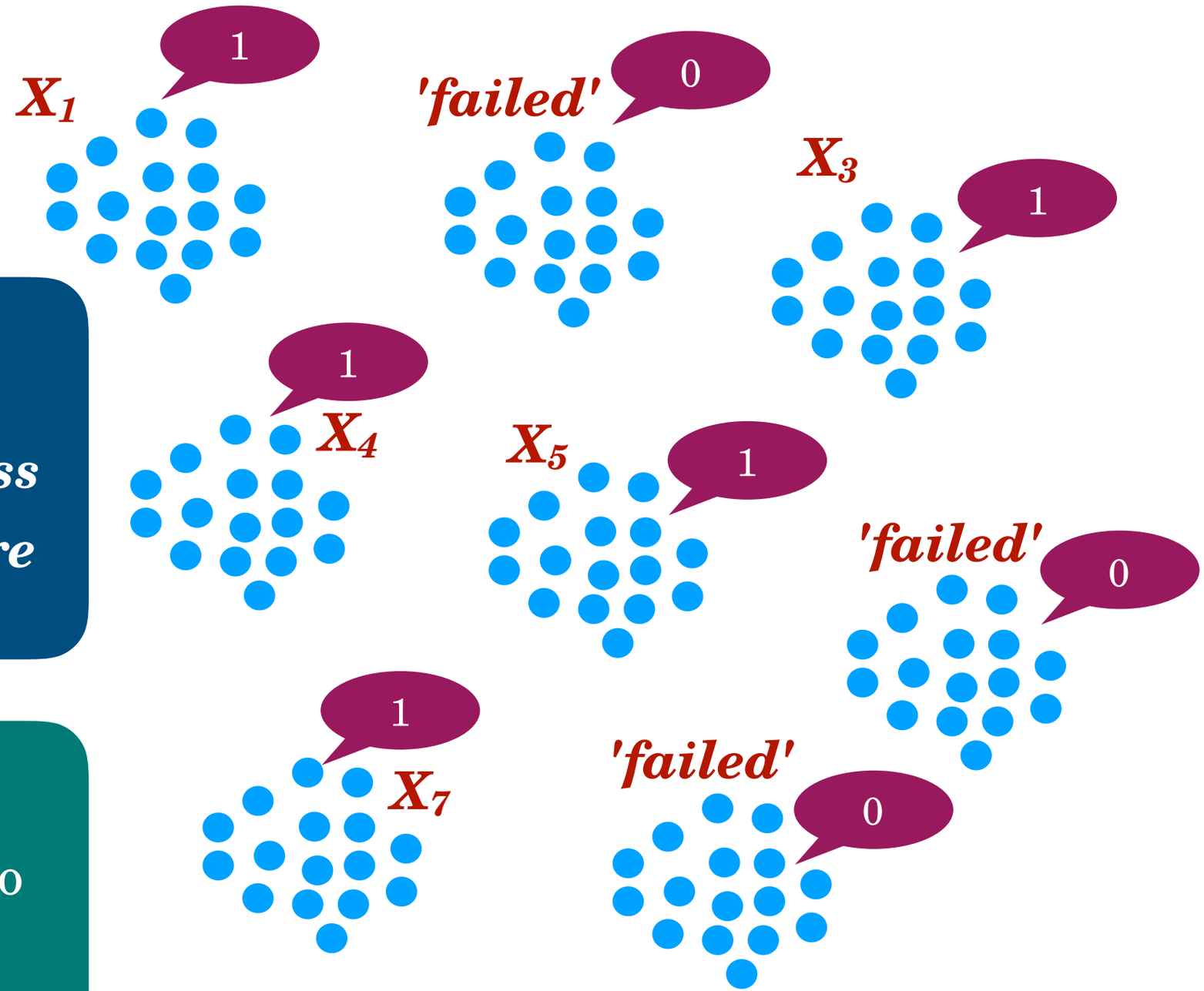

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Learning Phase: learning failures

Processor $i$ transmits

- 1 if recursion was ***success***
- 0 if recursion was ***failure***

Only need to receive one transmission from group to know if they failed or succeeded!



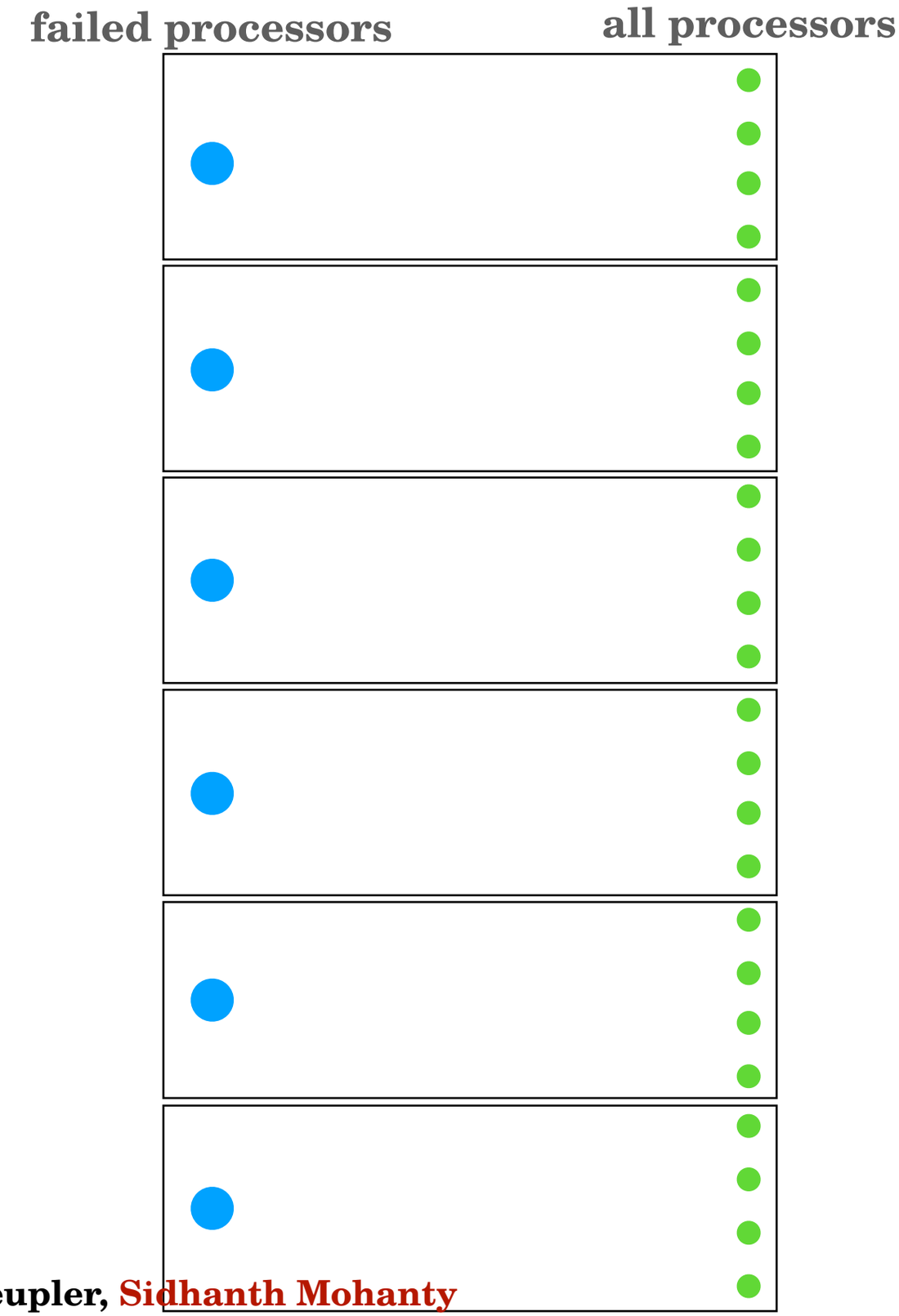

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Learning Phase: correcting failures

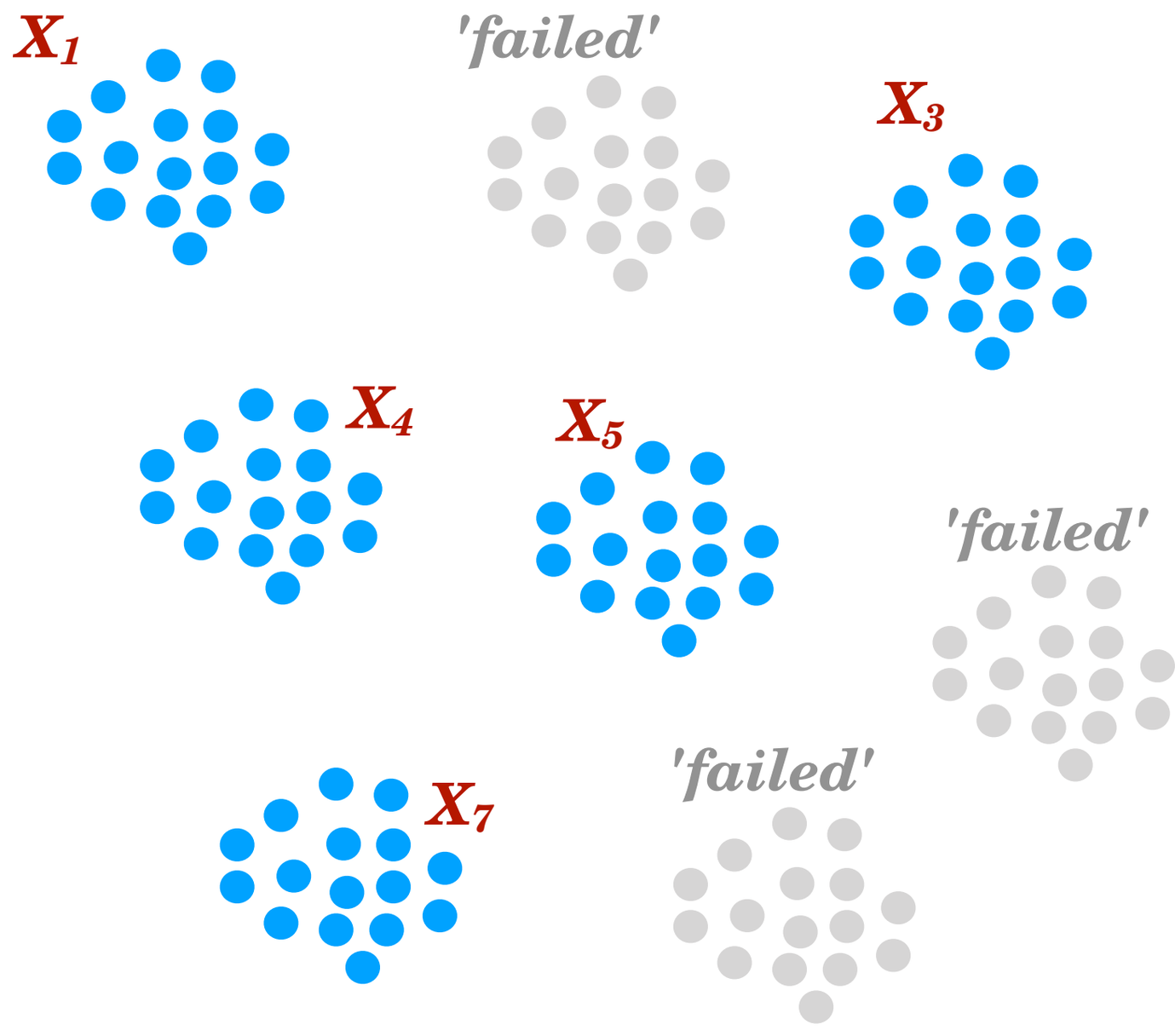
$i \log^2 n + 1$ to $(i+1) \log^2 n$
are ***responsible*** for
the $i$-th failed processor

- Each failed processor $x$ broadcasts its input bit
- Each processor *responsible* for $x$ broadcasts bit received by $x$

Processor $i$ can infer input of $x$ if it receives at least one transmission from processor responsible for $x$



# Learning Phase: success amplification

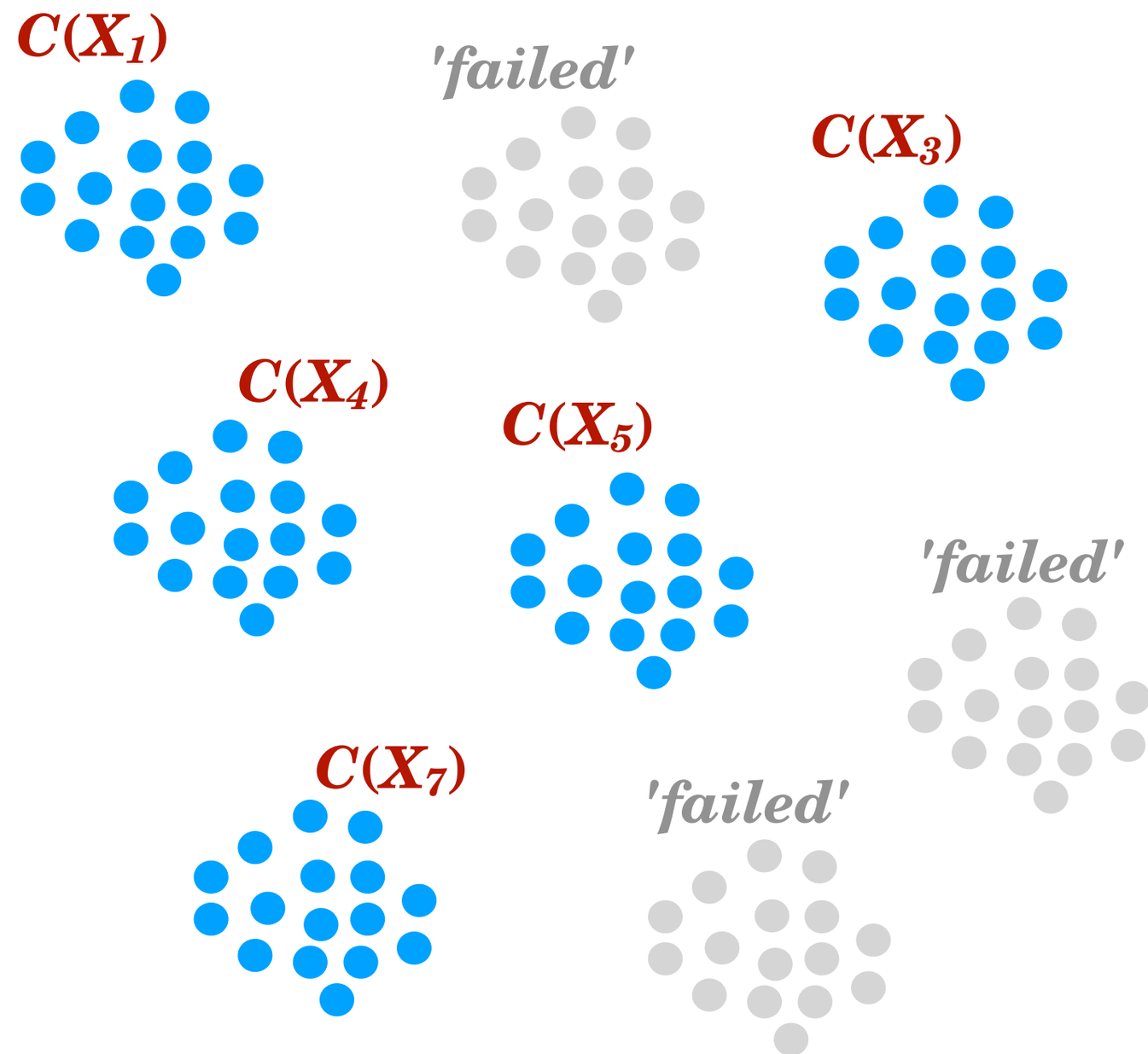



**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Learning Phase: success amplification

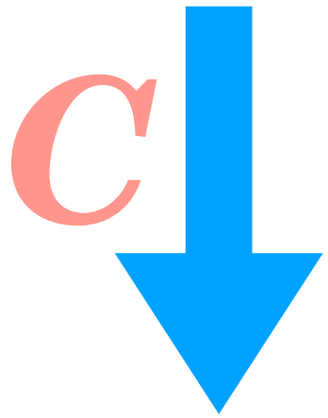
***C***: code with decoding radius 0.25 and rate constant ***K***

Processor from group $t$ broadcasts constant sized chunk of $C(X_t)$ in next $1/K$ rounds




**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

$$X_i = a_1 \dots a_n$$




$$C(X_i) = L_1 \dots L_n$$

Each $L_j$ has length $1/K$

Broadcast $L_j$ in next $1/K$ rounds

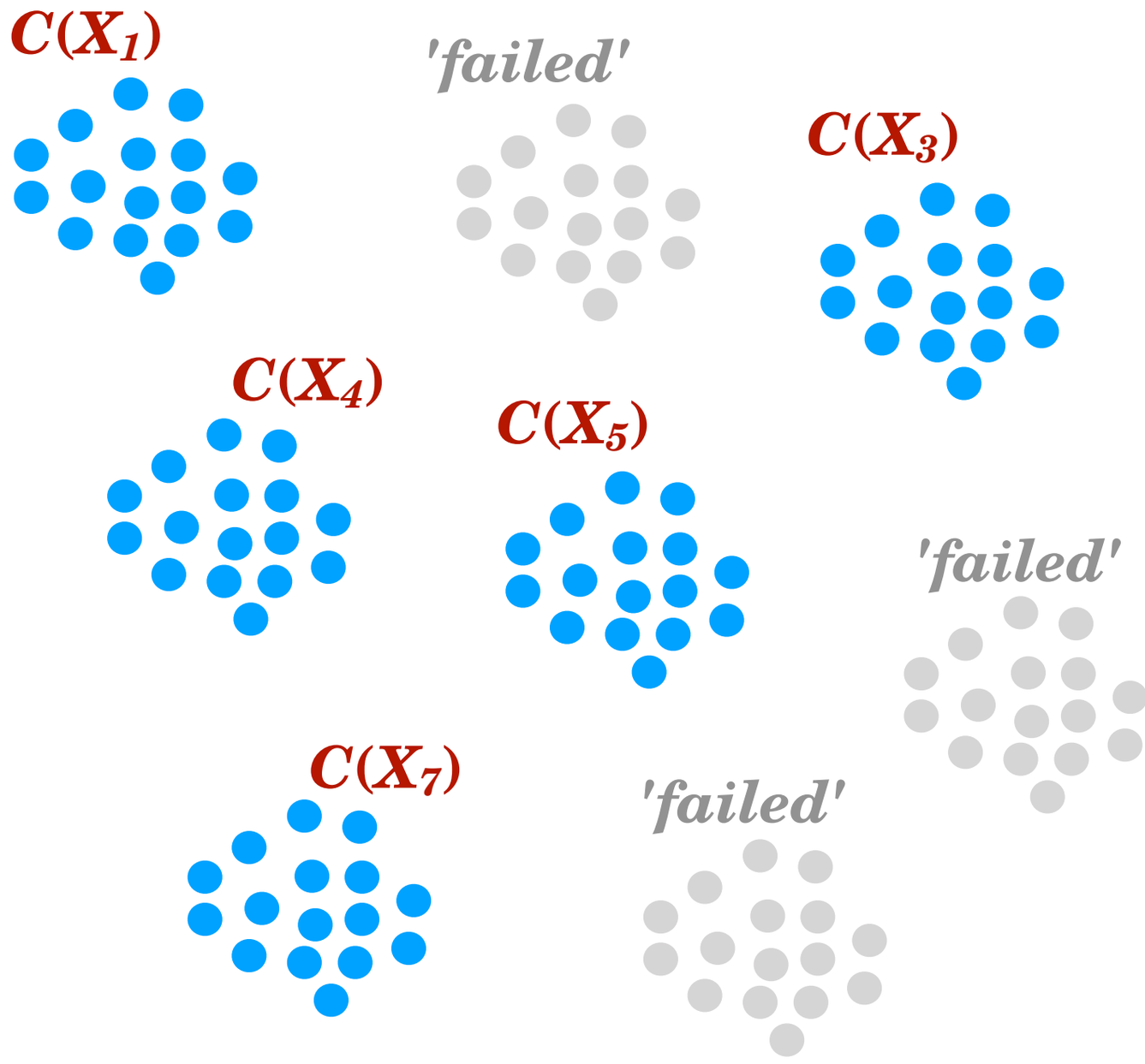
**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Learning Phase: success amplification

$C$: code with decoding radius 0.25 and rate constant $K$

Processor from group $t$ broadcasts constant sized chunk of $C(X_t)$ in next $1/K$ rounds

Processor $i$ can reconstruct $X_t$ as long as it receives enough bits from group $t$




**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Main Algorithm Outline

**Recap goal:** processors 1, ..., $n$ should learn $x_1 x_2 \ldots x_n$ with high probability

## $O(\log^* n)$ round algorithm outline:

A. **Learning Phase:** success probability of $1 - n^{-5}$ of all processors to learn string

B. **Validation Phase:** if A. failed, probability $1 - 2^{-7n}$ for all processors to detect failure

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Technical Ingredients

**Ingredient 1:** Algorithm to compute the **AND** of $x_1 \ldots x_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

Processor $i$ has string $\boldsymbol{X_i}$ as input

**Ingredient 2:** Algorithm to check if $\boldsymbol{X_1 = X_2 = \ldots = X_n}$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Technical Ingredients

**Ingredient 1:** Algorithm to compute the **AND** of $x_1 \ldots x_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

Processor $i$ has string $\boldsymbol{X_i}$ as input

**Ingredient 2:** Algorithm to check if $\boldsymbol{X_1 = X_2 = \ldots = X_n}$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

**Ingredient 1:** Algorithm to compute the **AND** of $x_1 \ldots x_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

## Each processor $i$ executes:

1. Broadcast $x_i$
2. Broadcast the **AND** of all received bits from round 1
3. Output the **AND** of all received bits from round 2

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Technical Ingredients

**Ingredient 1:** Algorithm to compute the **AND** of $x_1 \ldots x_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

Processor $i$ has string $\boldsymbol{X_i}$ as input

**Ingredient 2:** Algorithm to check if $\boldsymbol{X_1 = X_2 = \ldots = X_n}$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

Processor $i$ has string $X_i$ as input

**Ingredient 2:** Algorithm to check if $X_1 = X_2 = \ldots = X_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

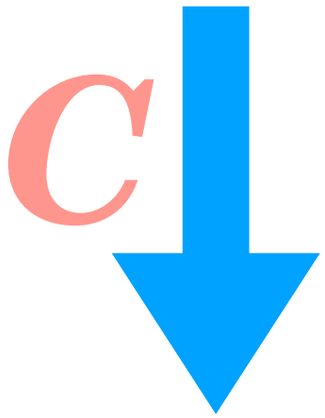
$C$: code with decoding radius 0.25 and rate constant $K$

## Outline of processor $i$ execution:

1. Encode $X_i$ with $C$ and split into chunks
2. Broadcast assigned chunk
3. Decode received string
4. Use **ingredient 1** to check if decoded string matches $X_i$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

$$X_i = a_1 \dots a_n$$




$$C(X_i) = L_1 \dots L_n$$

Each $L_j$ has length $1/K$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

Processor $i$ has string $X_i$ as input

**Ingredient 2:** Algorithm to check if $X_1 = X_2 = \ldots = X_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

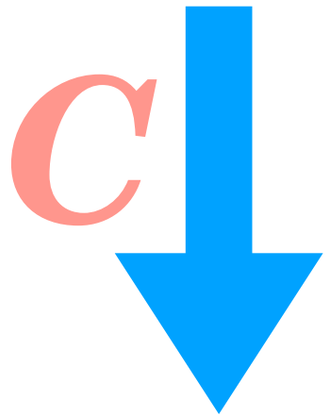
$C$: code with decoding radius 0.25 and rate constant $K$

## Outline of processor $i$ execution:

1. Encode $X_i$ with $C$ and split into chunks
2. Broadcast assigned chunk
3. Decode received string
4. Use **ingredient 1** to check if decoded string matches $X_i$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

$$X_i = a_1 \dots a_n$$




$$C(X_i) = L_1 \dots L_n$$

Each $L_j$ has length $1/K$

Broadcast $L_j$ in next $1/K$ rounds

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

Processor $i$ has string $X_i$ as input

**Ingredient 2:** Algorithm to check if $X_1 = X_2 = \ldots = X_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

$C$: code with decoding radius 0.25 and rate constant $K$

## Outline of processor $i$ execution:

1. Encode $X_i$ with $C$ and split into chunks
2. Broadcast assigned chunk
3. Decode received string
4. Use **ingredient 1** to check if decoded string matches $X_i$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

Processor $i$ receives $n/K$ bit string

$$S = S_1 \ldots S_n$$

with each $S_j$ length $1/K$

Obtain codeword $T$ within decoding radius of $S$ and let $Y_i$ be $C^{-1}(T)$, the 'decoded string'

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

Processor $i$ has string $X_i$ as input

**Ingredient 2:** Algorithm to check if $X_1 = X_2 = \ldots = X_n$ in $O(1)$ rounds with $\leq 2^{-\Omega(n)}$ failure probability

$C$: code with decoding radius 0.25 and rate constant $K$

## Outline of processor $i$ execution:

1. Encode $X_i$ with $C$ and split into chunks
2. Broadcast assigned chunk
3. Decode received string
4. Use **ingredient 1** to check if decoded string matches $X_i$ for all $i$

**Ofer Grossman, Bernhard Haeupler, Sidhanth Mohanty**

# Conclusions

- $\Omega(\log n)$ lower bound in ***substitution model*** on computing **AND** of $n$ bits with probability $\exp(-\Omega(n))$ [Goyal-Kindler-Saks'08]

- Same problem has $O(1)$ complexity in ***erasures model*** and makes $O(\log^* n)$ algorithm possible

- Can one show an $\Omega(\log^* n)$ lower bound?