Structured Encryption and Leakage Suppression

Tarik Moataz

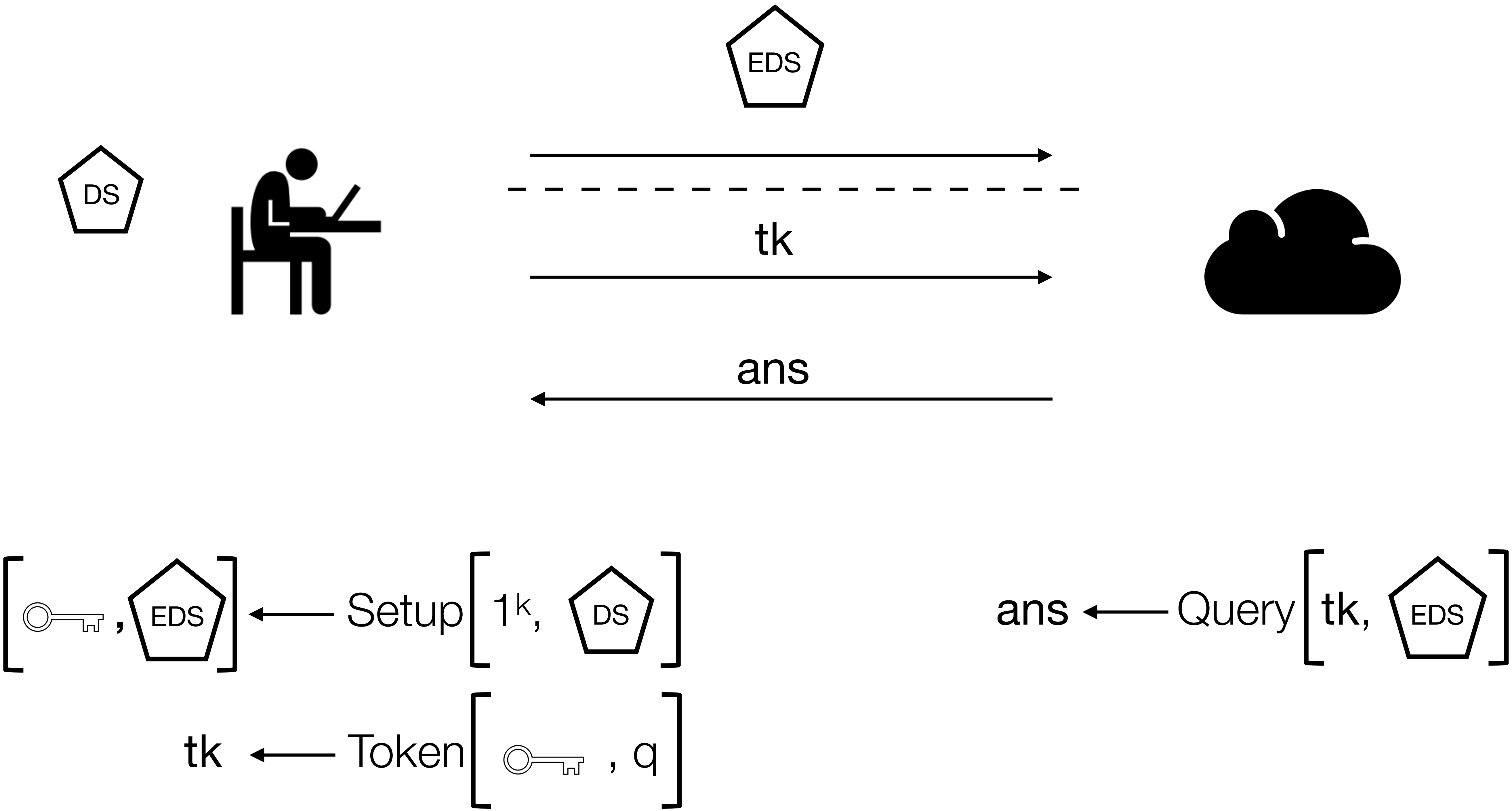
Part I is a joint work with Seny Kamara and Olya Ohrimenko

Part II is a joint work with Seny Kamara

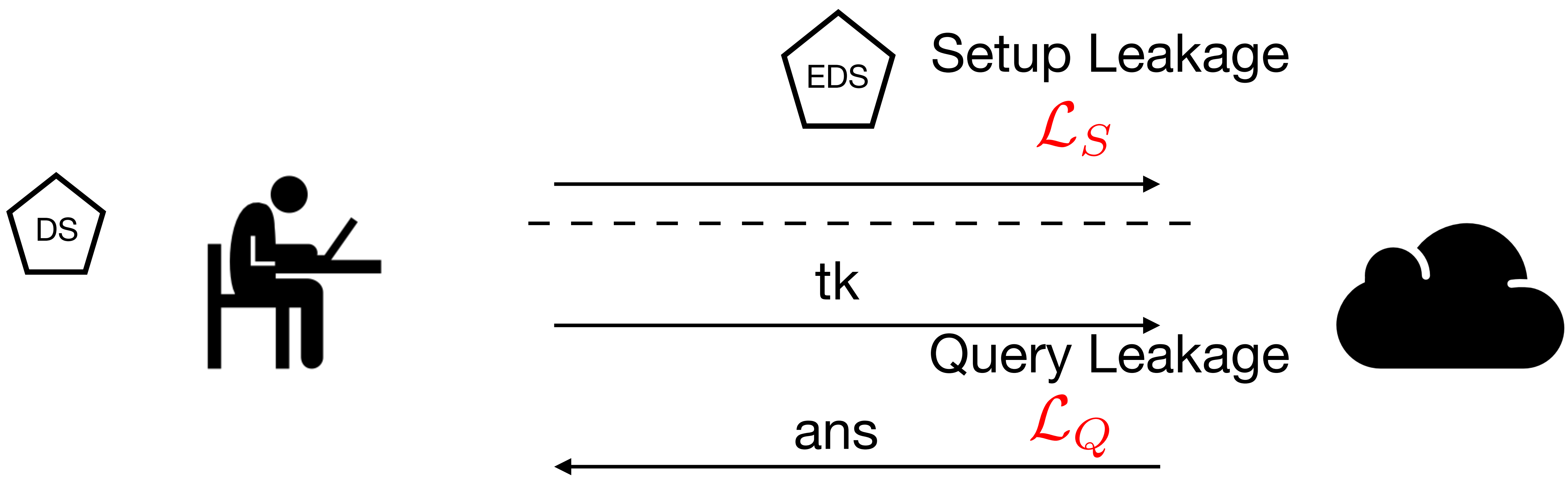
BROWN

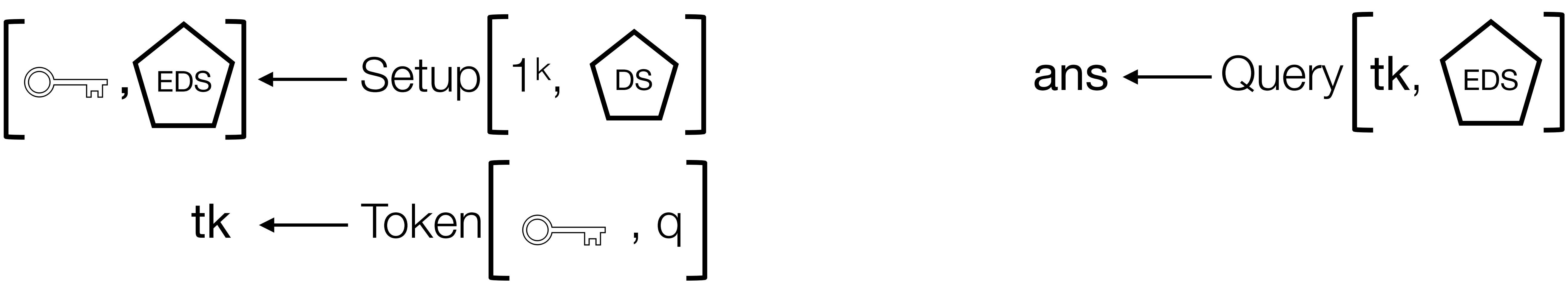
ENCRYPTED SYSTEMS LAB

Structured Encryption (STE) [CK10]



Structured Encryption [CK10]



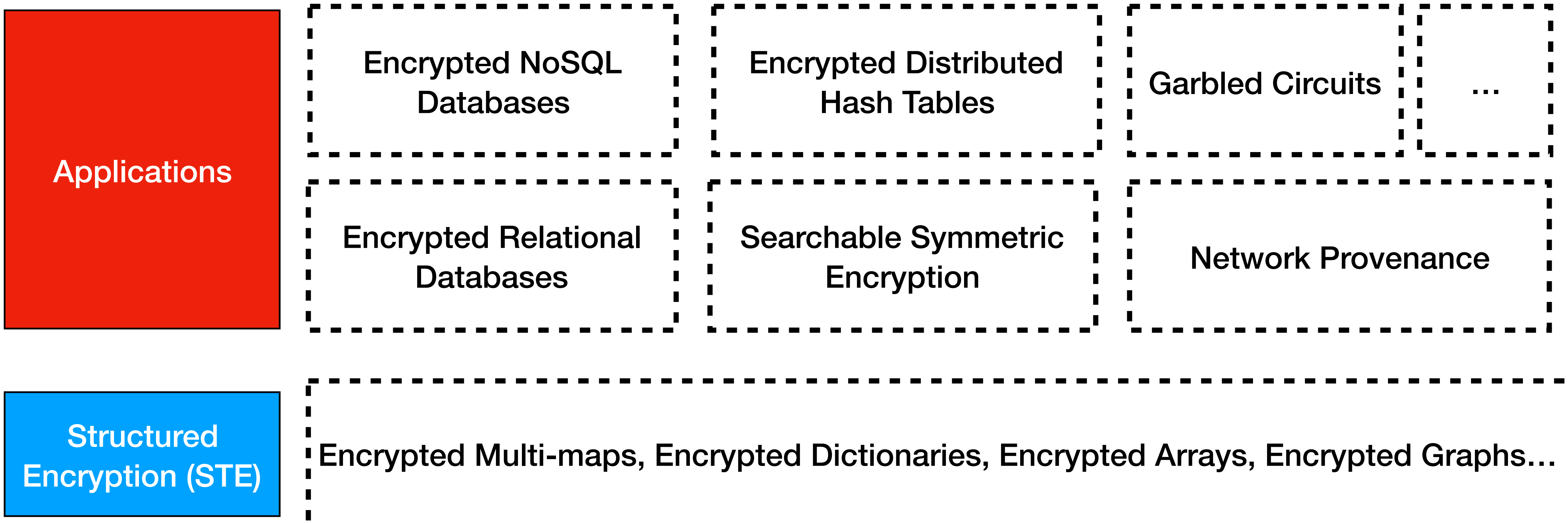


Structured Encryption [CK10]

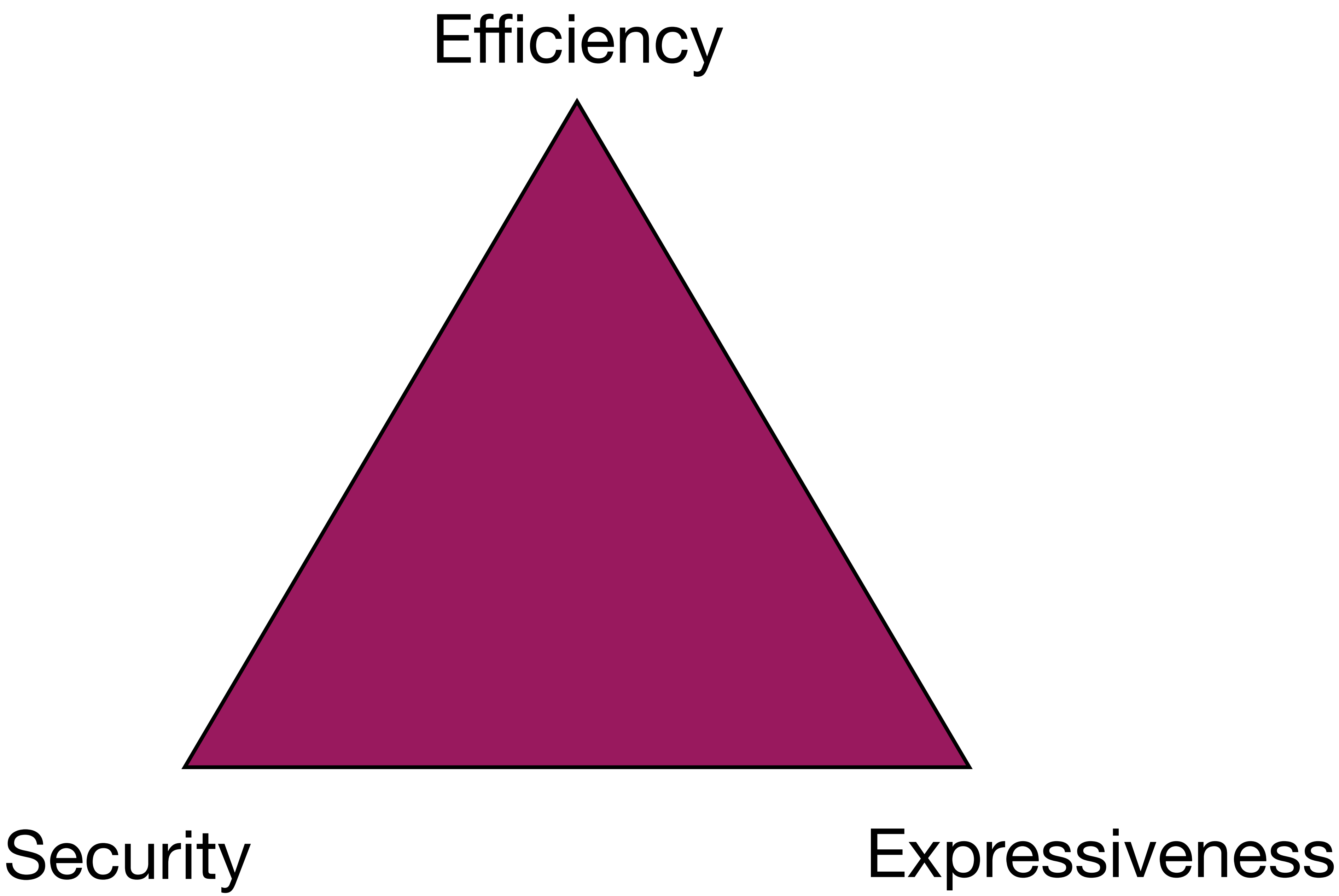
An STE scheme is $(\mathcal{L}_S, \mathcal{L}_Q)$-secure if

- It reveals no information about the structure beyond $\mathcal{L}_S$
- It reveals no information about the structure and queries beyond $\mathcal{L}_Q$

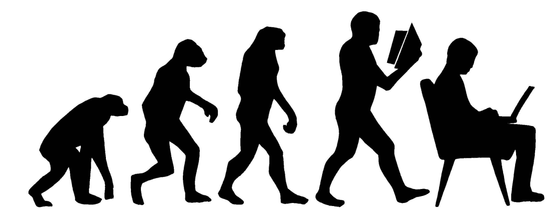
Structured Encryption [CK10]



Structured Encryption [CK10]



Structured Encryption Evolution



Efficiency

- ‘00 Linear per file [SWP00]
- ‘03 Linear [Goh03]
- ‘06 Optimal [CGKO06,CK10]
- ‘12 Dynamism [KPR12], [KP13], [CJJJKRS14]
- ‘14 I/O efficiency [CT14], [CJJJKRS14], [ANSS16], [DPP18], [ASS18]

Expressiveness

- ‘00 Single-keyword SSE [SWP00], [Goh03], [CGKO06], [CJJJKRS14]
- ‘06 Multi-user SSE [CGKO06], [JJKRS13], [PPY16], [HSWW18]
- ‘13 Boolean SSE [CJJKRS13], [PKVK+14], [KM17]
- ‘14 Range SSE [PKVK+14], [FJKNRS15]
- ‘18 STE-based SQL [KM18]

Security

- ‘06 Leakage-parametrized security definitions [CGKO06]
- ‘12 Adv. models [KO12],[BFP16], [AKM18]
- ‘12 Attacks [IKK12], [CGPR15], [ZKP16], [KMNO16], [LMP18], [GLMP18]
- ‘14 Forward/Backward Security [SPS14], [Bost16], [LC17], [BMO17], [AKM18]

What about Leakage?

What about Leakage?

Cryptanalysis

[IKK12]

Measure

?

Suppression

[KMO18]

Cryptanalysis

Def: Given a leakage profile, design attacks to recover the queries or the data under some assumptions

Goal: empirically learn the impact of a leakage pattern in real-world

Limitations: the gap between assumptions and reality can get wide

Measure

Def: Given a leakage profile, quantify (e.g., in bits) a specific leakage pattern

Goal: theoretically compare between leakage patterns

Limitations: (maybe) no possible total order (work in progress!)

Suppression

Def: Given a leakage profile, design a compiler or a transform to suppress a specific leakage pattern

Goal: develop tools to suppress various leakage patterns

Limitations: introducing some overhead

Part 1*

Suppressing Leakage

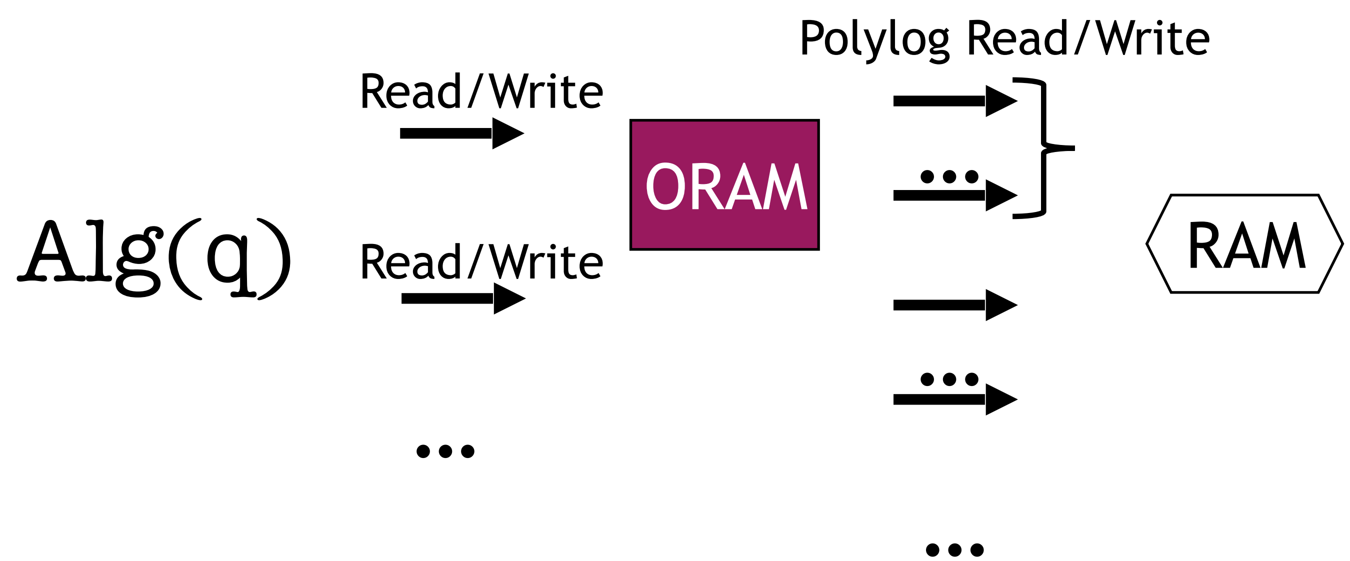
**joint work with Seny Kamara and Olya Ohrimenko*

https://eprint.iacr.org/2018/551

Q: is there an existing approach to reduce leakage?

Existing Approaches

- ORAM Simulation [GO96], [SvDSFRD13]
  - Generic
  - Small Leakage profile
  - Interactive
  - Efficiency



- Garbled RAM [LO13], [GHLORW14]
- Custom Schemes [WNLCSSH14], [BM16]

Q: are there more efficient ways to suppress leakage?

Background
Modeling Leakage

- qeq : query equality
  - search pattern
- did : data identity
- req : response equality
- rid : response identity
  - access pattern

- qlen : query length
- rlen : response length
  - volume pattern
- mqlen : maximum query length
- mrlen : maximum response length
- srlen : sequence response length
- dsize : data size

Background
Non-Repeating Sub-Pattern

- Non-repeating sub-pattern

$$\mathsf{patt}(\mathsf{DS}, q_1, \cdots, q_t) = \begin{cases} \mathsf{nrp}(\mathsf{DS}, q_1, \cdots, q_t) & \text{if } q_i \neq q_j, \forall i, j \in [t] \\ \mathsf{rp}(\mathsf{DS}, q_1, \cdots, q_t) & \text{otherwise.} \end{cases}$$

- Example

$$\mathsf{qeq}(\mathsf{DS}, q_1, \cdots, q_t) = \begin{cases} \perp & \text{if } q_i \neq q_j, \forall i, j \in [t] \\ \mathsf{rp}(\mathsf{DS}, q_1, \cdots, q_t) & \text{otherwise.} \end{cases}$$

Leakage Suppression

Through Compilation

STE → Compilation → STE’

$$\Lambda = \left(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}} = (\mathsf{patt}_1, \cancel{\mathsf{patt}_2})\right)$$

$$\Lambda' = \left(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}} = \mathsf{patt}_1\right)$$

Suppressing Query Equality

STE → Cache-Based Compiler (CBC) → **STE'**

$$\Lambda = \Big(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}} = (\cancel{\quad}\mathsf{q}, \mathsf{patt})\Big)$$

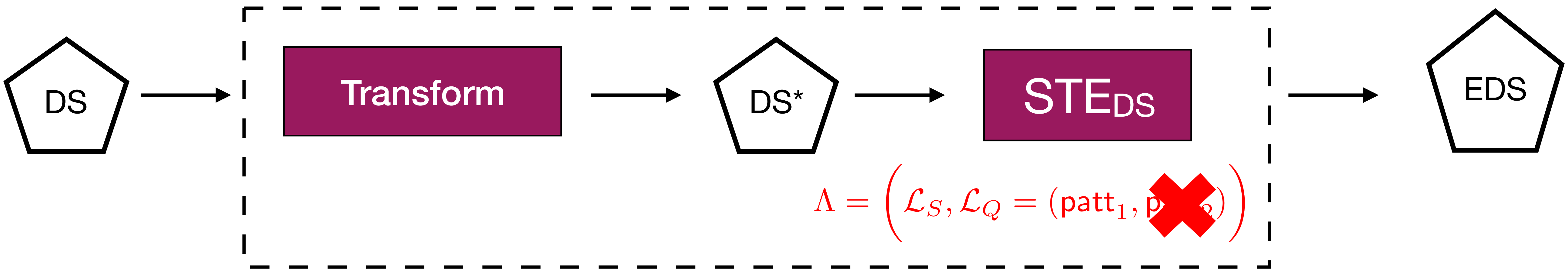
$$\Lambda' = \Big(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}} = \mathsf{nrp}\Big)$$

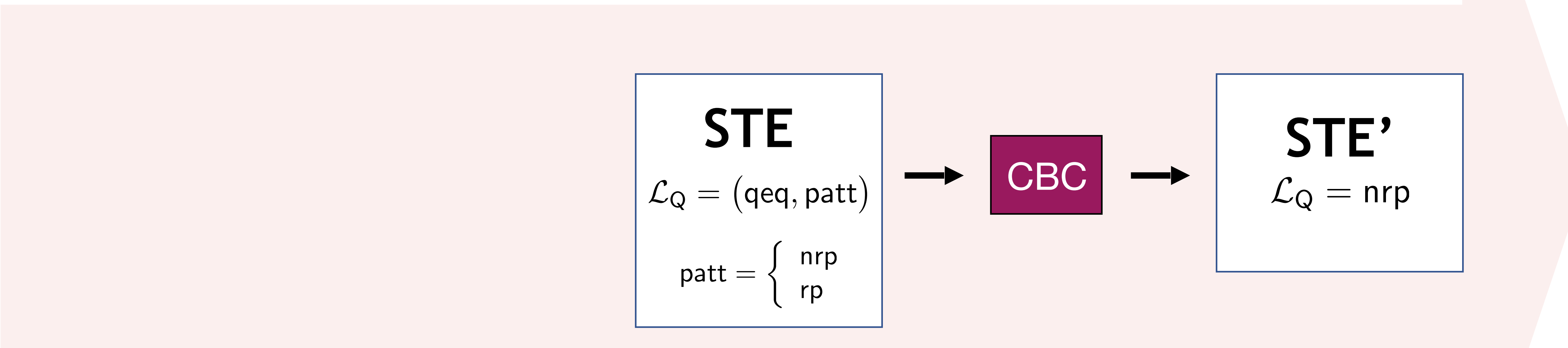
$$\mathsf{patt} = \begin{cases} \mathsf{nrp} \\ \cancel{\quad} \end{cases}$$

Leakage Suppression
Through Transformation

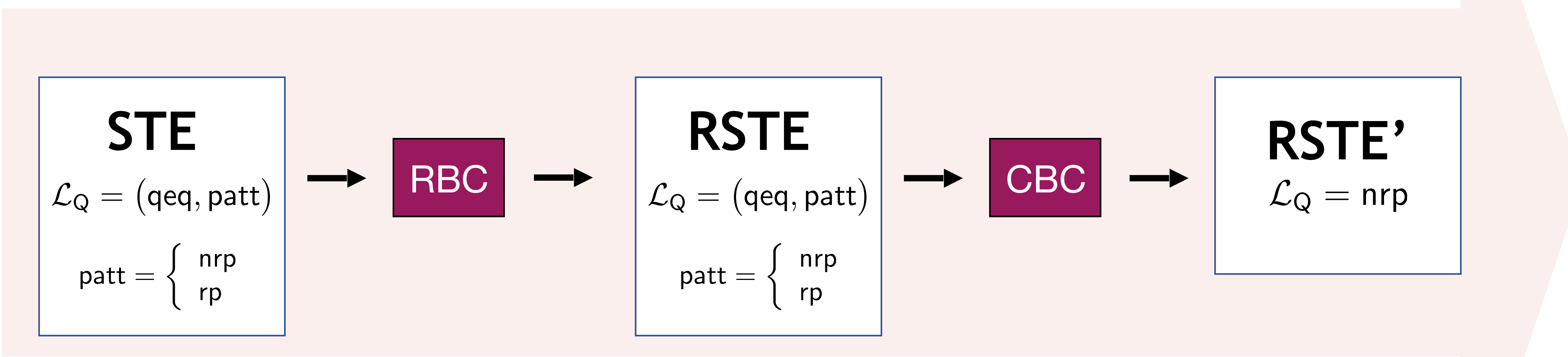
STE'$_{DS}$

$$\Lambda' = \Big(\mathcal{L}_S, \mathcal{L}_Q = \mathsf{patt}_1\Big)$$





- Cache-based Compiler (CBC)
    - suppresses the query equality and the repeating sub-pattern
    - induces an additive poly-log overhead
    - Requires a rebuildable STE



- Rebuild Compiler (RBC)
  - makes any STE scheme rebuildable
  - preserves the scheme’s query efficiency
  - adds a super-linear rebuild cost

STE

$\mathcal{L}_{\mathsf{Q}} = (\mathsf{qeq}, \mathsf{patt})$

$\mathsf{patt} = \begin{cases} \mathsf{nrp} \\ \mathsf{rp} \end{cases}$

$\longrightarrow$ **RBC** $\longrightarrow$

RSTE

$\mathcal{L}_{\mathsf{Q}} = (\mathsf{qeq}, \mathsf{patt})$

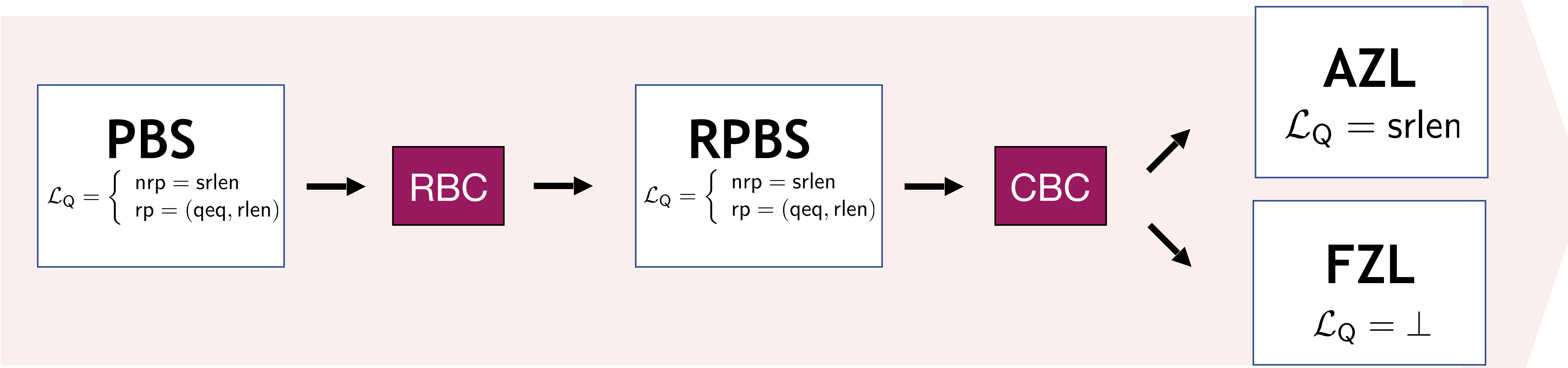
$\mathsf{patt} = \begin{cases} \mathsf{nrp} \\ \mathsf{rp} \end{cases}$

$\longrightarrow$ **CBC** $\longrightarrow$

RSTE’

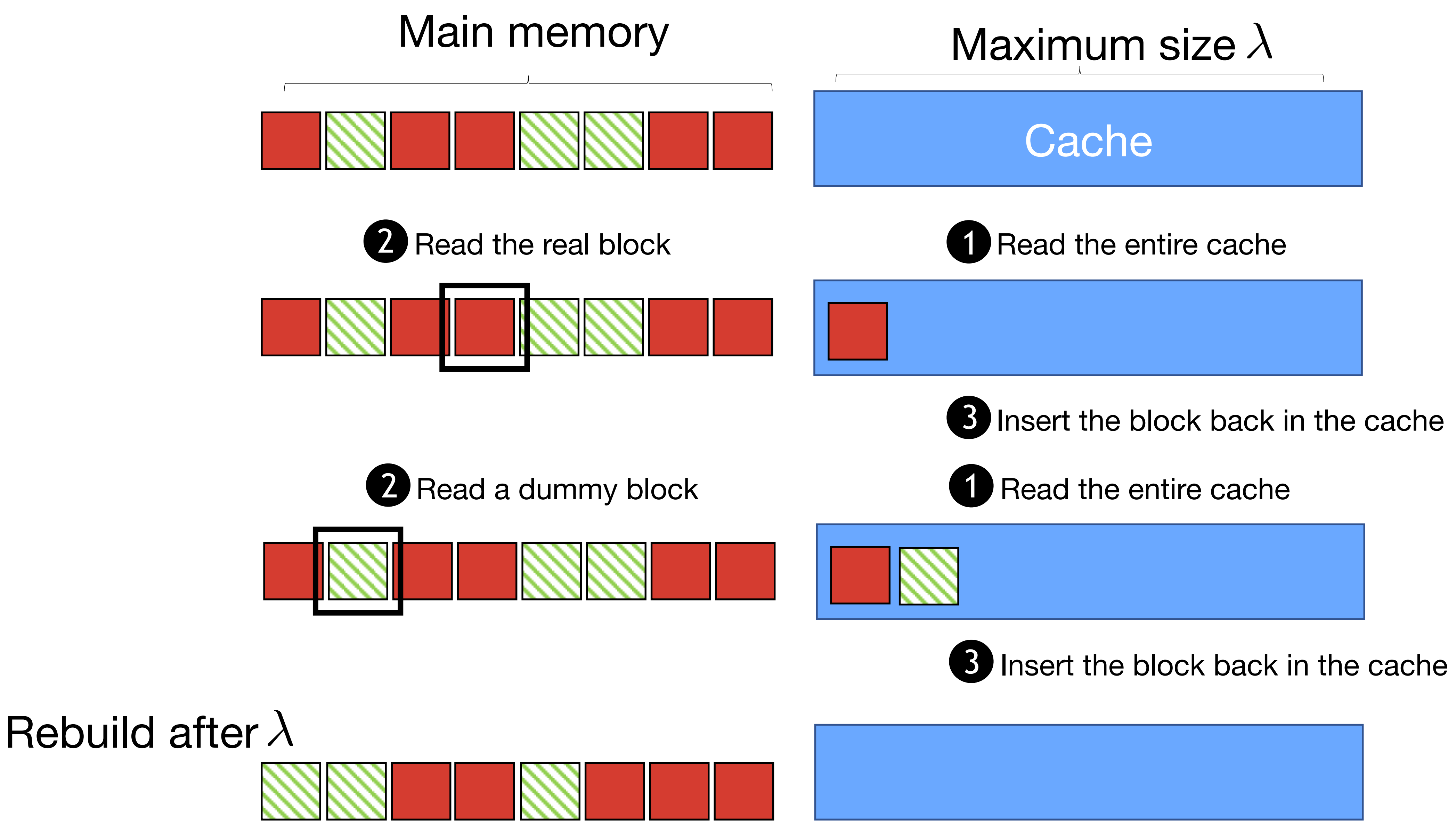
$\mathcal{L}_{\mathsf{Q}} = \mathsf{nrp}$

The problem boils down to reduce nrp of the base STE scheme

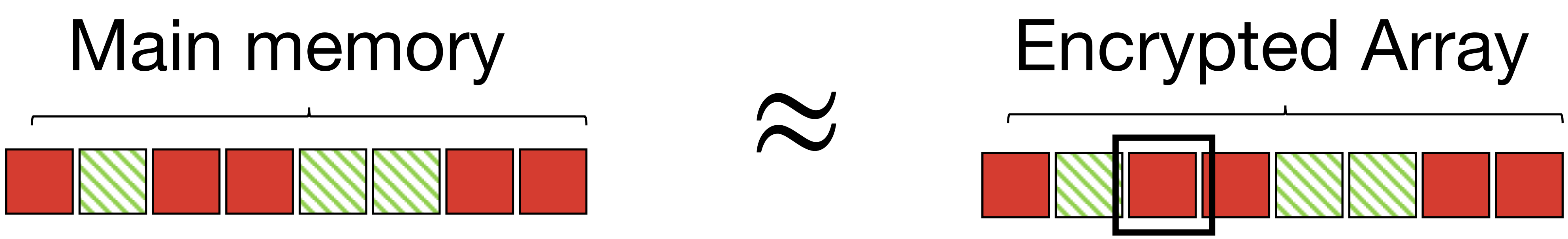


- Piggyback scheme (PBS)
  - hides the response length for non-repeating queries
  - introduces query latency

Square-Root ORAM [GO96]

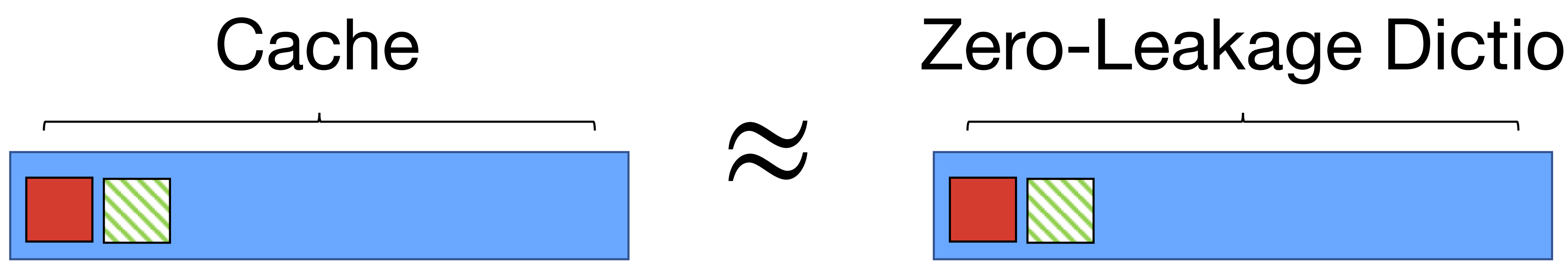


Reinterpreting the Square-Root Solution



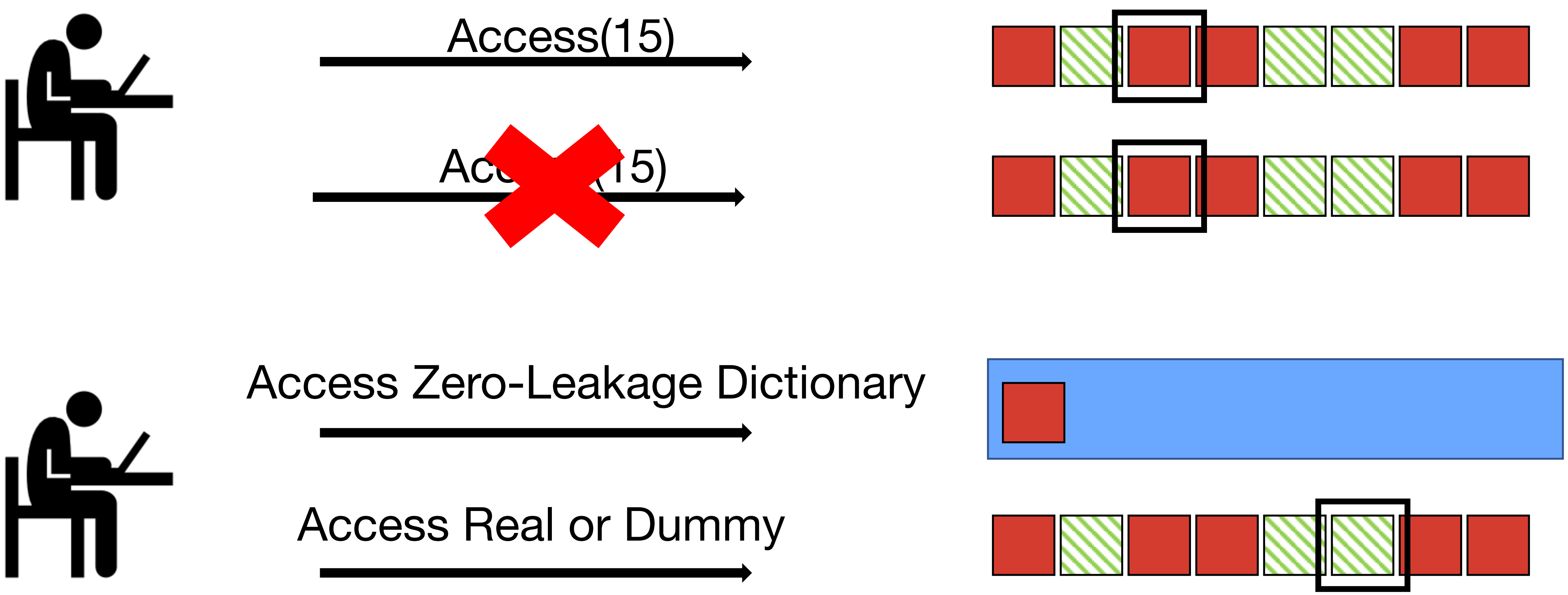
- Main memory is an encrypted array construction
- Accessing element is done deterministically through PRP evaluation
- Adversary learns if/when an access to the same element is repeated
  - Leaks query equality

Reinterpreting the Square-Root Solution

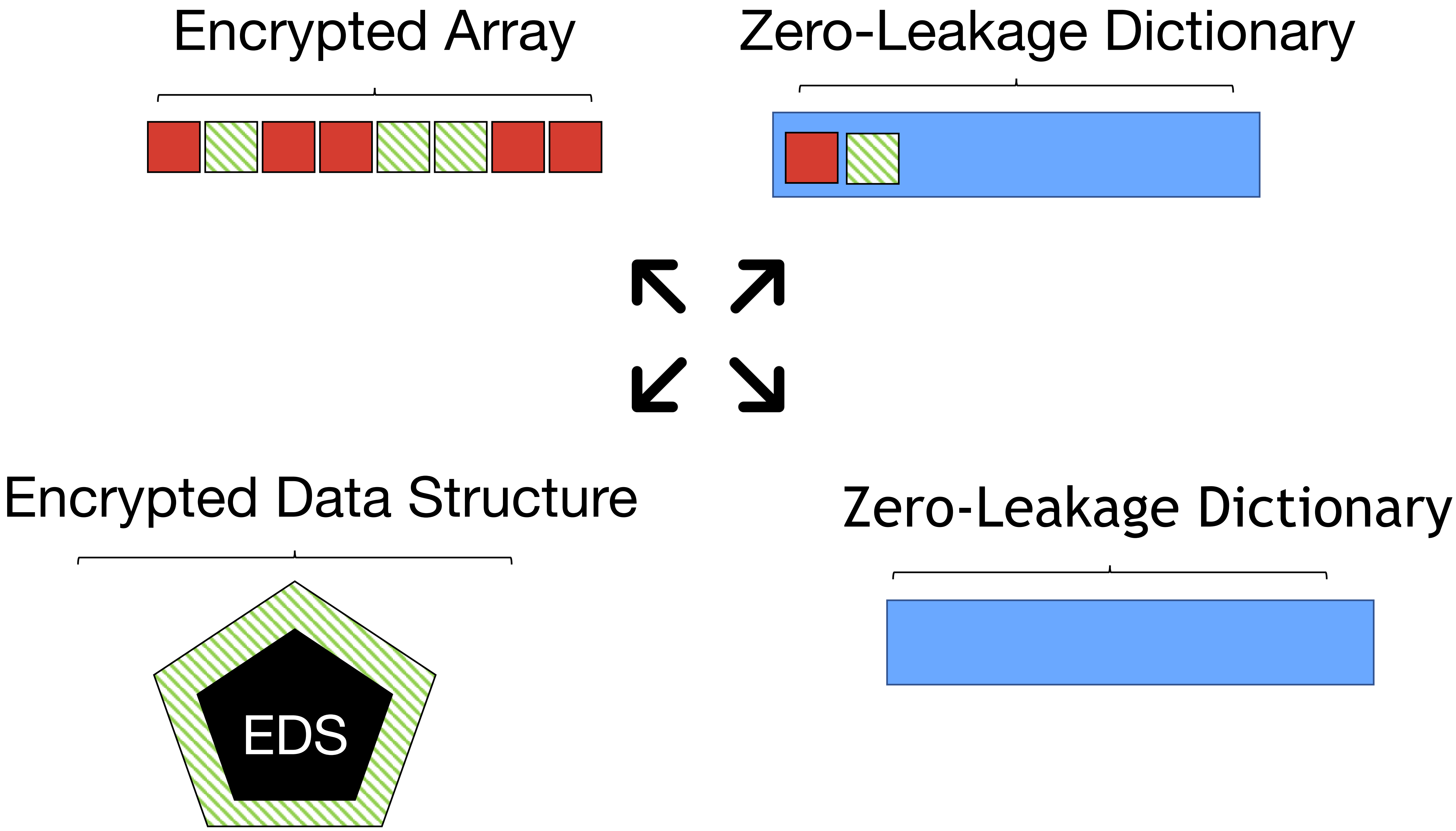


- The cache is an encrypted dictionary data structure
  - Given a label, it outputs an element or ⊥
- The cache is accessed in its entirety
  - Most trivial zero-leakage dictionary construction; therefore no query leakage

Reinterpreting the Square-Root Solution

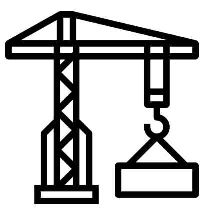


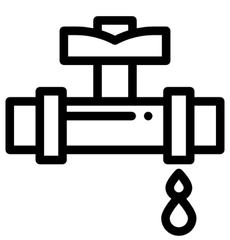
Reinterpreting the Square-Root Solution



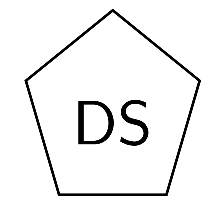
Reinterpreting the Square-Root Solution

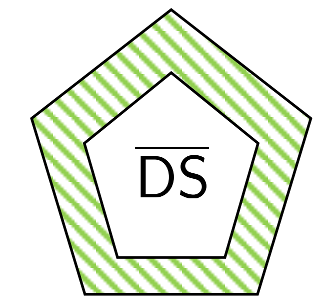
- Requirements
  - EDS scheme has to be rebuildable
  - Data structure has to be extendable and safe
  - Base scheme has to have smaller non-repeating sub-pattern





Data Structure Extension



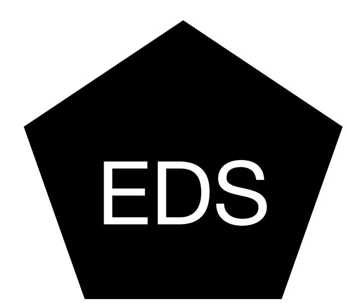


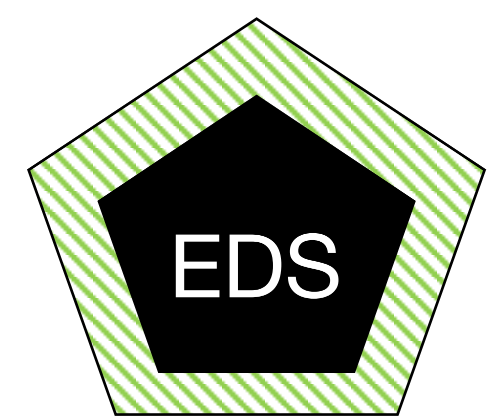
- λ-extension:
  - Extend the query space of the data structure with λ dummies
  - $\forall q \in \overline{\mathbb{Q}} \setminus \mathbb{Q}$ s.t. $\mathbb{Q} \subseteq \overline{\mathbb{Q}}$

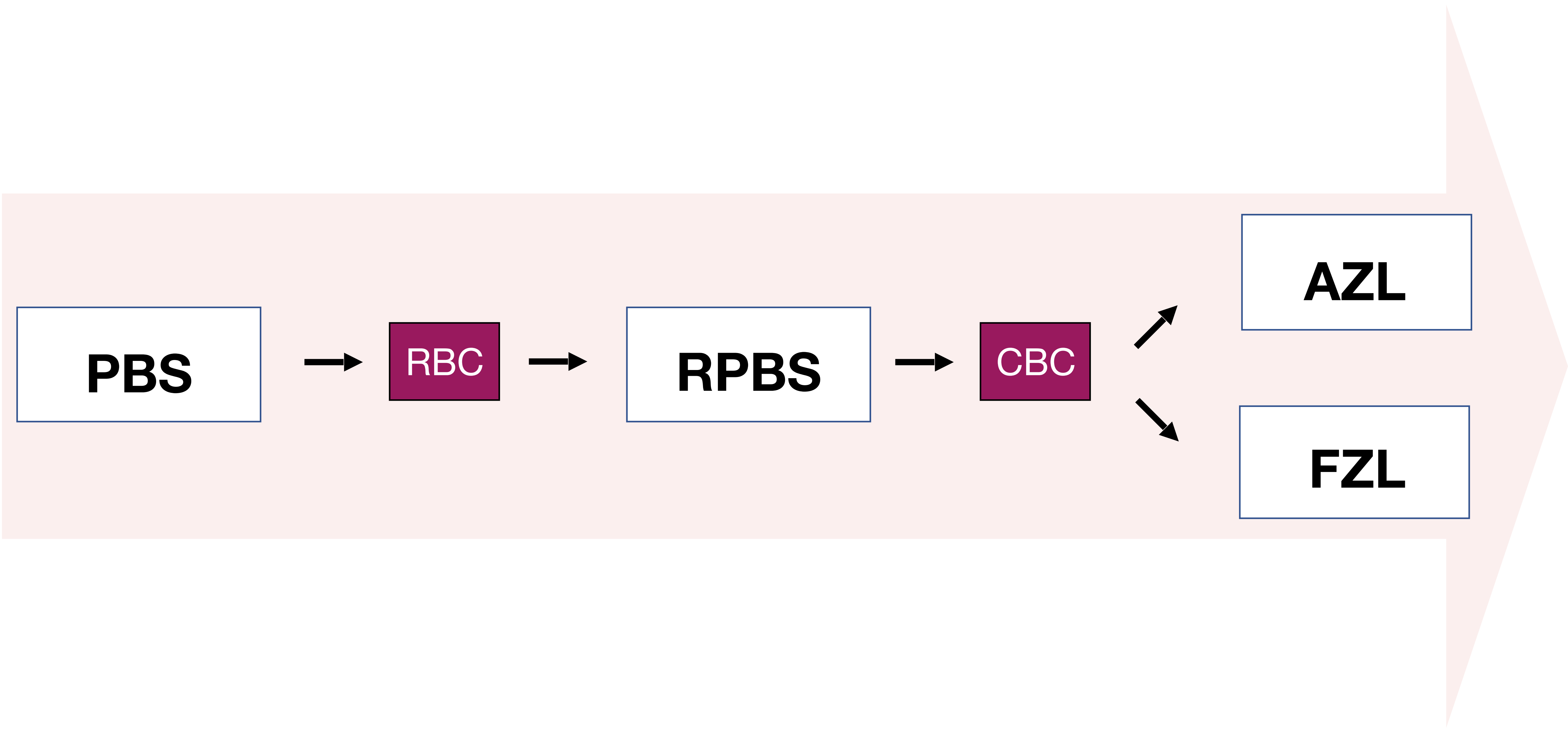
$$\mathsf{Query}\left(\overline{\mathsf{DS}}, q\right) = \perp$$

- Safe λ-extension:

$$\begin{aligned} \mathcal{L}_{\mathsf{S}}\left(\overline{\mathsf{DS}}\right) &\leq \mathcal{L}_{\mathsf{S}}\left(\mathsf{DS}\right) \\ \mathcal{L}_{\mathsf{Q}}\left(\overline{\mathsf{DS}}, q\right) &\leq \mathcal{L}_{\mathsf{Q}}\left(\mathsf{DS}, q\right) \end{aligned}$$

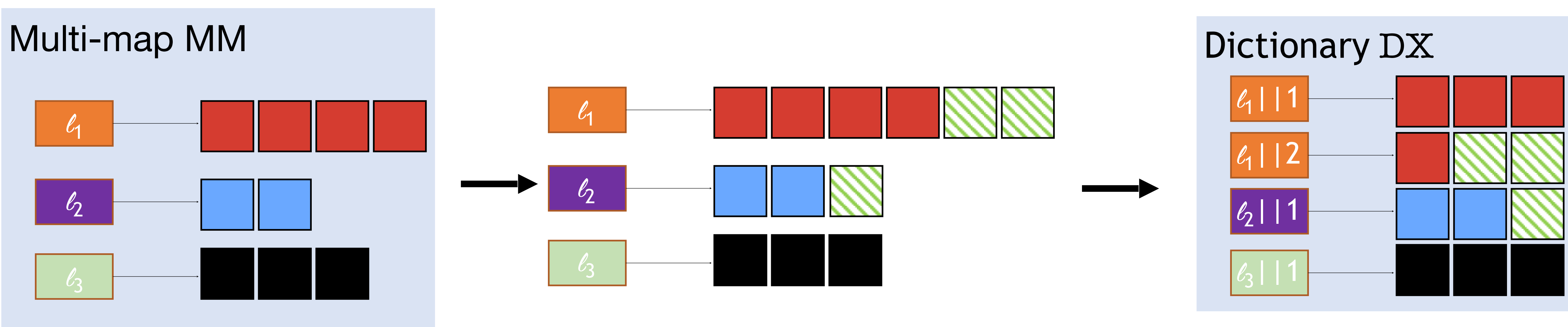




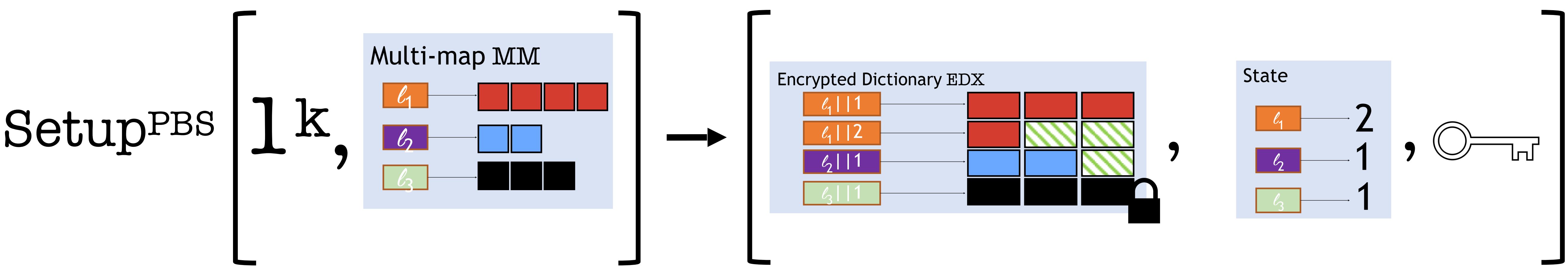

PBS
RBC
RPBS
CBC
AZL
FZL

PBS: Data transformation

- Batch size (ex: $\alpha = 3$)
- Pad all responses to a multiple of α

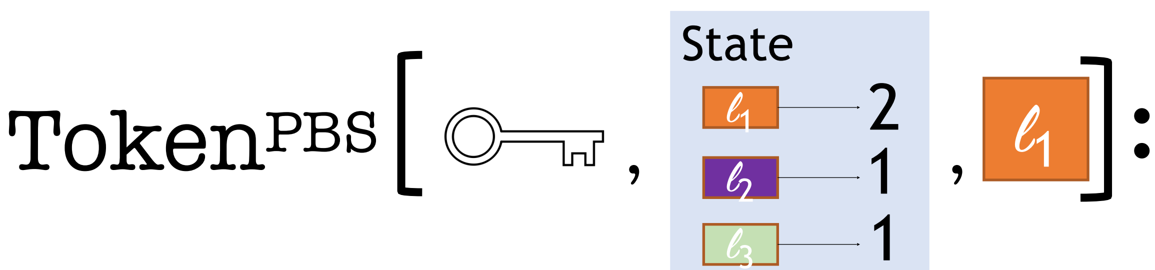


PBS Details


SetupPBS
1k,
Multi-map MM
ℓ1
ℓ2
ℓ3
Encrypted Dictionary EDX
ℓ1||1
ℓ1||2
ℓ2||1
ℓ3||1
State
ℓ1 2
ℓ2 1
ℓ3 1

PBS Details

- Consider a sequence of labels $\mathbf{q} = (\ell_1, \ell_2)$

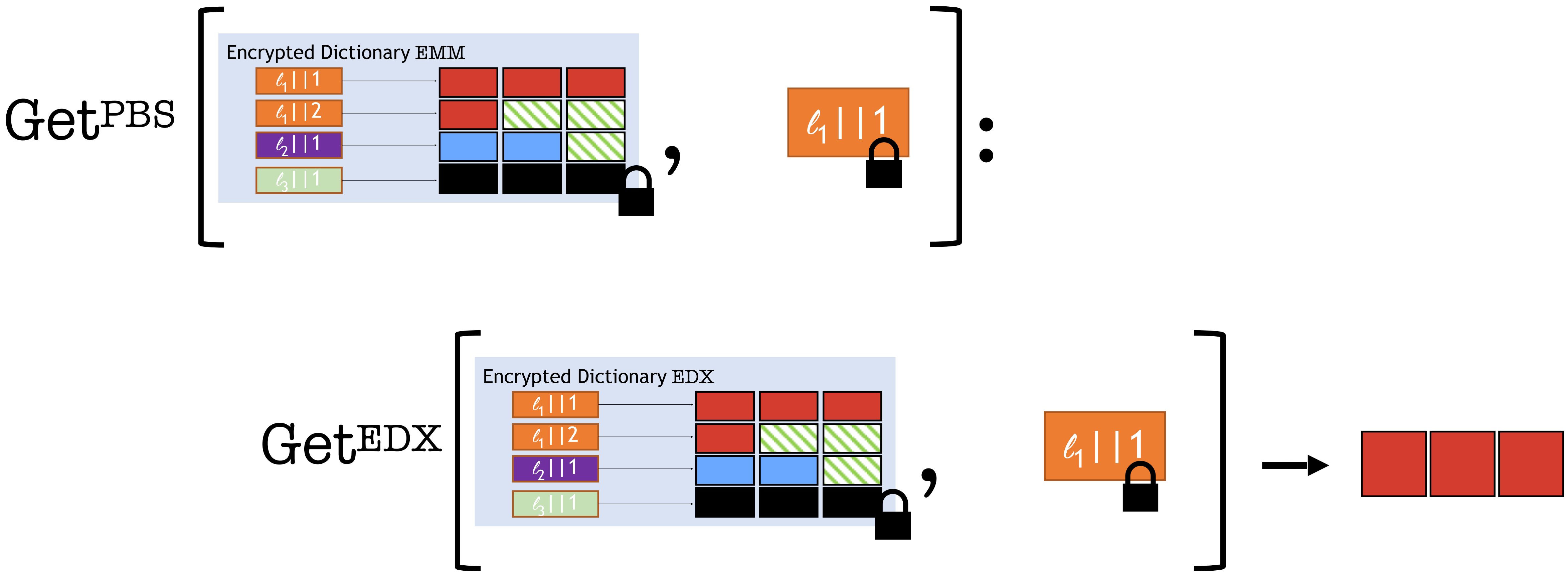


1. ℓ_1 has 2 batches
2. Instantiate a queue

3. Compute

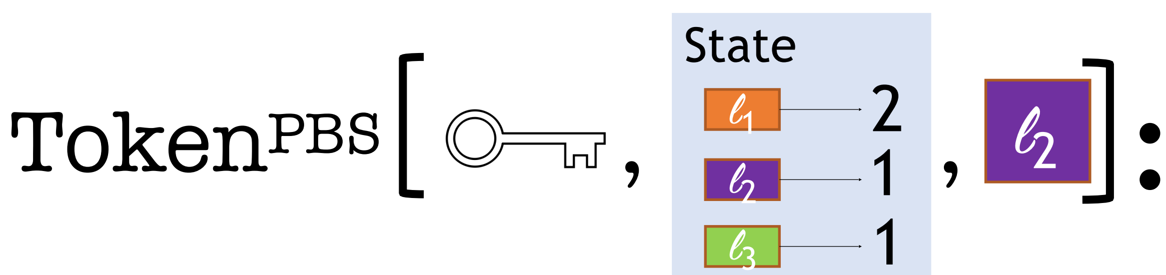
4. Update queue

PBS Details



Get^PBS
Encrypted Dictionary EMM
ℓ1||1
ℓ1||2
ℓ2||1
ℓ3||1
ℓ1||1
Get^EDX
Encrypted Dictionary EDX
ℓ1||1
ℓ1||2
ℓ2||1
ℓ3||1
ℓ1||1

PBS Details

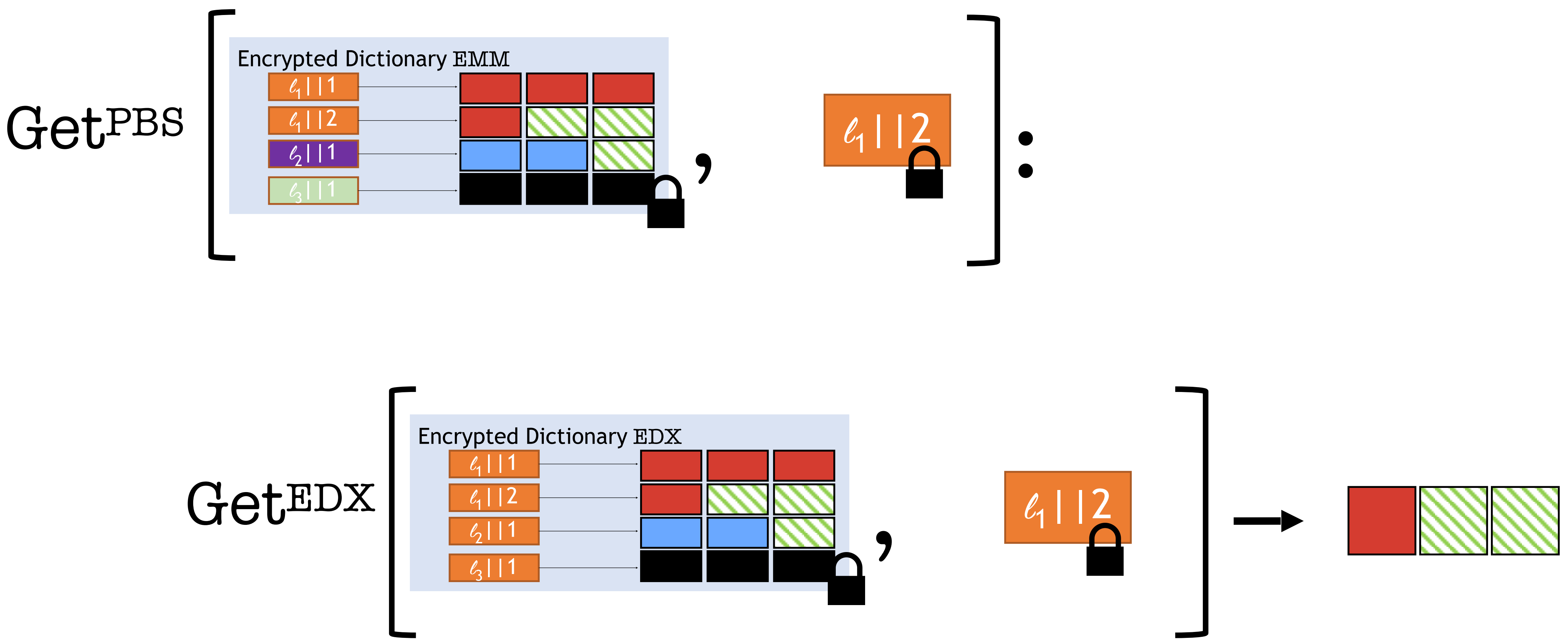


1. ℓ_2 has 1 batch
2. Update the queue

3. Compute

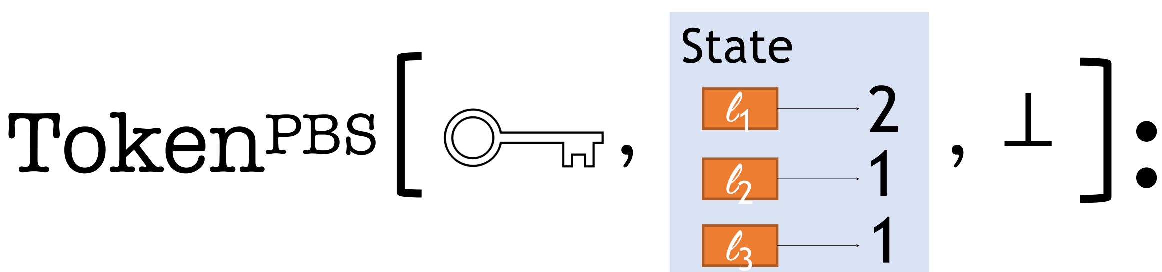
4. Update queue

PBS Details



Get^PBS
Encrypted Dictionary EMM
ℓ1||1
ℓ1||2
ℓ2||1
ℓ3||1
ℓ1||2
Get^EDX
Encrypted Dictionary EDX
ℓ1||1
ℓ1||2
ℓ2||1
ℓ3||1
ℓ1||2

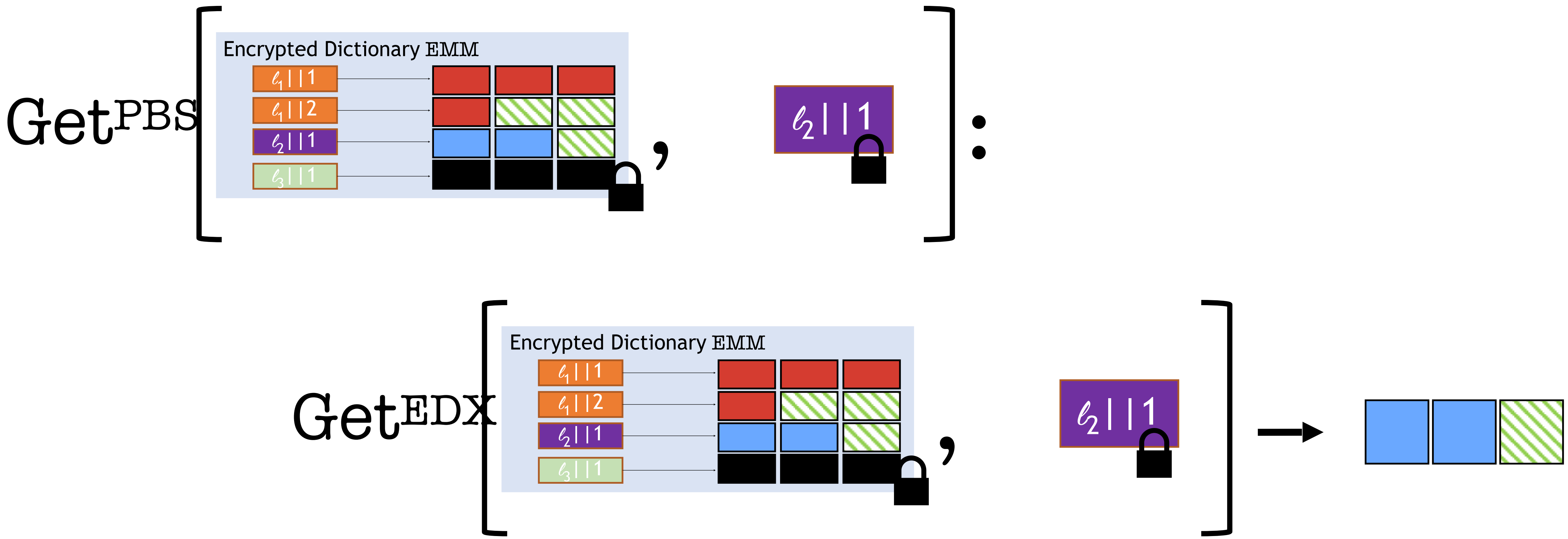
PBS Details



1. Compute

2. Update queue

PBS Details


Get^PBS
Encrypted Dictionary EMM
ℓ1||1
ℓ1||2
ℓ2||1
ℓ3||1
ℓ2||1
Get^EDX
Encrypted Dictionary EMM
ℓ1||1
ℓ1||2
ℓ2||1
ℓ3||1
ℓ2||1

PBS Latency

- The worst-case query sequence of size t has latency

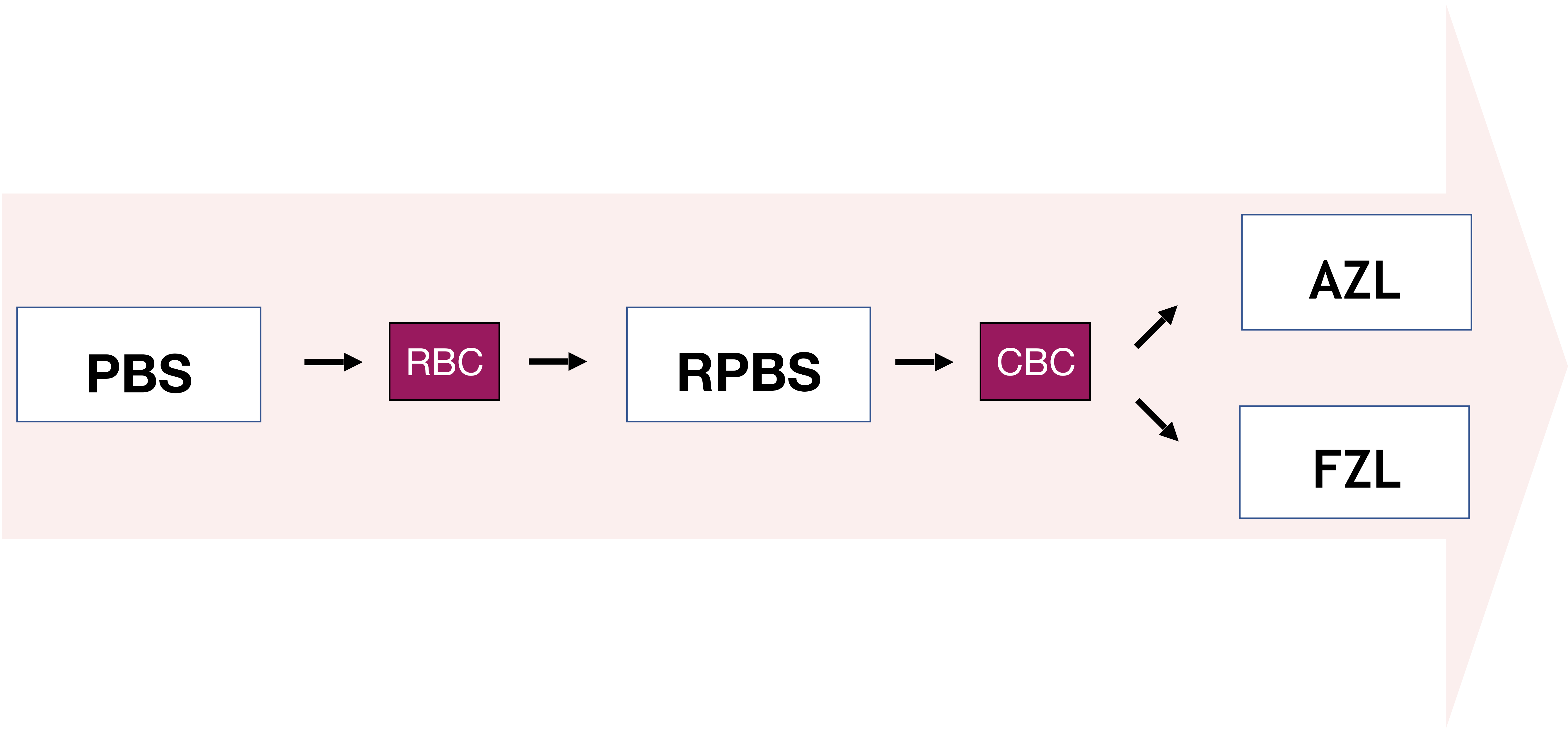
$$t \cdot \left(\frac{\max_{r \in \mathbb{R}_{\mathsf{DS}}} |r|_w}{\alpha} - 1 \right)$$

- Real-world sequences have latency

$$\varepsilon \cdot t$$

with probability at least $1 - \exp\left(-2t \left(\varepsilon \cdot \frac{\alpha}{\max_{r \in \mathbb{R}_{\mathsf{DS}}} |r|_w} \right)^2 \right)$

where queries are drawn from a Zipf distribution and longer responses are mapped to less frequent labels


PBS
RBC
RPBS
CBC
AZL
FZL

AZL Analysis

- Worst-case query complexity over λ queries

$$\sum_{i=1}^{\lambda} \mathsf{T}^{\mathsf{eds}}(q_i) + O\left(\lambda \cdot \max_{q \in \mathbf{q}} |r|_w \cdot \log^2 \lambda\right) + O\left(\sum_{r \in \mathbb{R}_{\mathsf{DS}}} |r|_w \cdot \log^2 \#\mathbb{Q}_{\mathsf{DS}}\right)$$

- Comparison to ORAM simulation (Path-ORAM [SvDSFRD13])

$$\mathsf{T}^{\mathsf{eds}}(q_1, \ldots, q_\lambda) = o\left(\mathsf{T}^{\mathsf{tree}}(q_1, \ldots, q_\lambda)\right)$$

when

$$\sum_{r \in \mathbb{R}_{\mathsf{DS}}} |r|_w = o\left(\sum_{i=1}^{\lambda} B(q_i) \cdot \max_{r \in \mathbb{R}_{\mathsf{DS}}} |r|_w\right) \text{ and } \lambda \cdot \max_{q \in \mathbf{q}} |\mathsf{qu}(\mathsf{DS}, q)|_w = o\left(\sum_{r \in \mathbb{R}_{\mathsf{DS}}} |r|_w\right)$$

Natural Assumption:
If response lengths are power–law distributed

Part 2*

Suppressing Volume

**joint work with Seny Kamara*

https://eprint.iacr.org/2018/978

Leakage Suppression

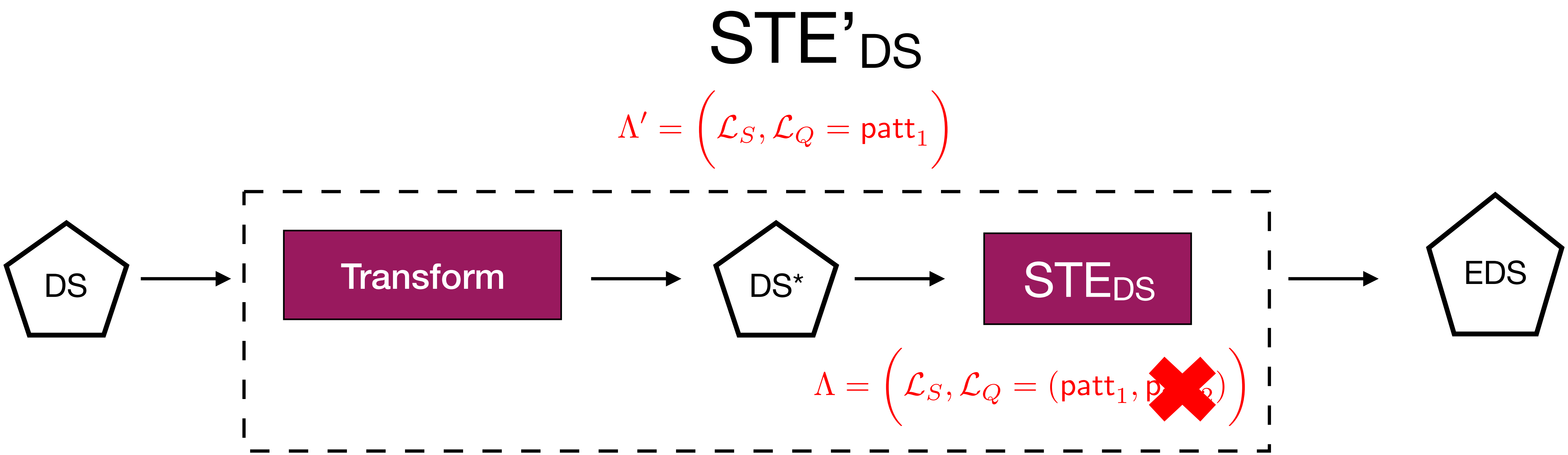
Through Compilation

STE → Compilation → STE'

$$\Lambda = \left(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}} = (\mathsf{patt}_1, \cancel{\mathsf{patt}_2})\right)$$

$$\Lambda' = \left(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}} = \mathsf{patt}_1\right)$$

Leakage Suppression
Through Transformation



Q: is there any other approach to suppress leakage?

Suppression

Black-box
Compilation

Data structure
Transformation

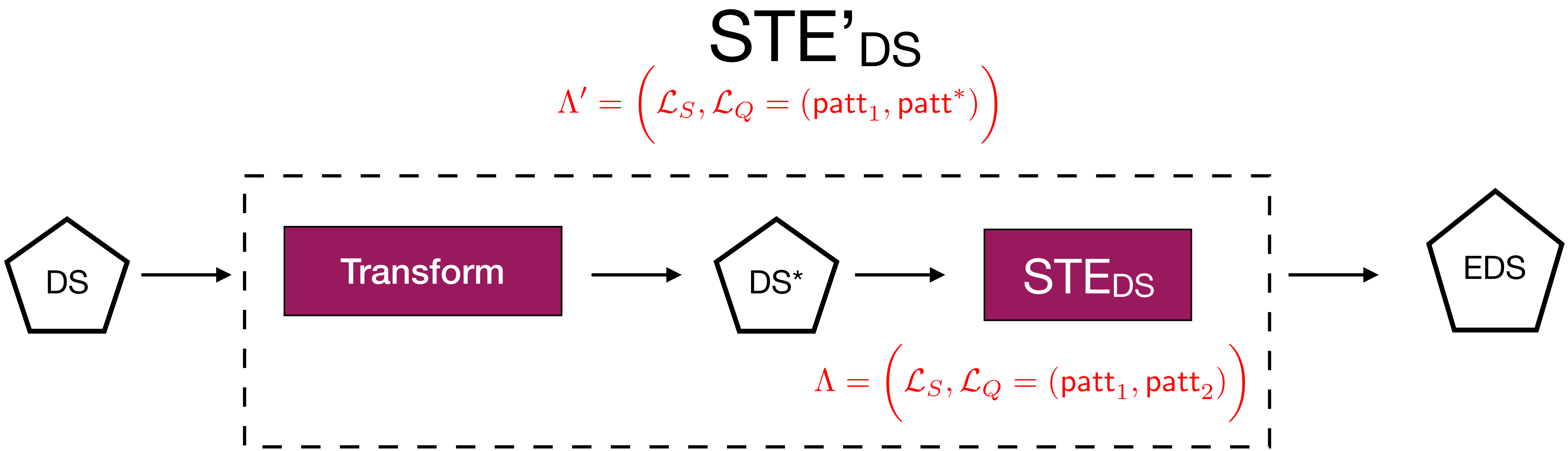
against
unbounded
adversary

against
bounded
adversary

Computationally-Secure Leakage

Unbounded Adversary vs. Bounded Adversary

Leakage Suppression [KMO18]
Through Transformation



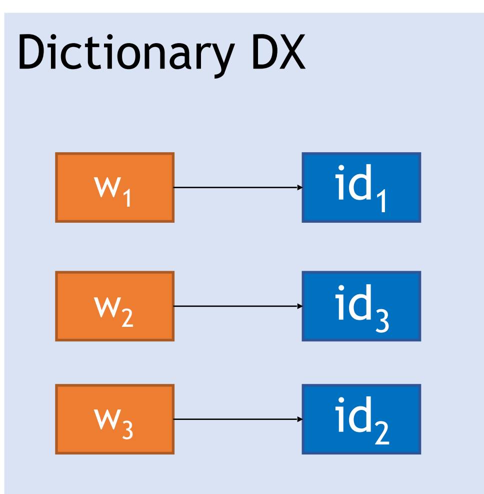
$$\mathsf{patt}^* \simeq \perp$$

Q: can we suppress the response length pattern?

Background

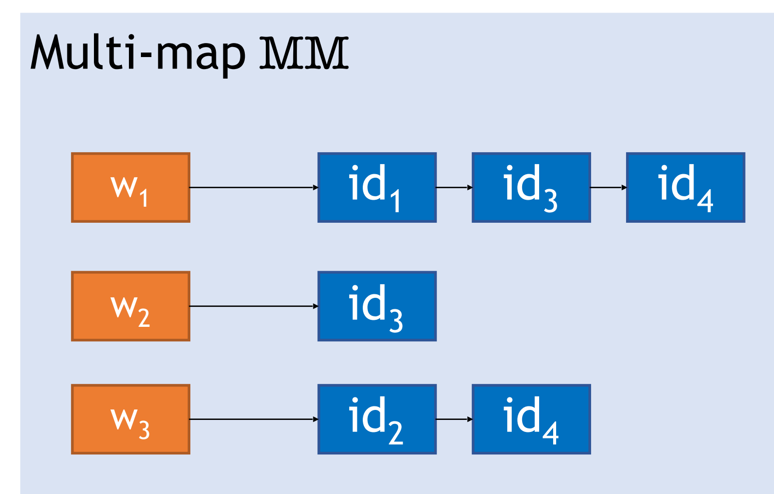
Dictionary and Multi-Map data structures

- DXs map labels to values



- Get: DX[w_3] returns id_2

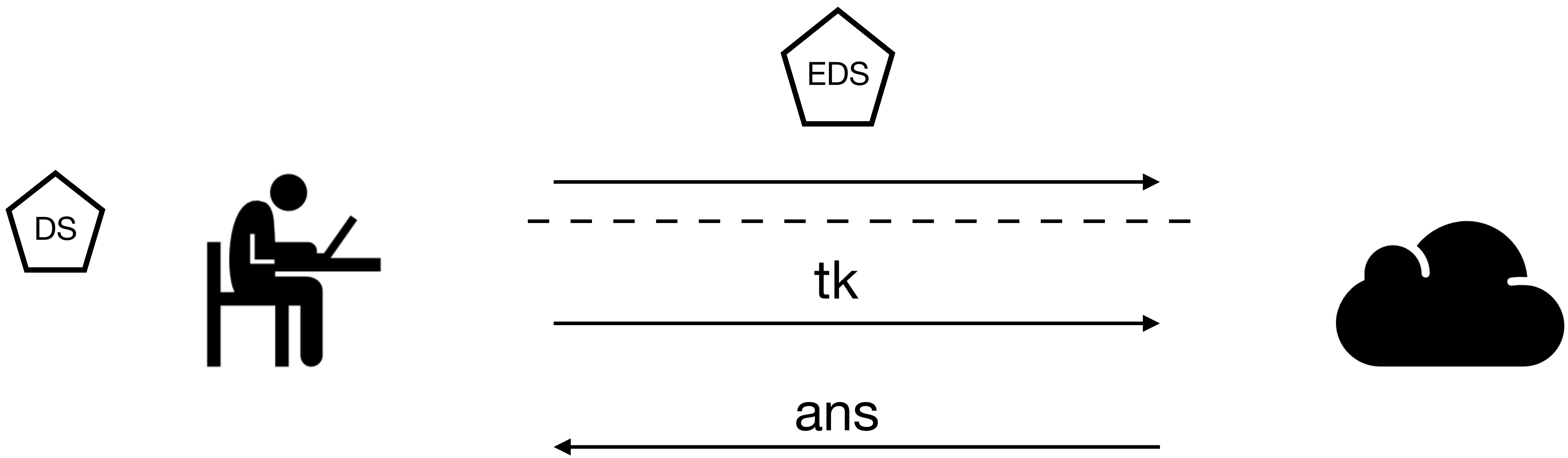
- MMs map labels to tuples



- Get: MM[w_3] returns (id_2 , id_4)

Background

Response Length Pattern (rlen) or Volume Pattern

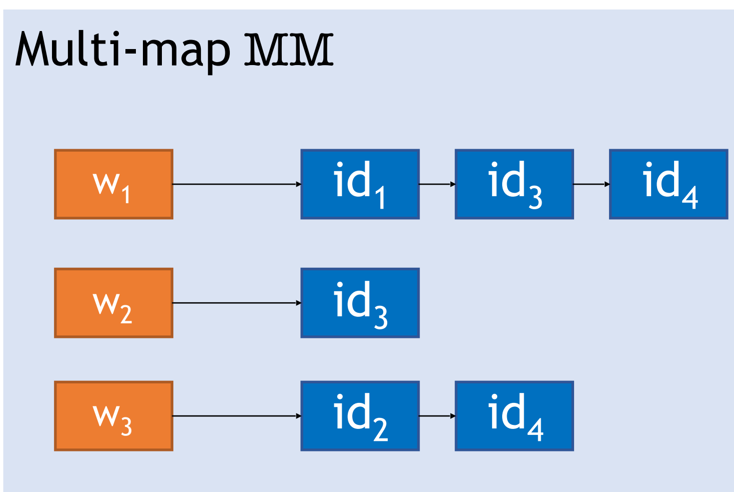


$$\mathcal{L}_Q = \Big(\cdot, \mathsf{rlen}\Big) \quad \text{s.t.} \quad \mathsf{rlen}\Big(\mathsf{DS}, q\Big) = |\mathsf{ans}|$$

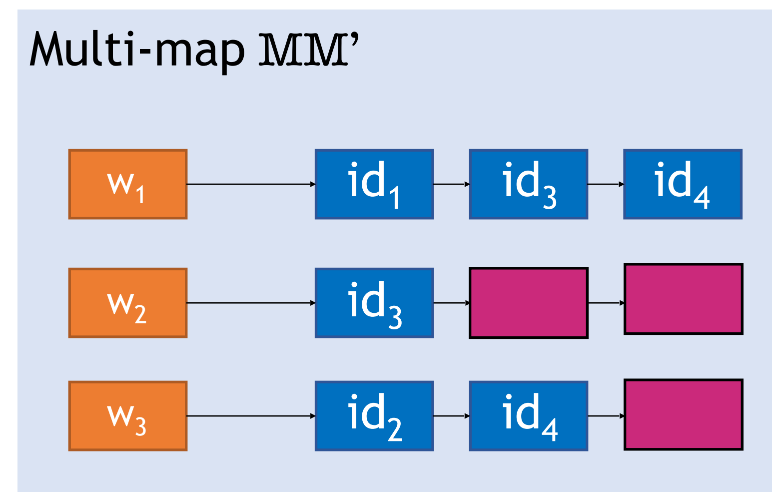
Naive Approaches to Hide Volume

Through Naive Padding

STE'$_{MM}$ $\Lambda' = \left(\mathcal{L}_S, \mathcal{L}_Q = \mathsf{qeq}\right)$



Adding Dummies



STE$_{MM}$

$\Lambda = \left(\mathcal{L}_S, \mathcal{L}_Q = (\mathsf{qeq}, \mathsf{rlen})\right)$

e.g., [CGKO06], [CK10], [CJJJKRS14]

Naive Approach to Hide Volume

Through Naive Padding

✖ • Query complexity

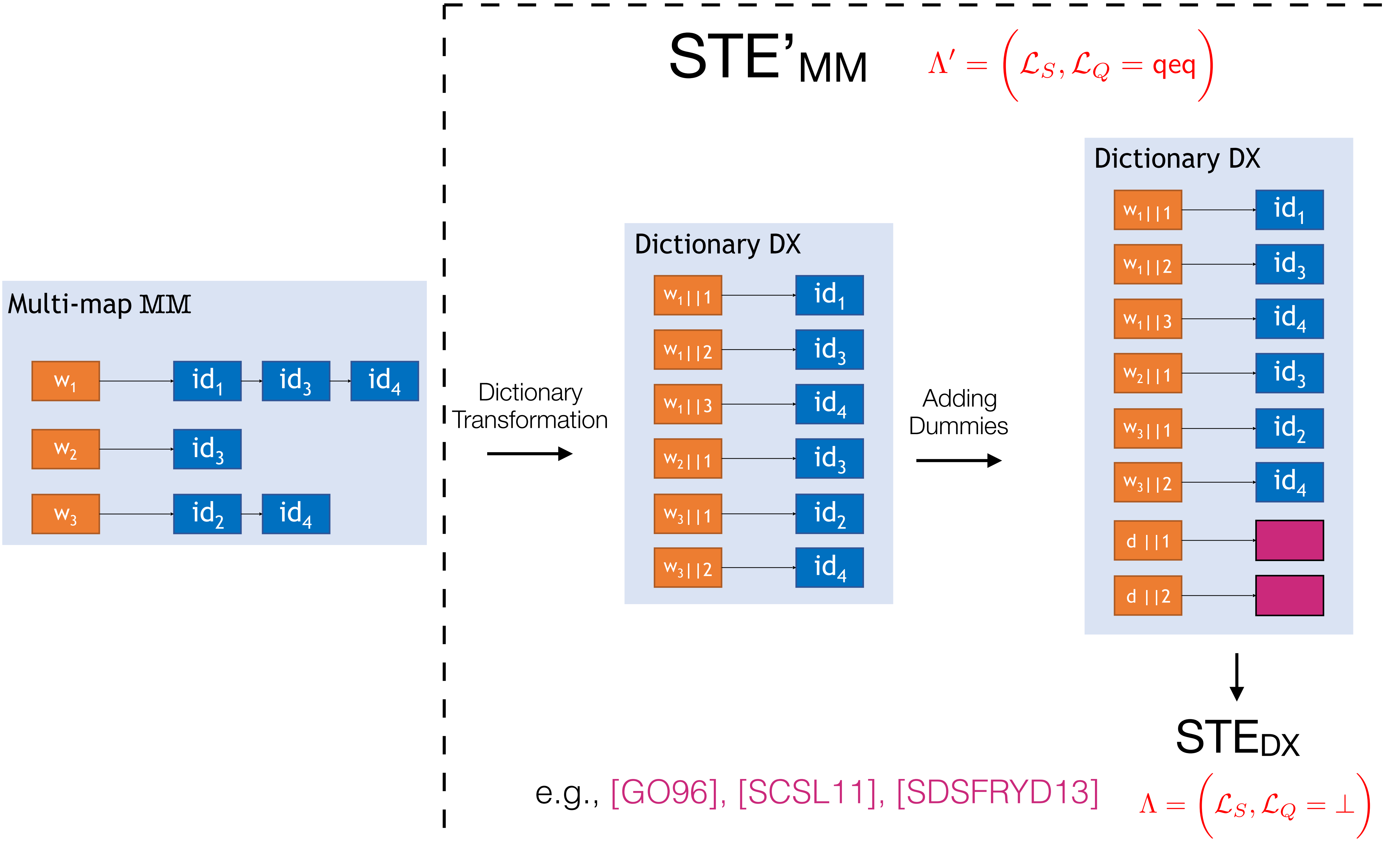
$$O(\max_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell])$$

✖ • Storage complexity

$$O(\#\mathbb{L}_{\mathsf{MM}} \cdot \max_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell])$$

☑ • Non-interactive

Naive Approach to Hide Volume
Through Leakage-Free Dictionary



Naive Approach to Hide Volume

Through Leakage-Free Dictionary (w/ [SDSFRYD13])

- ✖ Query complexity

$$O\left(\max_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell] \cdot \log^2\left(\sum_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell]\right)\right)$$

- ☑ Storage complexity

$$O\left(\sum_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell]\right)$$

- ✖ Interactive

Q: can we achieve the best of both worlds?

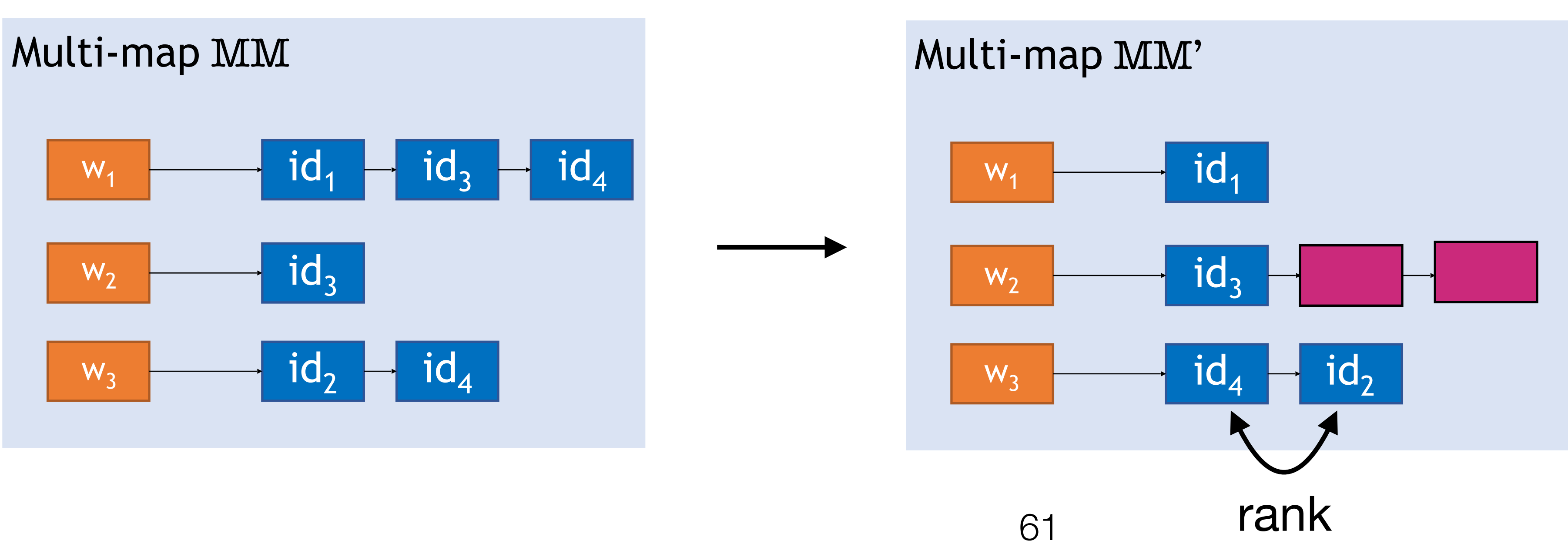
Contributions

- Pseudo-Random Transform (PRT)
- Volume Hiding Multi-Map Encryption scheme (VLH)
- Densest-Subgraph Transform (DST)
- Advanced Volume Hiding Multi-Map Encryption scheme (AVLH)
- Dynamism

Pseudo-Random Transform (PRT)

- Pseudo-random function $F : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^{\log \nu}$
- Minimum response length λ
- Replace the response length of ℓ by $\lambda + F_K(\ell \| \#\mathsf{MM}[\ell])$
  - Truncate if $\lambda + F_K(\ell \| \#\mathsf{MM}[\ell]) \leq \#\mathsf{MM}[\ell]$
  - Pad if $\lambda + F_K(\ell \| \#\mathsf{MM}[\ell]) > \#\mathsf{MM}[\ell]$
- Rank the response identities

E.g., $\lambda = 1$ and $\nu = 3$



$$F_K(w_1 \| 3) = 0$$
$$F_K(w_2 \| 1) = 2$$
$$F_K(w_3 \| 2) = 1$$

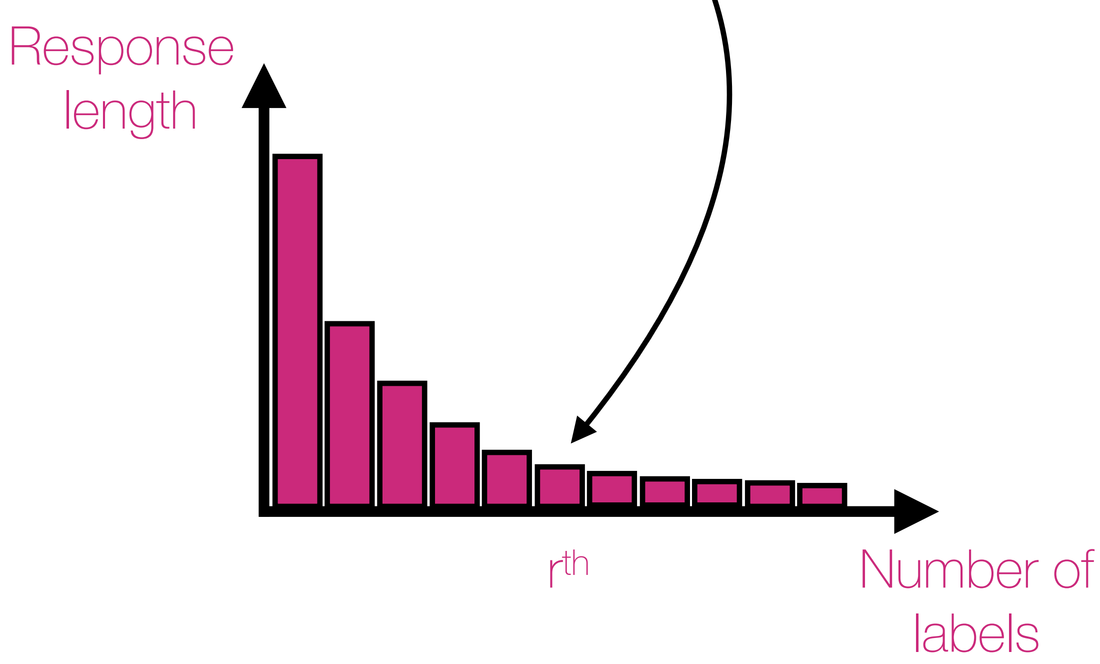
Q: what about the number of truncations and storage overhead?

Pseudo-Random Transform (PRT)

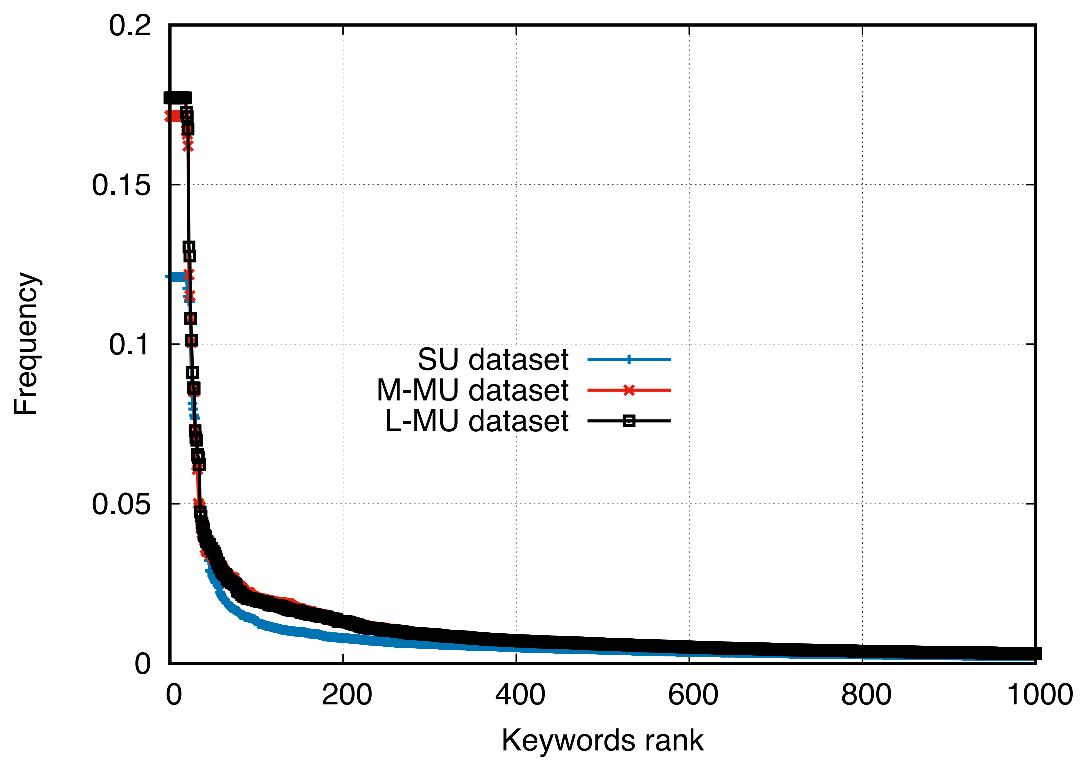
Zipf-Distributed MM

A MM is Zipf-distributed if the r^th response has length:

$$\frac{1}{r \cdot H_{\#\mathbb{L}_{\mathsf{MM}},1}} \cdot \sum_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell]$$



- Common in real-world datasets [Zipf35], [CCKS07]
- Ex: Enron 0.5M emails (2004)



Pseudo-Random Transform (PRT)

Analysis

- Let α be the storage reduction multiplicative factor
- If for $1/2 < \alpha < 1$, then with probability at least
  - $1 - \exp\Big(- \#\mathbb{L}_{\mathsf{MM}} \cdot (2\alpha - 1)^2/8 \Big)$ the size of the MM is at most

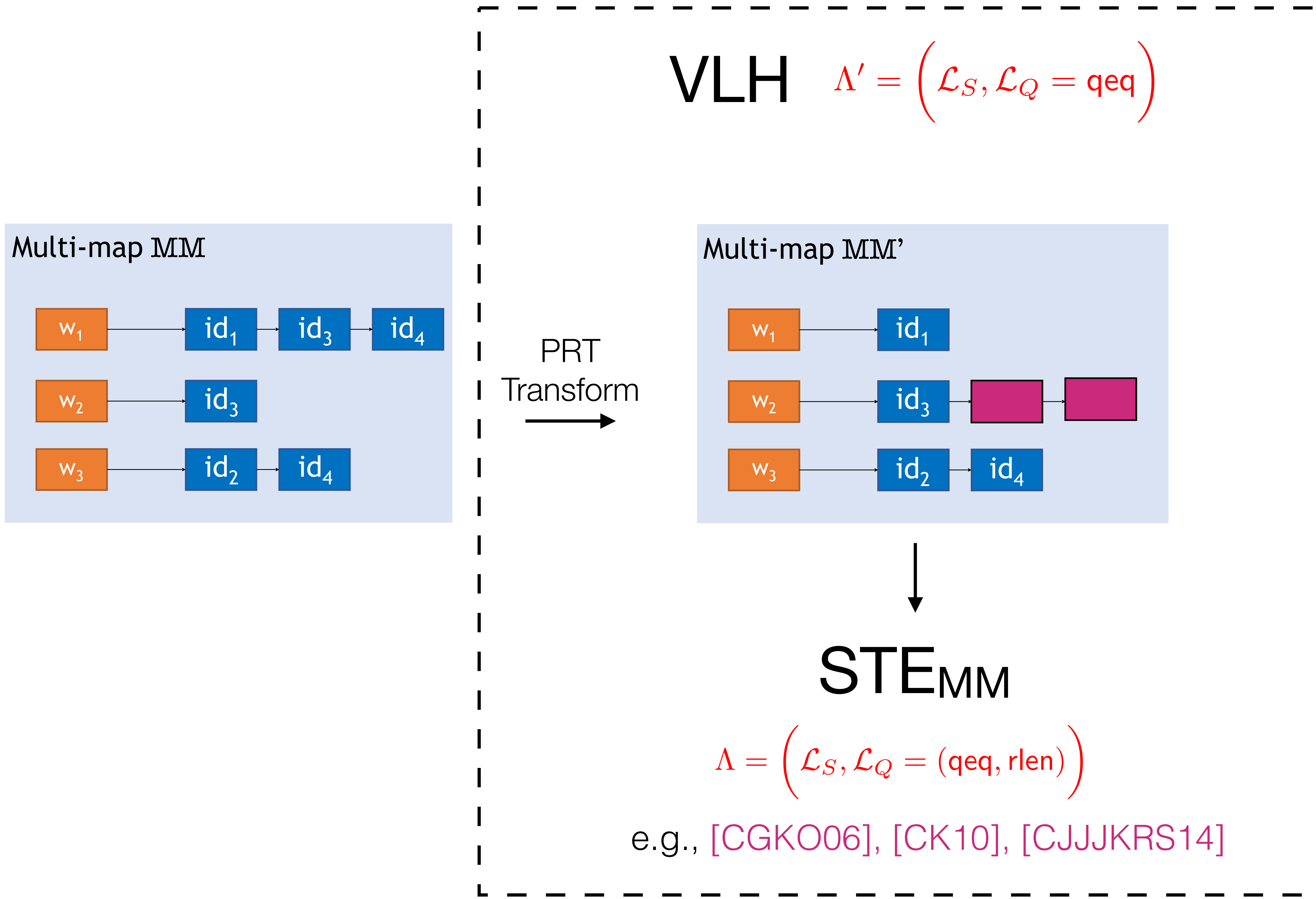
$$\alpha \cdot \#\mathbb{L}_{\mathsf{MM}} \cdot \max_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell]$$

  - $1 - \exp\Big(- 2\#\mathbb{L}_{\mathsf{MM}} \cdot \log^2(\#\mathbb{L}_{\mathsf{MM}}) \Big)$ the number of truncations is at most

$$\frac{1}{\log(\#\mathbb{L}_{\mathsf{MM}})} \cdot \#\mathbb{L}_{\mathsf{MM}}$$

Volume Hiding EMM (VLH)

Design



Volume Hiding EMM (VLH)

Analysis (with standard EMMs)

- Query complexity (worst-case)

$$O(\lambda + \nu)$$

- Storage complexity

$$O(\lambda \cdot \#\mathbb{L}_{\mathsf{MM}} + \sum_{\ell \in \mathbb{L}_{\mathsf{MM}}} n_\ell) \qquad \text{s.t.} \qquad n_\ell \xleftarrow{\$} [\nu]$$

and w.h.p. when $1/2 < \alpha < 1$

$$O(\alpha \cdot (\nu - 1) \cdot \#\mathbb{L}_{\mathsf{MM}})$$

- Non-Interactive

- Lossy

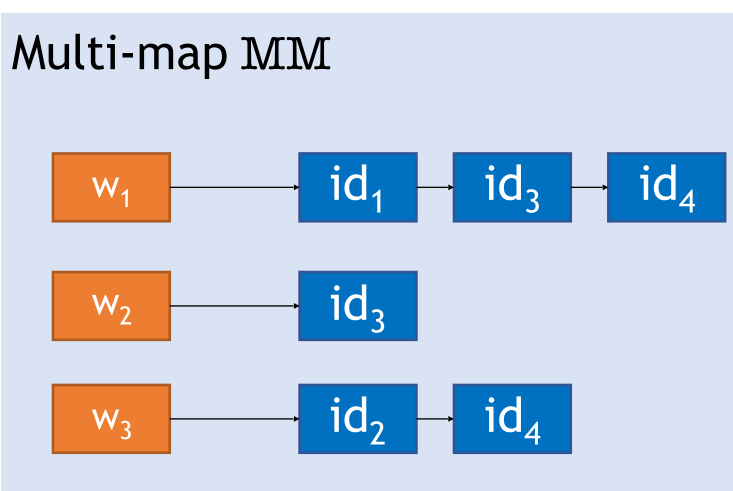
Densest-Subgraph Transform (DST)

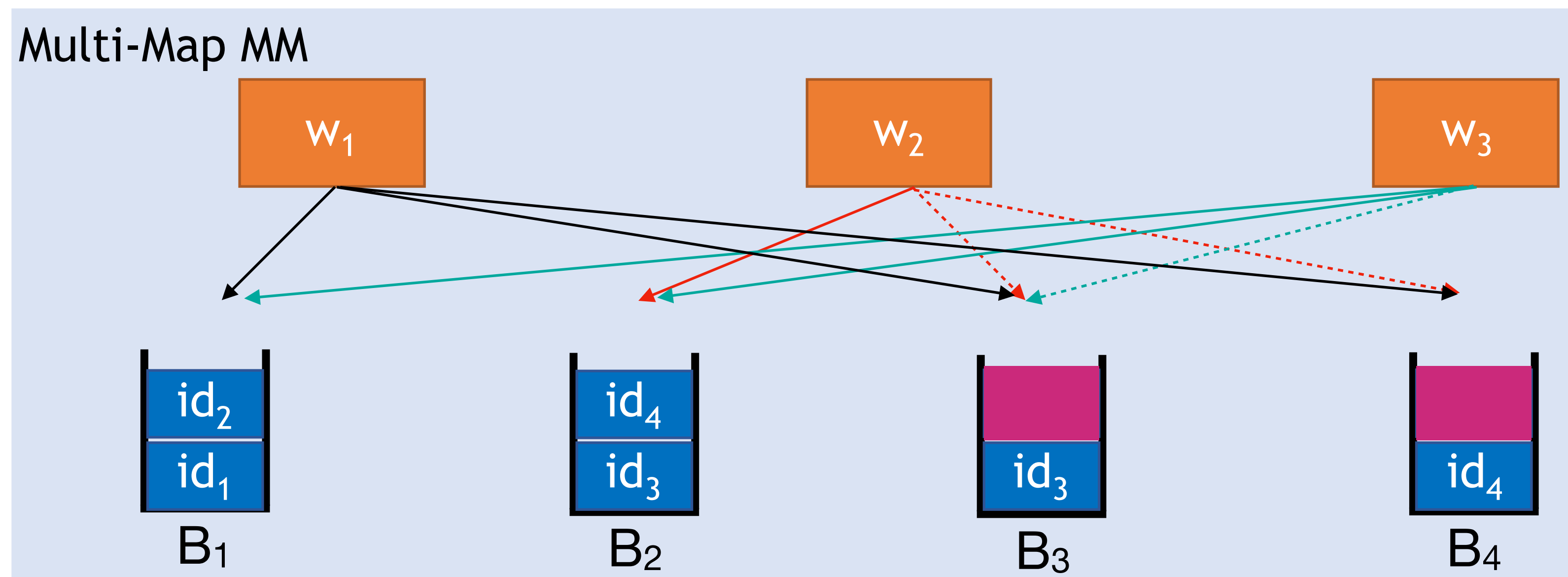
Overview

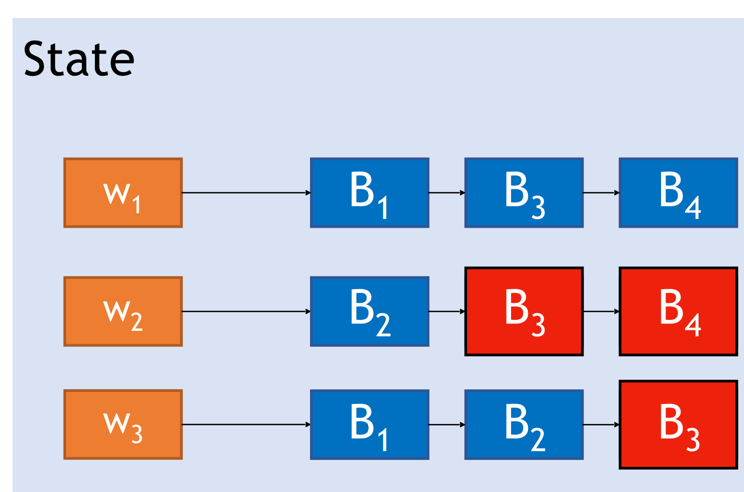
- We view a MM as a bi-partite graph $G = \Big((\mathbb{L}_{\mathsf{MM}}, \mathbf{B}), E \Big)$
  - top vertices: labels $\mathbb{L}_{\mathsf{MM}}$
  - bottom vertices: bins $\mathbf{B}$
- Given MM we build a Erdös-Rényi random graph
- All labels in MM have the same number of edges
- Goal: given a label, fetch the same number of bins
  - reduce the load of the bin

Densest-Subgraph Transform (DST)

Details







Storage overhead

$$O(\#\mathbb{L}_{\mathsf{MM}} \cdot \max_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell])$$

Similar to Naive Padding

Densest-Subgraph Transform (DST)

Details

Edge Generation

$\mathsf{rand}_{w_1} \xleftarrow{\$} \{0,1\}^k$

$\boldsymbol{F_K}(\mathsf{rand}_{w_1} \| 1) = 1$

$\boldsymbol{F_K}(\mathsf{rand}_{w_1} \| 2) = 3$

no collision

$\boldsymbol{F_K}(\mathsf{rand}_{w_1} \| 3) = 4$

$\mathsf{rand}_{w_2} \xleftarrow{\$} \{0,1\}^k$

$F_K(\mathsf{rand}_{w_2} \| 1) = 2$

$F_K(\mathsf{rand}_{w_2} \| 2) = 3$

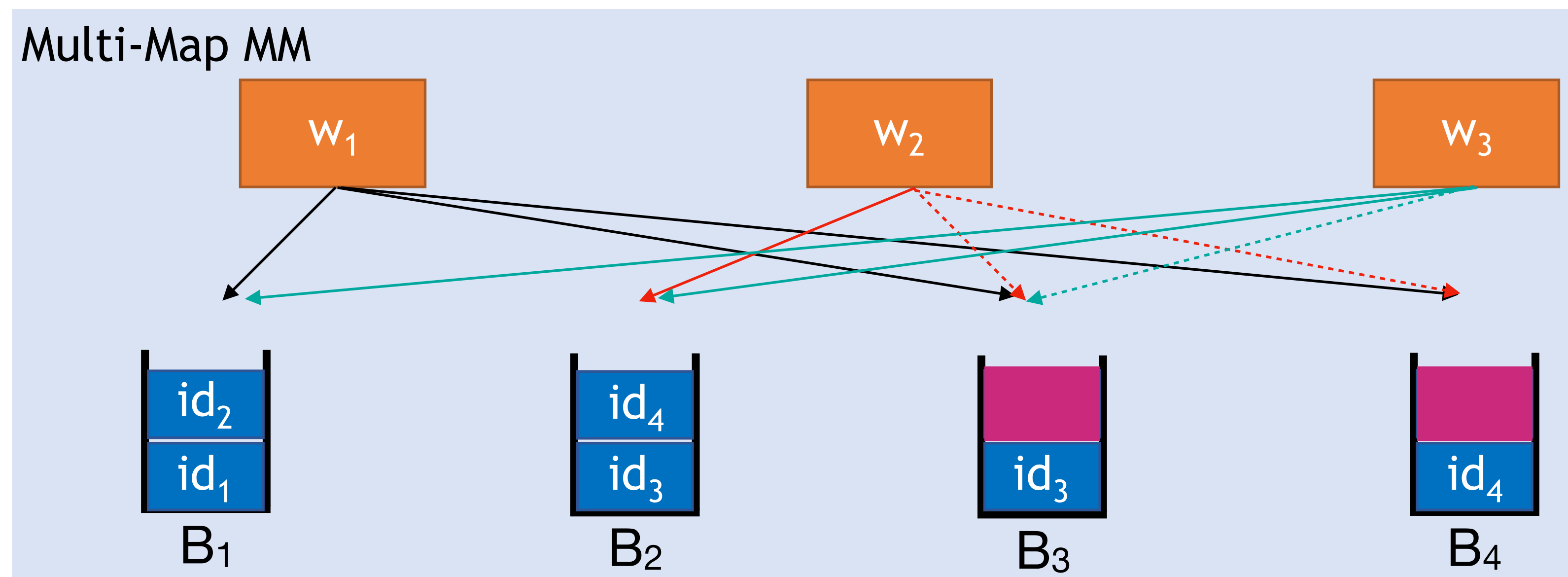
$F_K(\mathsf{rand}_{w_2} \| 3) = 4$

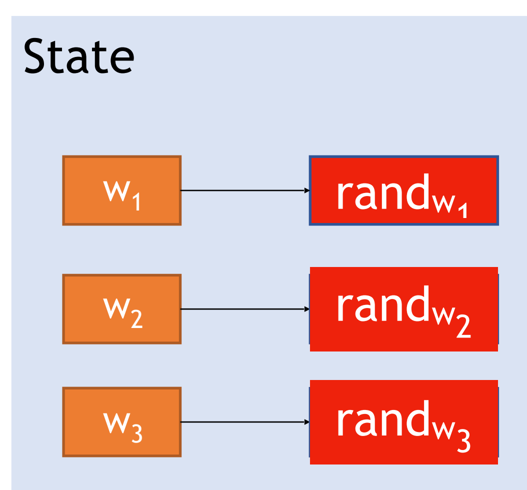
$\mathsf{rand}_{w_3} \xleftarrow{\$} \{0,1\}^k$

$F_K(\mathsf{rand}_{w_3} \| 1) = 1$

$F_K(\mathsf{rand}_{w_3} \| 2) = 2$

$F_K(\mathsf{rand}_{w_3} \| 3) = 3$





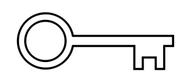
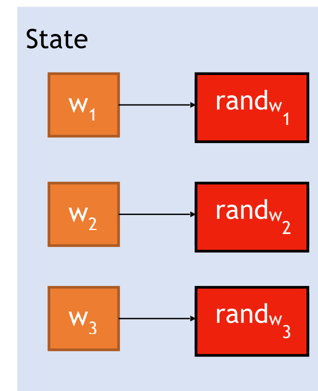
$$O(\#\mathbb{L}_{\mathsf{MM}})$$

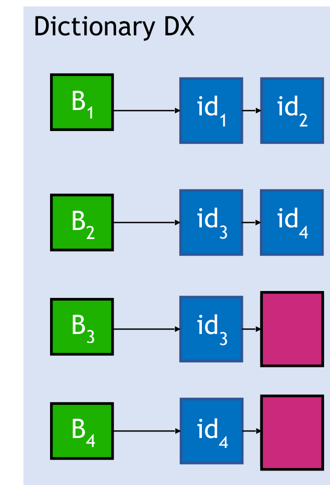
$$\ll$$

$$O(\#\mathbb{L}_{\mathsf{MM}} \cdot \max_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell])$$

Densest-Subgraph Transform (DST)

Details

- The output of DST is equal to:  , 

and 

- -

- To fetch a keyword w_1, retrieve rand_{w_1} from the state
- Compute bins' identifiers $F_K(\mathsf{rand}_{w_1}\|1)$, $F_K(\mathsf{rand}_{w_1}\|2)$, $F_K(\mathsf{rand}_{w_1}\|3)$
- Retrieve all the bins from the dictionary DX

Q: what about the load of a bin?

Densest-Subgraph Transform (DST)

Analysis

With probability at least $1 - \varepsilon$, the load of a bin is

$$\frac{N}{n} + \frac{\ln(1/\varepsilon)}{3}\left(1 + \sqrt{1 + \frac{18N}{n \cdot \ln(1/\varepsilon)}}\right)$$

where $N = \sum_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell]$

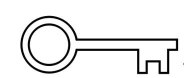
The size of the transformed multi-map MM is $O(N)$

The size of the state is $O(\#\mathbb{L}_{\mathsf{MM}}) \lll O(N)$

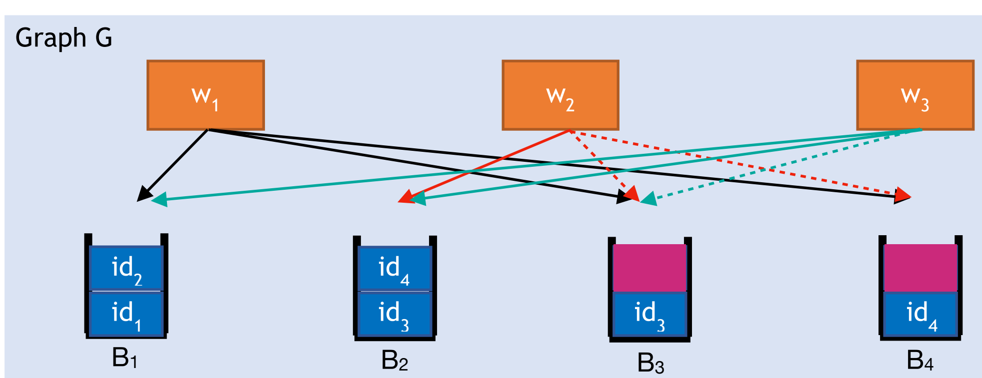
Advanced Volume-Hiding EMM (AVLH)

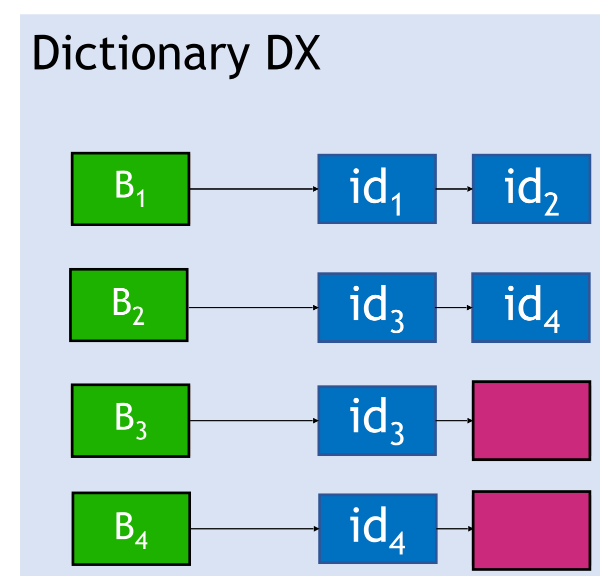
Setup (1)

$$\mathrm{Setup}\left[1^k, \text{ Multi-map MM}\right]$$

1. $\mathrm{DST}\left[1^k, \text{ Multi-map MM}\right] \longrightarrow \Big[$  $_1$, State, Graph G $\Big]$

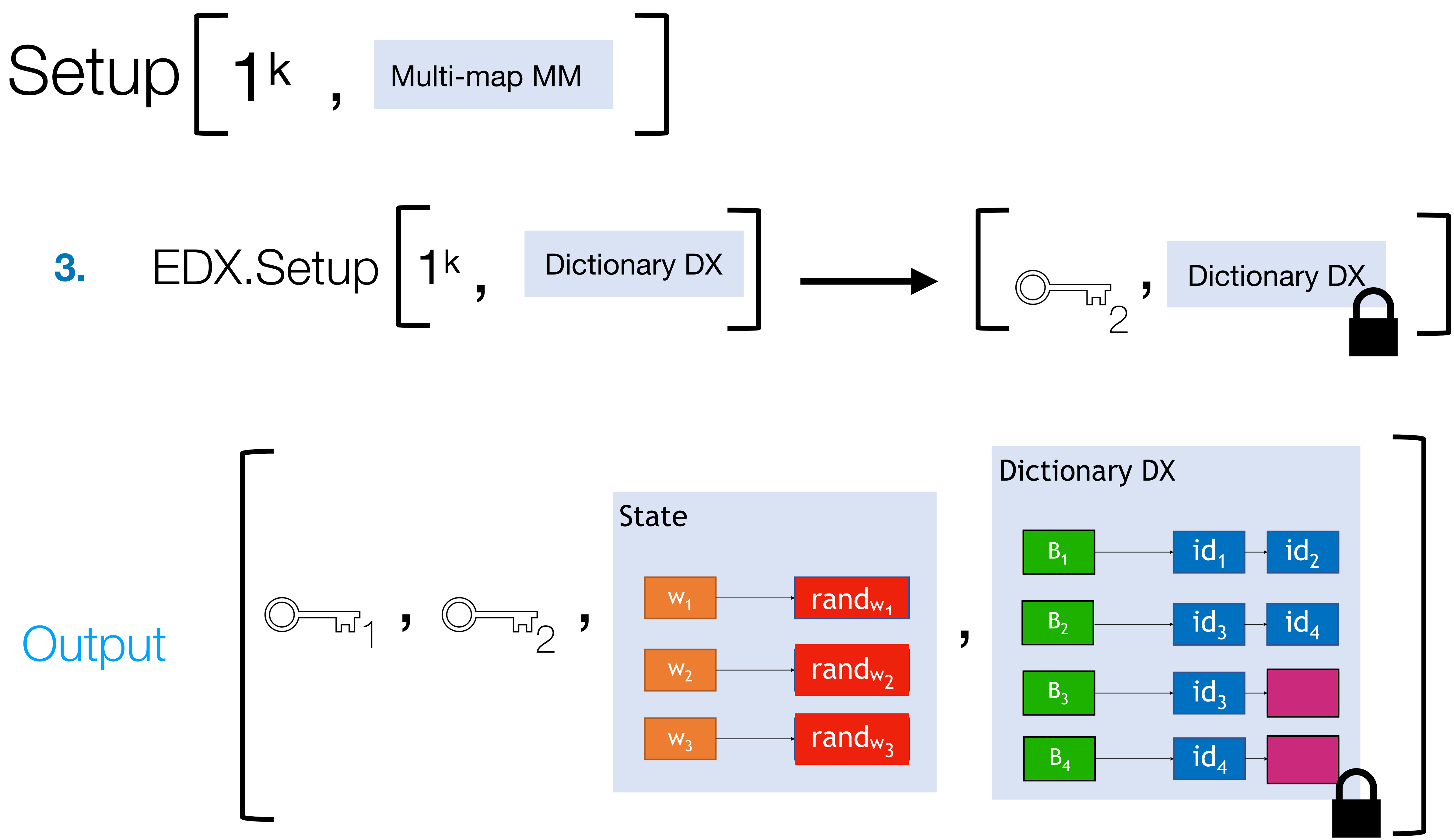
2.





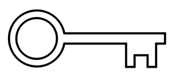
Advanced Volume-Hiding EMM (AVLH)

Setup (2)



Advanced Volume-Hiding EMM (AVLH)

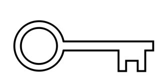
Token

Token [ , State , w_1]

1. Fetch rand_{w_1} from State

2. Compute $\mathsf{t} = \left(F_K(\text{rand}\|i)\right)_{i\in[3]}$

3. for each identifier i in t add to tk

EDX.Token [ , i] $\longrightarrow$ tk_t

Output tk

Advanced Volume-Hiding EMM (AVLH)

Query

Query [tk , Dictionary DX]

1. for each sub-token tk_i in tk

EDX.Query [tk_i , Dictionary DX] $\longrightarrow$ ct_i

Output $ct = (ct_1, ct_2, ct_3)$

Advanced Volume Hiding EMM (VLH)

Analysis ([CGKO06])

- Query complexity w.h.p.

$$O\left(t \cdot \frac{N}{\#\mathbb{L}_{\mathsf{MM}} \cdot \mathsf{polylog}(\#\mathbb{L}_{\mathsf{MM}})}\right)$$

  where t is the maximum length and $N = \sum_{\ell \in \mathbb{L}_{\mathsf{MM}}} \#\mathsf{MM}[\ell]$

- Storage complexity w.h.p.

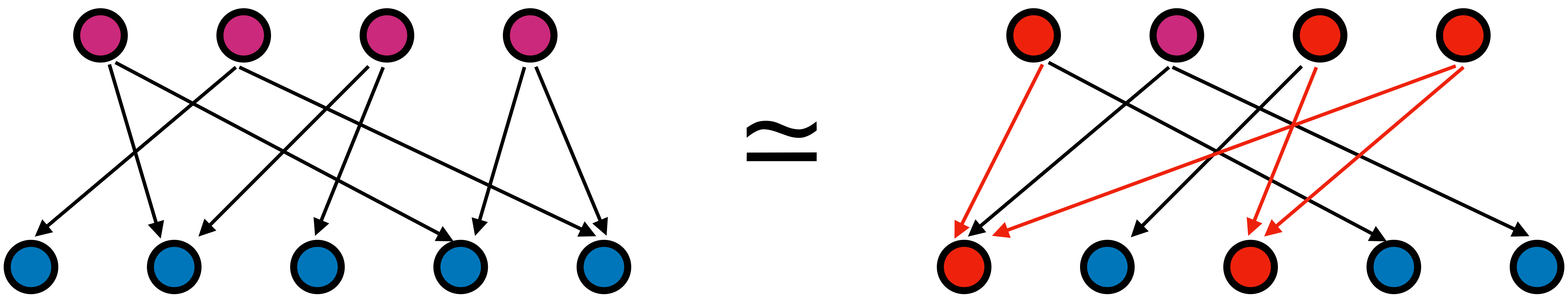
$$O(N)$$

- Non-Interactive

- Non-Lossy

Densest-Subgraph Transform (DST)

Improving Storage



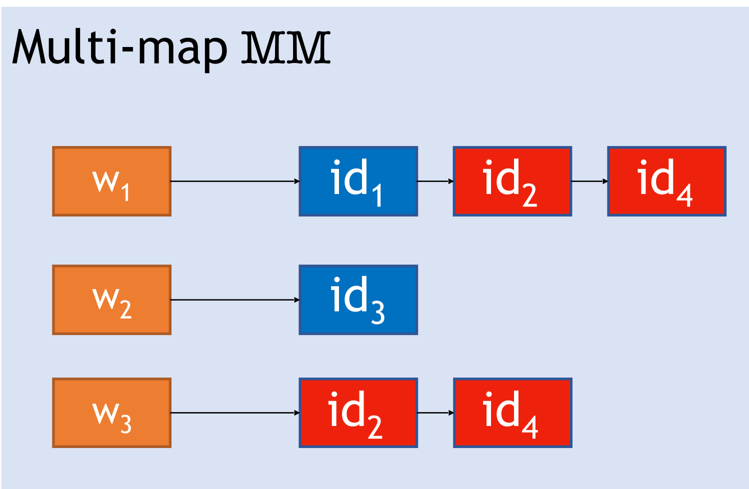
Erdös-Rényi graph

Erdös-Rényi graph with planted dense subgraph

Found applications in public-key cryptography [ABW10] and computational complexity of financial products [ABBG11]

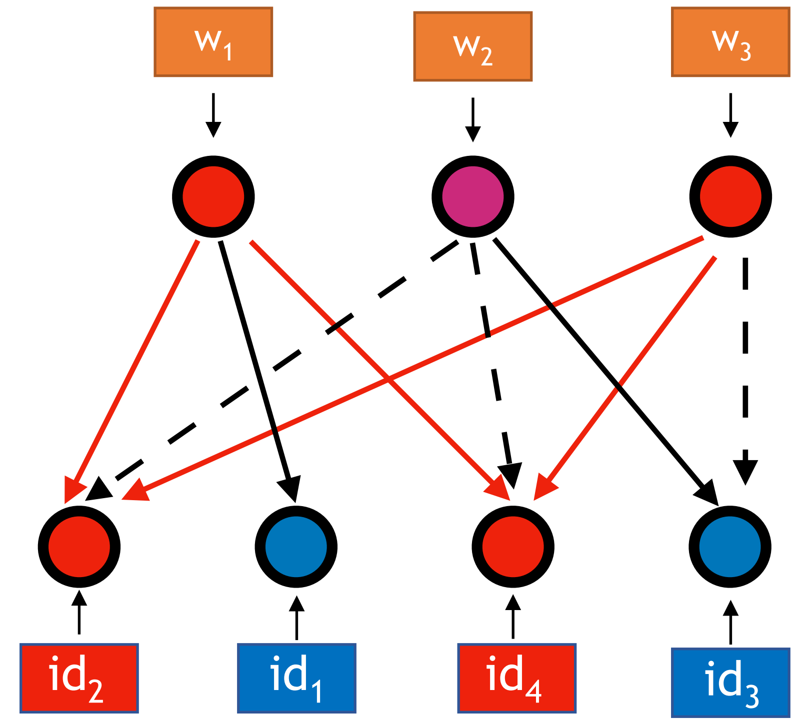
Densest-Subgraph Transform (DST)

Improving Storage



Concentrated MM: labels with non-empty intersection

id2 and id4 constitute the concentrated part



Add the concentrated part only once to the graph

Result: Reduce the load of bins

Densest-Subgraph Transform (DST)
Analysis

With probability at least $1 - \varepsilon$, the load of a bin is

$$\frac{N - N_{\text{DS}}}{n} + \frac{\ln(1/\varepsilon)}{3}\left(1 + \sqrt{1 + \frac{18(N - N_{\text{DS}})}{n \cdot \ln(1/\varepsilon)}}\right)$$

where N_{DS} is the size of the concentrated part.

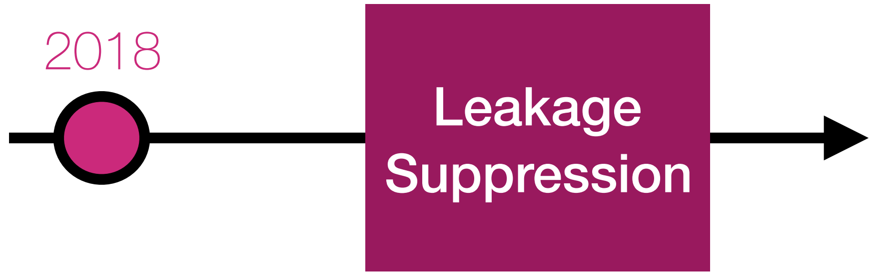
Instead of

$$\frac{N}{n} + \frac{\ln(1/\varepsilon)}{3}\left(1 + \sqrt{1 + \frac{18N}{n \cdot \ln(1/\varepsilon)}}\right)$$

Takeaways

Takeaways

- Introduce a new direction in encrypted search



- A general framework that suppresses the search pattern
- First solution to hide response-length pattern (volume pattern)
- A general compiler that makes any STE scheme rebuildable
- First scheme to leak at most the sequence response length (very hard to exploit)
- The first scheme that leaks (nothing)
- Introduces a new tradeoff: query latency vs. security

Takeaways

- Volume pattern has been recently leveraged as an attack vector [KKNO16], [GLMP18]
- Without trivial naive padding, hiding volume is extremely hard
- Hiding volume is an important step for leakage suppression

- The first non-trivial schemes that hide the volume pattern
  - VLH based on a new lossy *pseudo-random transform* (PRT)
  - AVLH based on a new non-lossy *densest-subgraph transform* (DST)

Takeaways

- Leveraging computational assumptions to suppress leakage
  - Intuitively it is hard to hide volume information theoretically without padding
  - Get around this leveraging computational assumptions
  - first to do so for any pattern, and for volume in particular
  - possibility to leverage computational assumptions to suppress other patterns
- Introducing a new tradeoff: correctness vs. security
- Hiding volume can help thwart many existing attacks: [IKK12], [CGPR15], [KKNO16], [LMP18], [GLMP18], [LMP19]

Thank you!