Deterministic Document Exchange Protocols, and Almost Optimal Binary Codes for Edit Errors

Kuan Cheng, **Zhengzhong Jin**, Xin Li, Ke Wu

JOHNS HOPKINS UNIVERSITY

Background

- Edit Distance:

$$x = \mathrm{a}\ \cancel{\mathrm{b}}\ \mathrm{b}\ \mathrm{a}\ \mathrm{a}\ \mathrm{b}\ \mathrm{a}\ \cancel{\mathrm{a}}\ \mathrm{b}\ \mathrm{a}$$

a

$$y = \mathrm{a\ b\ a\ a\ a\ b\ a\ b\ a}$$

$ED(x, y) \stackrel{\text{def}}{=}$ Minimum number of insertions/deletions transforming x to y.

- Hamming Distance:

$$x = 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0$$

$$y = 0\ 1\ 1\ \boxed{1}\ 0\ 1\ \boxed{1}\ 0\ 1\ 0$$

Number of different bits in corresponding indexes

Document Exchange Protocol

Alice

Bob

$x \in \{0,1\}^n$

$ED(x,y) \le k$

$y \in \{0,1\}^{n'}$

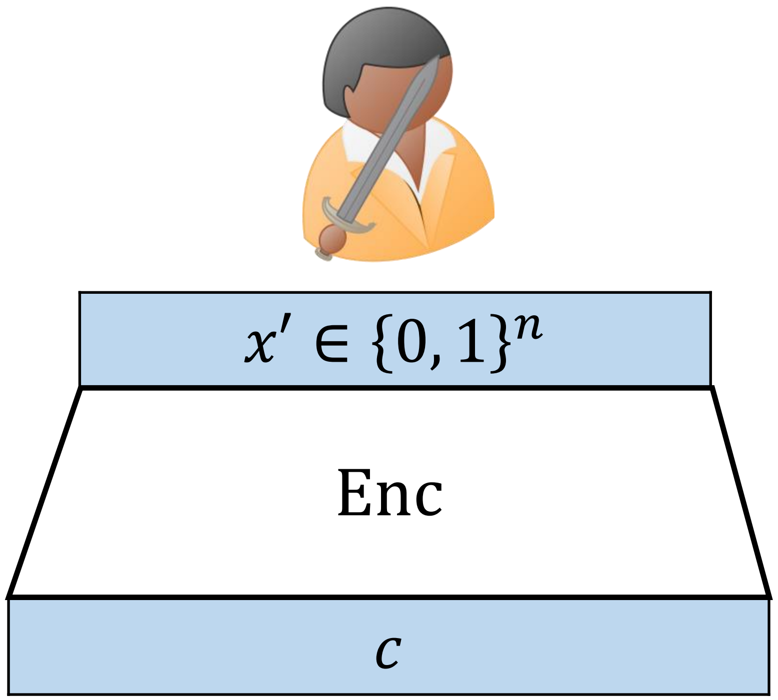
Sketch

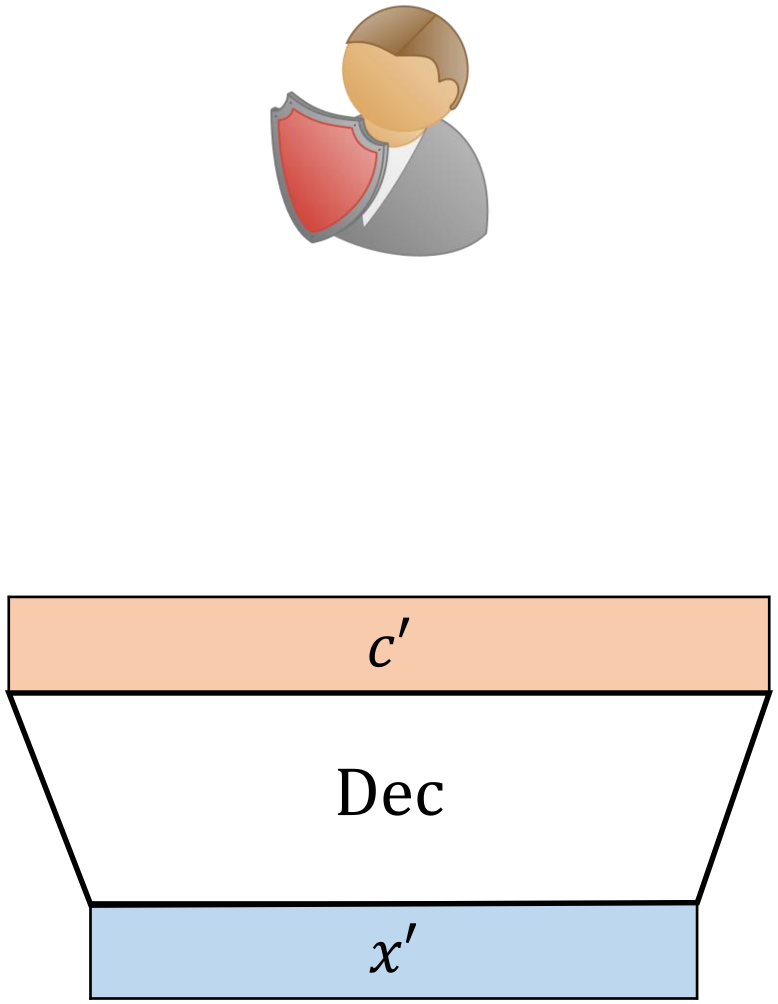
$sk \longrightarrow \underbrace{sk}$

$x = \mathrm{Recover}(sk, y)$

Minimize Sketch Size

Error Correcting Code





- Minimize Redundancy $= |c| - |x'|$
- Maximize Information Rate $= |x'|/|c|$

Previous Results For Hamming Errors

- Document Exchange from Systematic Error Correting Code

sketch

$$\mathrm{Enc}(x) = \boxed{\quad x \quad}\boxed{\text{Redundancy}}$$

	Reed-Solomon Code	Algebraic Geometry Code (Optimal)
Redundancy (sketch size)	$k \log n$	$\Theta\left(k \log \frac{n}{k}\right)$
Information Rate	$1 - \Theta(\epsilon \log n)$	$1 - \Theta\left(\epsilon \log \frac{1}{\epsilon}\right)$

- $\epsilon = k/n$

Results

	Previous	Our Results	Bounds
Error Correcting Code for Edit Distance:			
Redundancy	$O(k^2 \log k \log n)$ for $k = O(1)$ [BGZ17] $O(k^2 + k\log^2 n)$ [Bel15]	$O(k \log n)$ $O\left(k \log^2 \frac{n}{k}\right)$	$\Theta\left(k \log \frac{n}{k}\right)$
Information Rate	$1 - \tilde{O}(\sqrt{\epsilon})$ [GL16, GW17]	$1 - \tilde{O}(\epsilon)$	$1 - \Theta(\epsilon \log(\frac{1}{\epsilon}))$
Document Exchange:			
Sketch Size	$O(k^2 + k\log^2 n)$ [D] [Bel15] $O(k \log^2 \frac{n}{k})$ [R] [IMS05]	$O\left(k \log^2 \frac{n}{k}\right)$ [D]	$\Theta\left(k \log \frac{n}{k}\right)$

- $\epsilon = k/n$
- $O(k \log n) = O(k \log \frac{n}{k})$, when $k = n^{1-a}$, for any $a > 0$.

Overview: Almost optimal ECC for Edit Distance

Document exchange for a uniform random string

Derandomize the random string
using Pseduorandom Generator (PRG)

Error Correcting Code

Document Exchange for a Uniform Random String

Stage I:

- Alice partitions her string into blocks of size $\mathbf{poly}(\log n)$.
- Send $O(k \log n)$ bits to help Bob learn her partition.

Stage II:

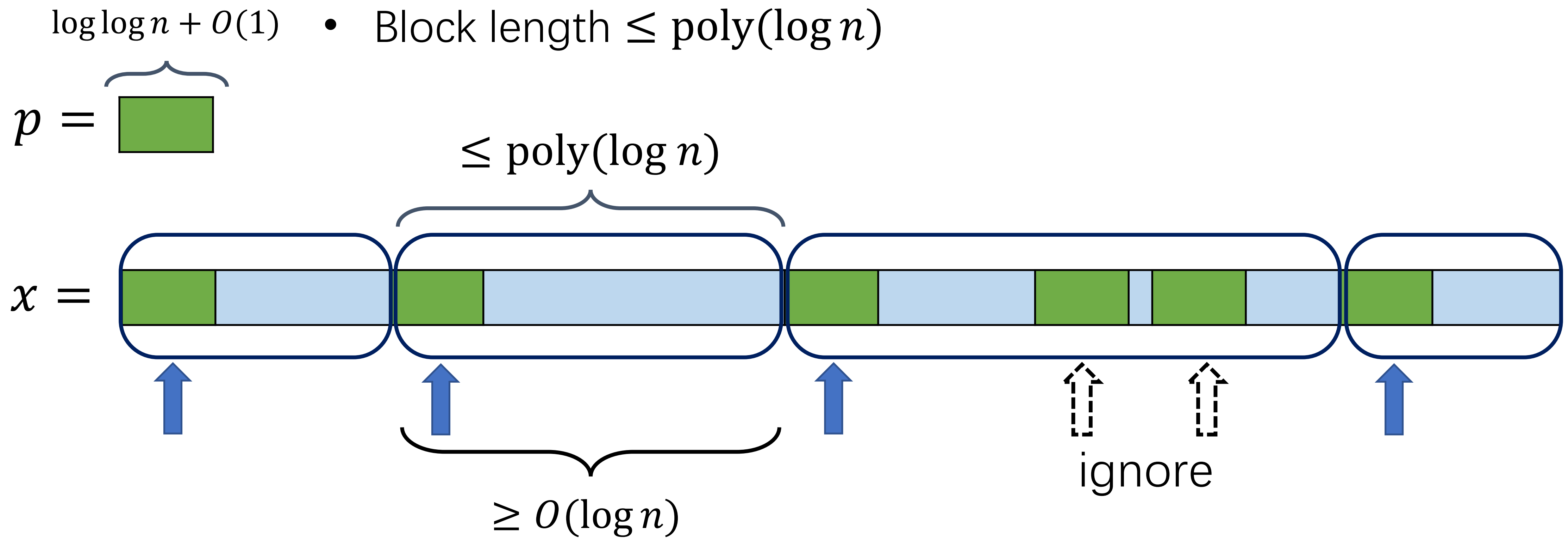
- $O(1)$ levels.
- In each level, divide each block into $O(\log^{0.4} n)$ smaller blocks.
- In each level, Alice sends $O(k \log n)$ bits to help Bob learn most blocks, except $O(k)$ of them are incorrect/unfilled.

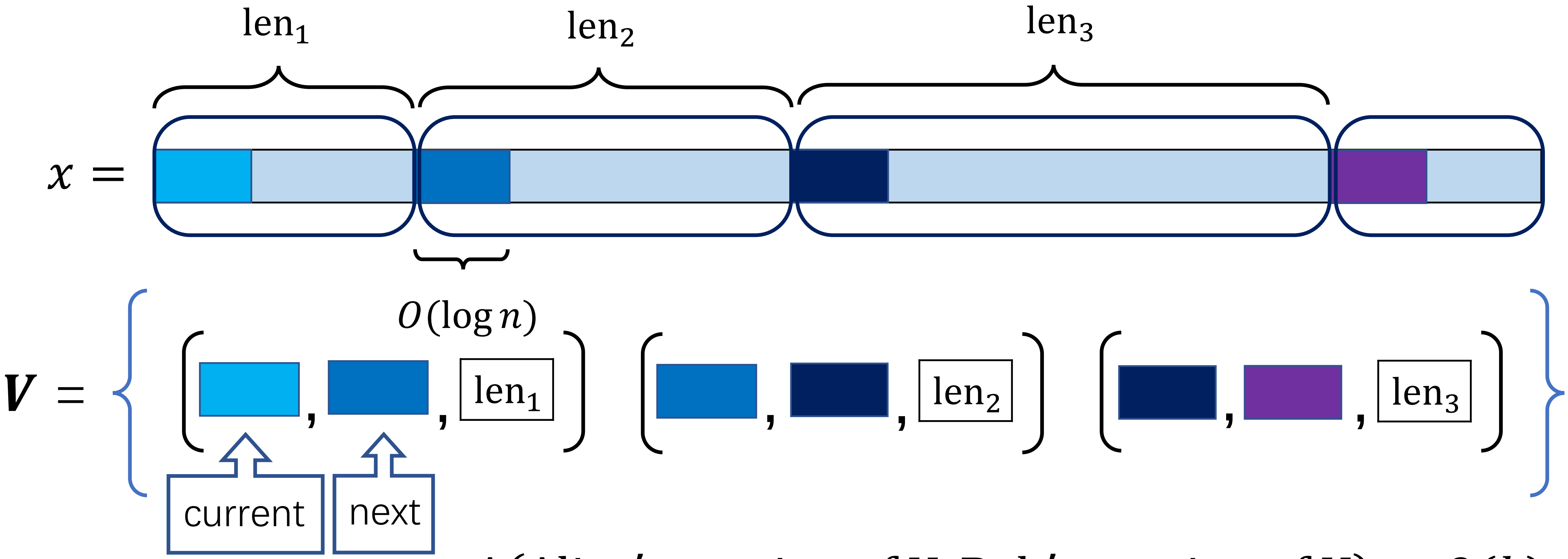
Stage I (Alice)

- Suppose x is uniformly random, use a fixed string p to partition x.

With probability $1 - 1/\mathrm{poly}(n)$,

- Any two substrings of length $O(\log n)$ are distinct.
- Block length $\leq \mathrm{poly}(\log n)$

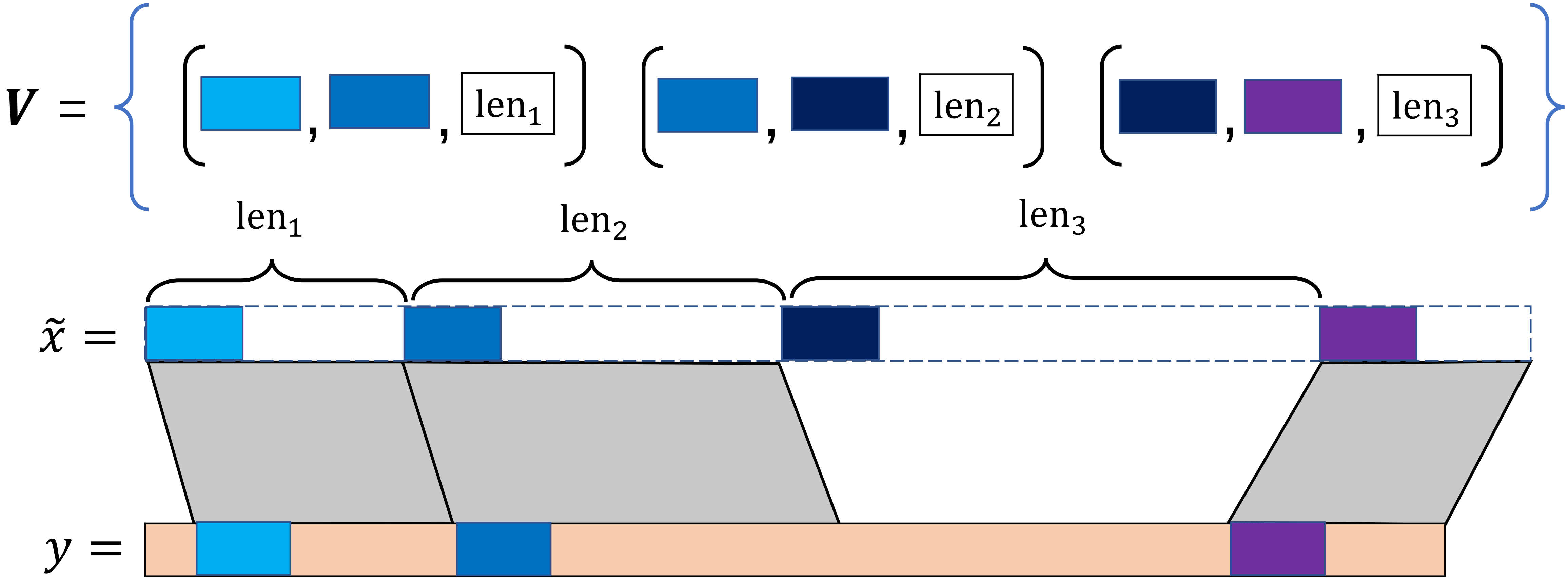




$\Delta(\text{Alice's version of } V, \text{Bob's version of } V) \leq O(k)$

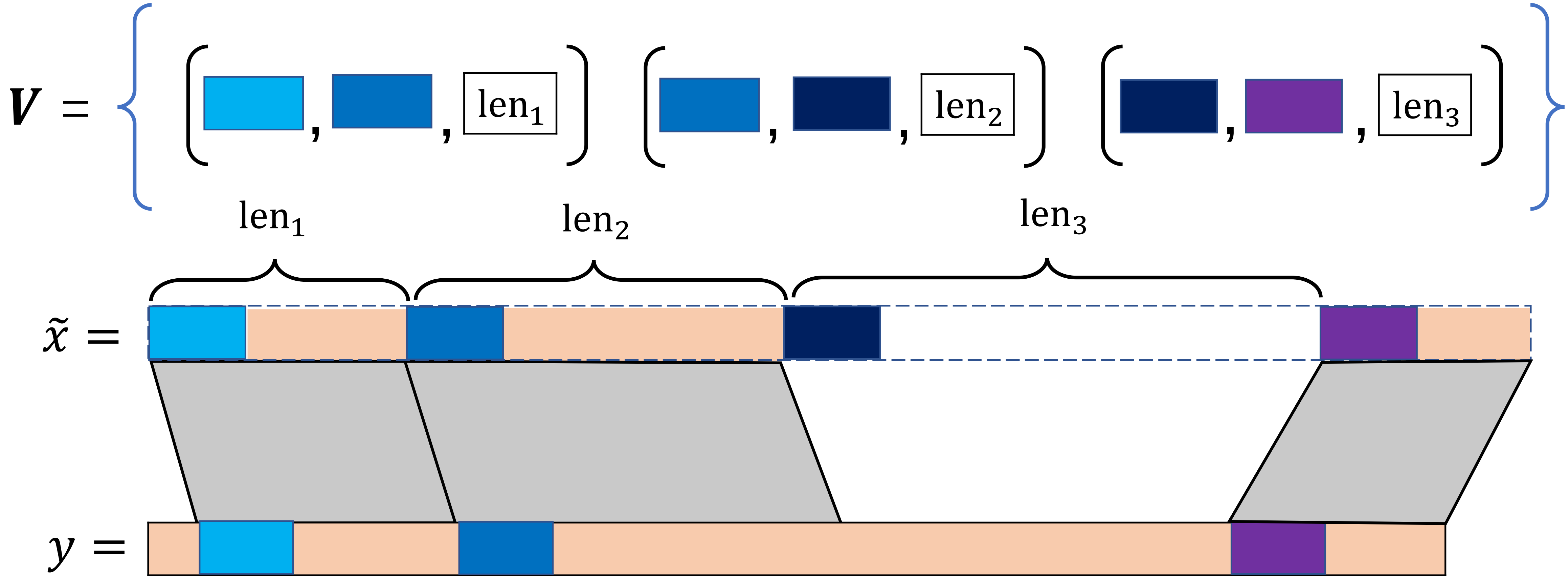
- Using Reed-Solomon Code, Alice computes a redundancy of $O(k \log n)$ bits to help Bob correct $\boldsymbol{V}$ from $O(k)$ Hamming Errors.

Stage I (Bob)



- Match the string y using the $O(\log n)$-prefixes, then fill $\tilde{x}$ using y.

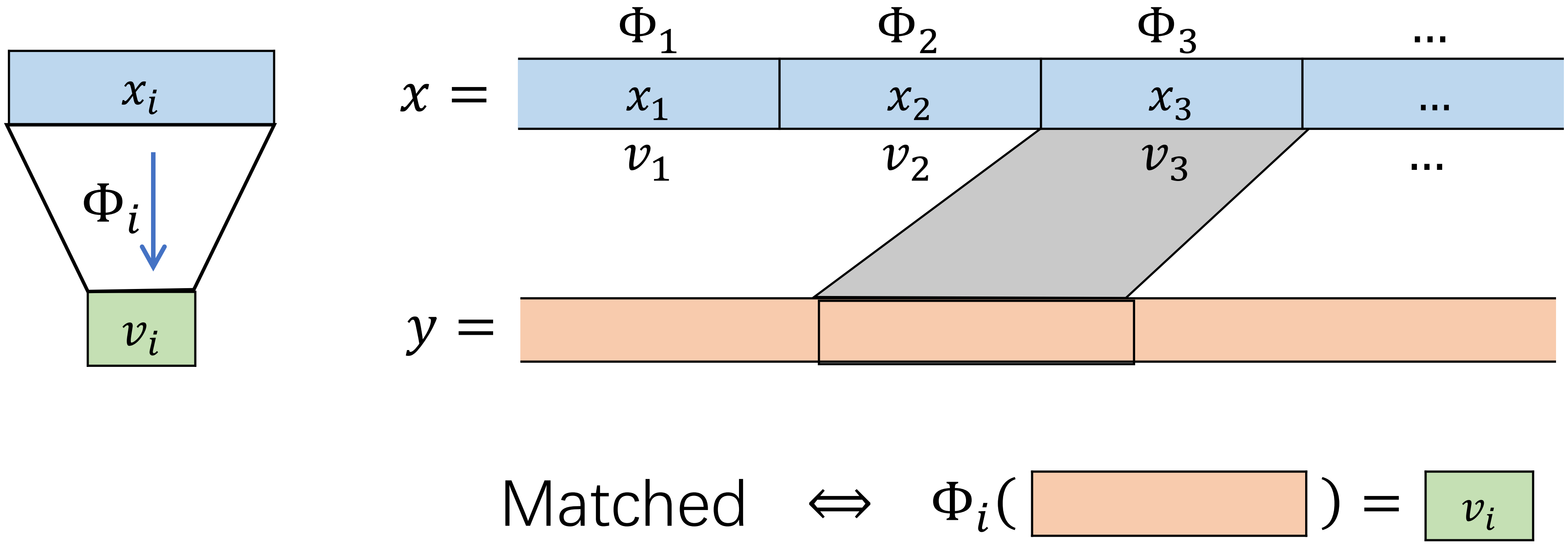
Stage I (Bob)



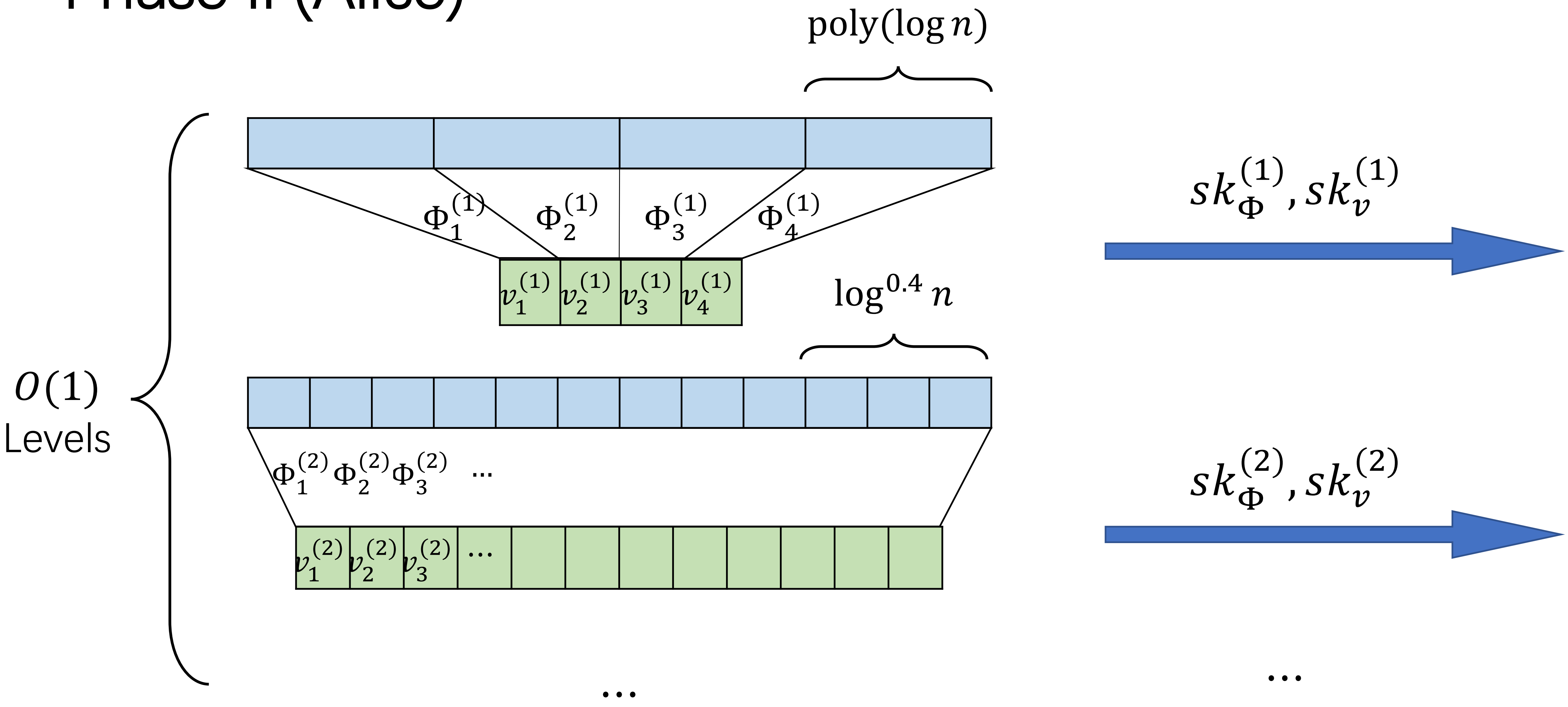
- Match the string y using the $O(\log n)$-prefixes, then fill $\tilde{x}$ using y.

Matching under Hash Functions

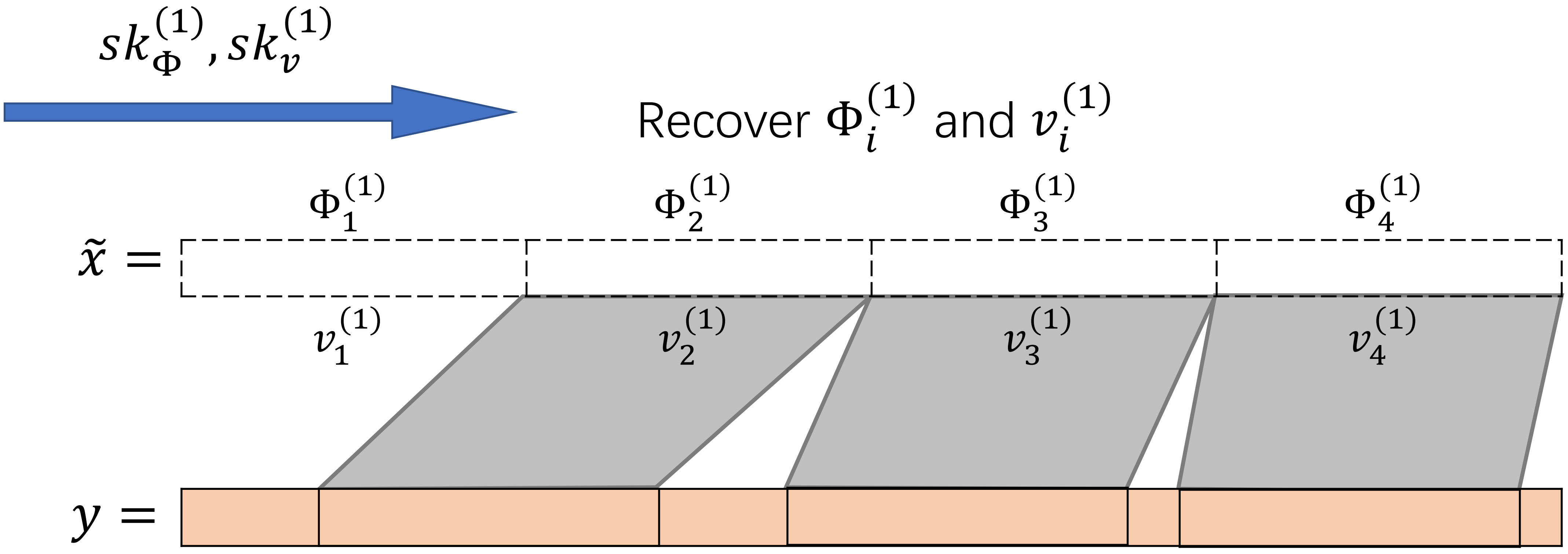
- Match between x and y under hash functions
- Block Size = T, Φ_i are hash functions.



Phase II (Alice)

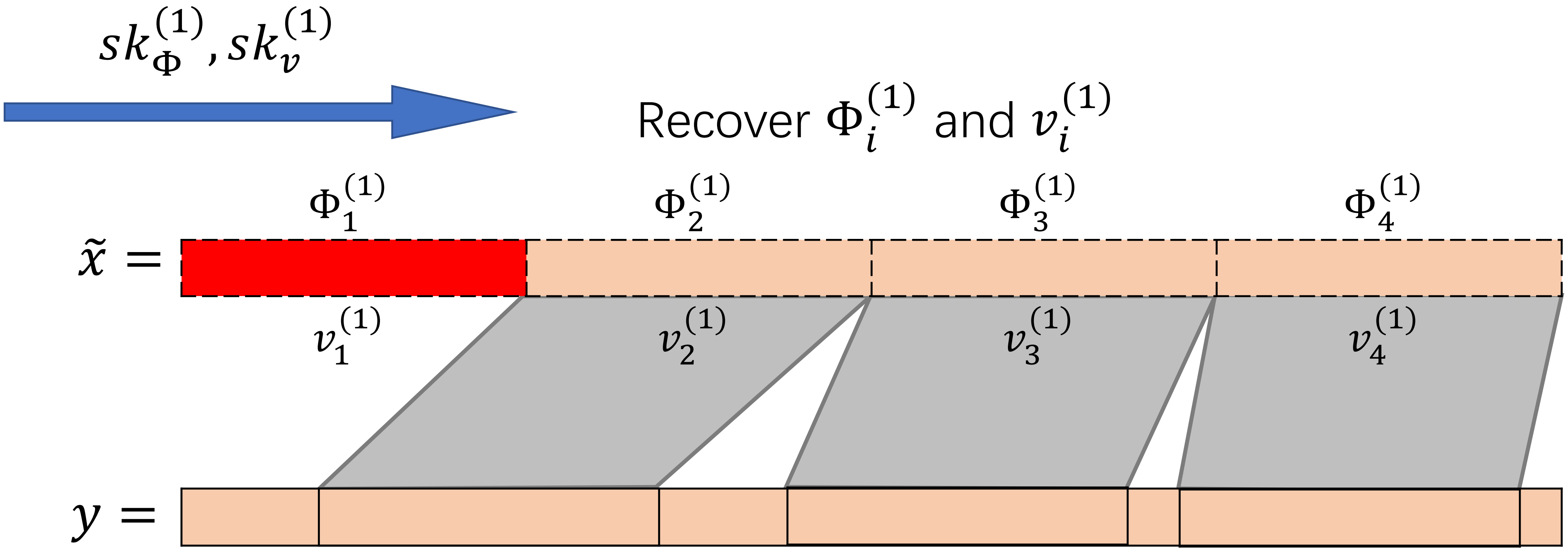


Phase II (Bob)



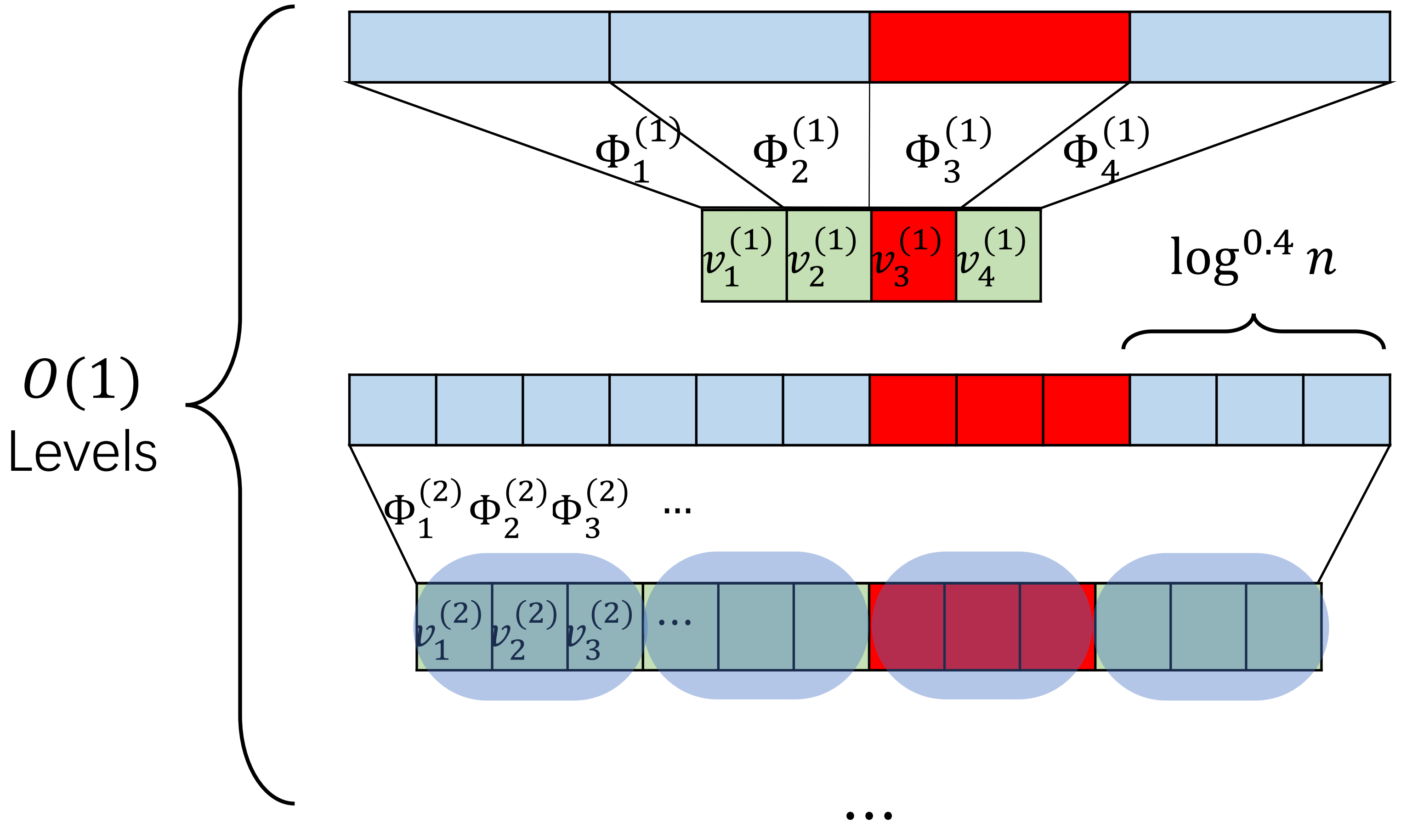
- Find the maximum monotone matching between x and y under Φ_i.
- Fill $\tilde{x}$ using the matching. Objective: #incorrect / unmatched blocks $\leq O(k)$

Phase II (Bob)



- Find the maximum monotone matching between x and y under Φ_i.
- Fill $\tilde{x}$ using the matching. Objective: #incorrect / unmatched blocks $\leq O(k)$

Phase II : Packing



$O(k)$ incorrect/unfilled blocks

~~$O(k \log^{0.4} n)$ incorrect/unfilled blocks~~

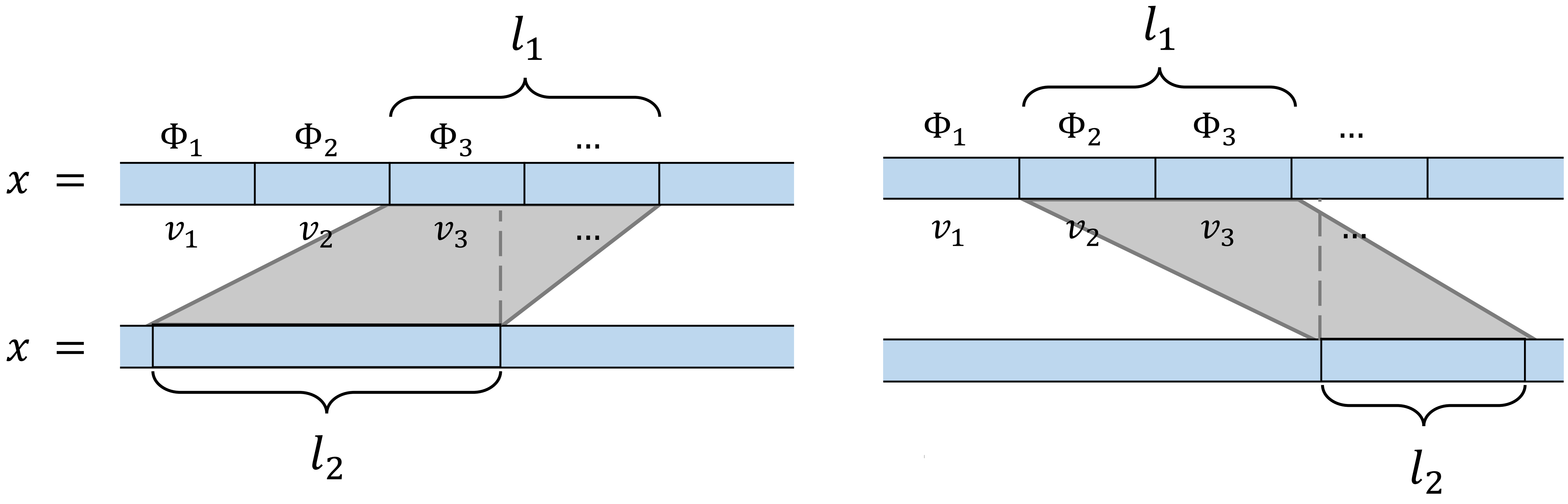
$O(k)$ incorrect/unfilled packages

- Pack every $O(\log^{0.4} n)$ successive hash values.

- Minimize the package size (or hash value size).

ϵ-Synchronization Hash Functions: Definition

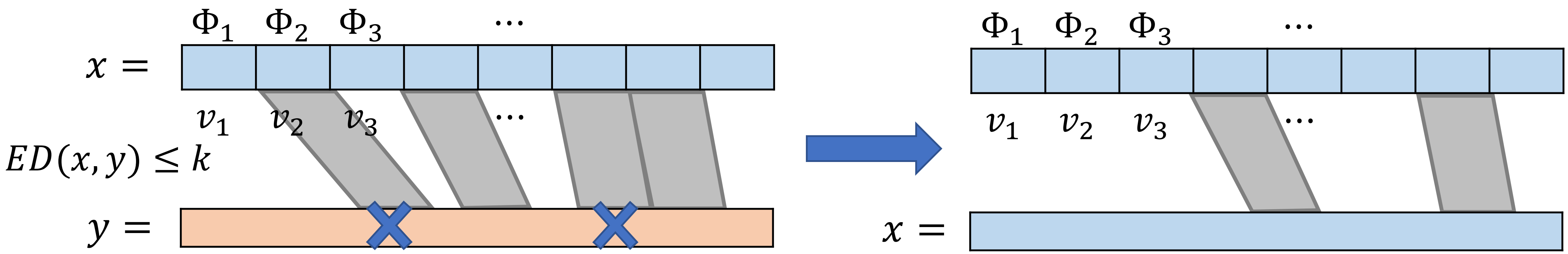
- 'Skew-Shape': Two adjacent intervals.



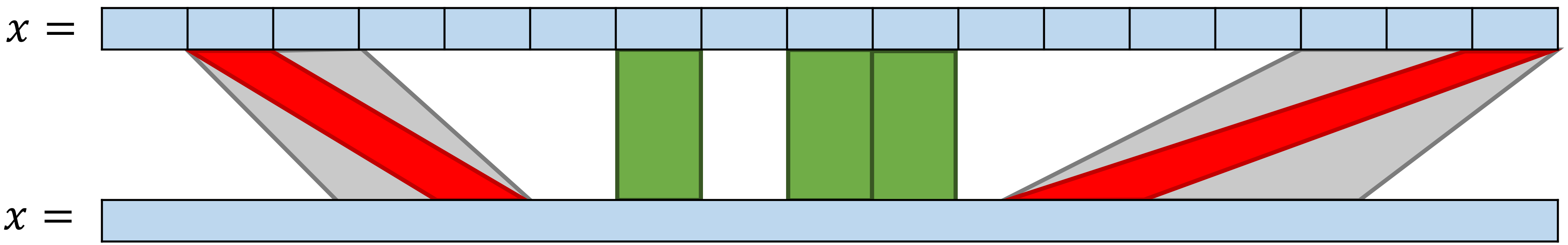
- $\forall$ 'Skew-Shape', $\#\mathrm{MATCHING} \leq \epsilon\left(l_1 + \frac{l_2}{T}\right)$, ($T$ = block size).

ϵ-Synchronization Hash Functions: Properties

- Matching between x and y induces a 'self-matching' on x itself.



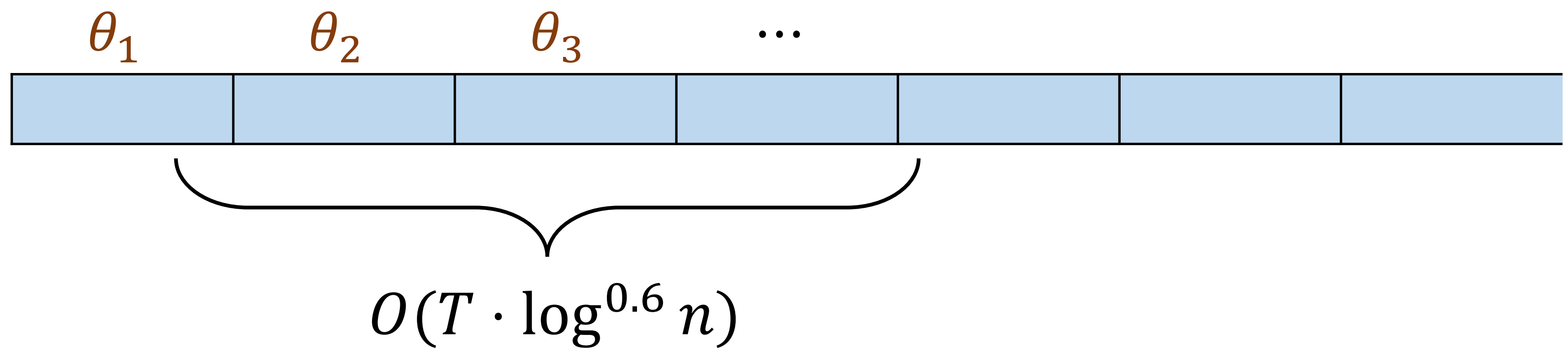
- The wrong self-matchings can be grouped into 'Z-Shapes'



- The two properties keeps #incorrect/unfilled blocks $\le O(k)$, while minimize the hash value size.

ϵ-Synchronization Hash Functions: Construction

- For large intervals ($l_1 + l_2/T \geq \log^{0.6} n$): use uniform random functions ϕ_i (Probability Method).
- For small intervals: Use ‘locally injective’ hash functions θ_i.



- $\Phi_i(\square) = \left(\phi_i(\square),\ \theta_i(\square)\right),\ \forall\, i$

Pseudorandom Generator (PRG)

- A generator $g\colon \{0,1\}^r \to \{0,1\}^n$ is a PRG against a function $f\colon \{0,1\}^n \to \{0,1\}$ with error ϵ, if

$$|\Pr[f(U_n) = 1] - \Pr[f(g(U_r)) = 1]| \leq \epsilon$$

- ϵ-almost κ-wise independence generator: An explicit construction of $g\colon \{0,1\}^d \to \{0,1\}^n$, $(X_1, X_2, \dots, X_n) = g(U_r)$, s.t.

$$\forall i_1, i_2, \dots, i_\kappa \in [n], \qquad \left|\Pr\left[X_{i_1}, X_{i_2}, \dots, X_{i_\kappa} = x\right] - 2^{-\kappa}\right| \leq \epsilon$$

ϵ-Synchronization Hash Functions: Derandomization

Use almost κ-wise independence generator:

- A large fraction of random seed satisfies the properties we want.
- Use exhaustive search to find the appropriate random seed.
- For ϕ_i, random seed size $= O(\log n)$. Send it directly.
- For θ_i, random seed size $= O(\log\log n)$ bits, packing them and send a redundancy in $O(k\log n)$ bits.

Error Correcting Codes from Sketch

$$x'$$

$$\oplus$$

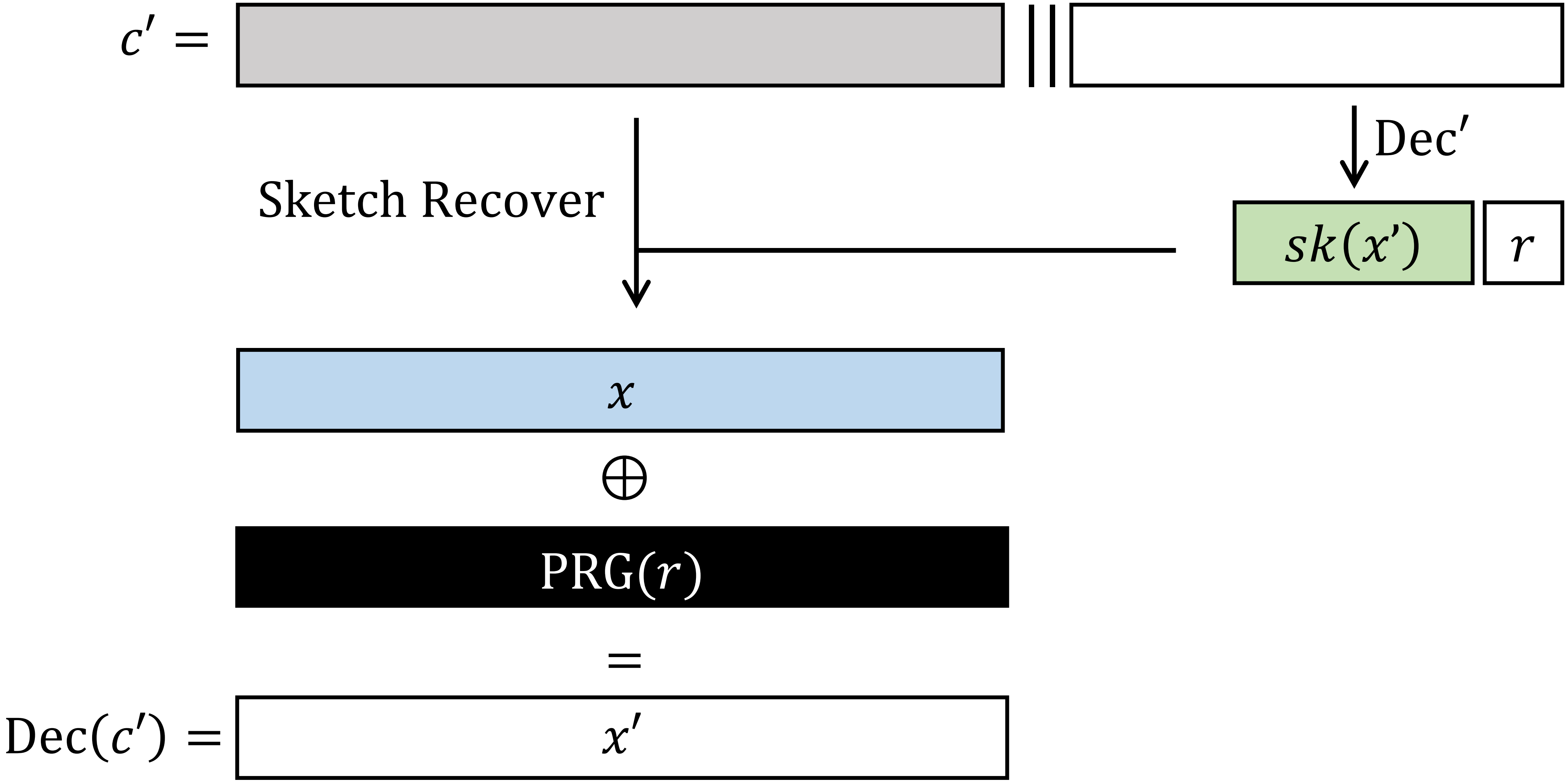
$$\mathrm{PRG}(r)$$

$$=$$

$$\mathrm{Enc}(x) \stackrel{\text{def}}{=} \boxed{x} \;\|\; \mathrm{Enc}'(\boxed{sk(x)}\,\boxed{r})$$

<u>With probability $1 - 1/\mathrm{poly}(n)$,</u>

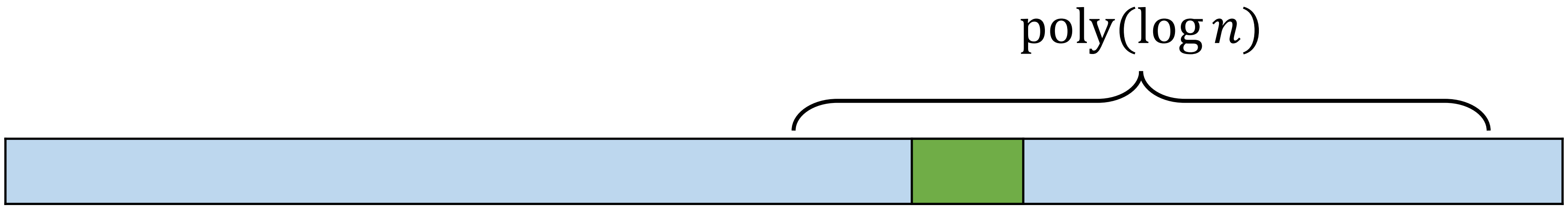
- Any two substrings of length $O(\log\, n)$ are distinct.
- Block length $\le \mathrm{poly}(\log n)$


$c' =$
Dec′
Sketch Recover
$sk(x')$
r
x
$\oplus$
PRG(r)
$=$
$\mathrm{Dec}(c') =$
x'

Derandomize the Uniform Random String

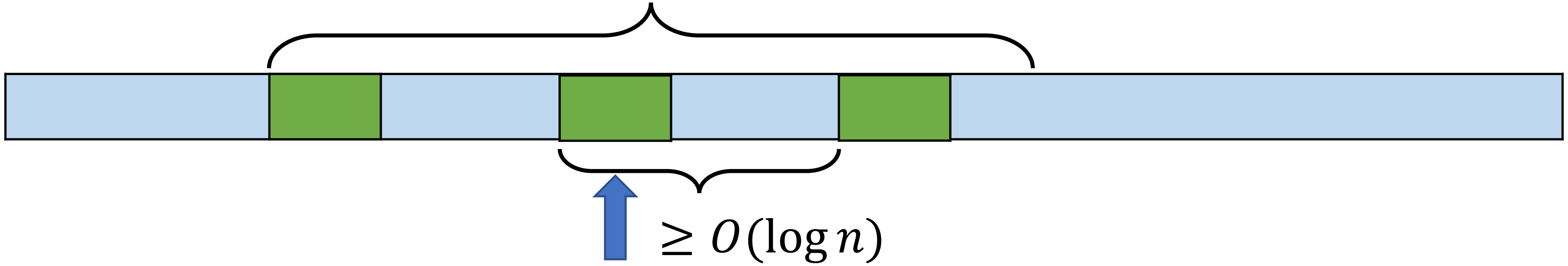
With probability $1 - 1/\mathrm{poly}(n)$:

1. Any two substrings of length $O(\log\ n)$ are distinct. PRG_1
2. Any interval of length $\mathbf{poly}(\mathbf{log}\, n)$ contains p.



PRG_2

3. Any interval of length $\mathbf{poly}(\mathbf{log}\, n)$ starting from p contains a chosen p.



PRG_3

$$\mathrm{PRG}(r) = \mathrm{PRG}_1(r) \oplus \mathrm{PRG}_2(r) \oplus \mathrm{PRG}_3(r), \qquad |r| = O(\log n)$$

Open Questions

- Optimal Error Correcting Code and Deterministic document exchange for all k.

Thank You!