# SQL on Structurally-Encrypted Databases

Seny Kamara Tarik Moataz

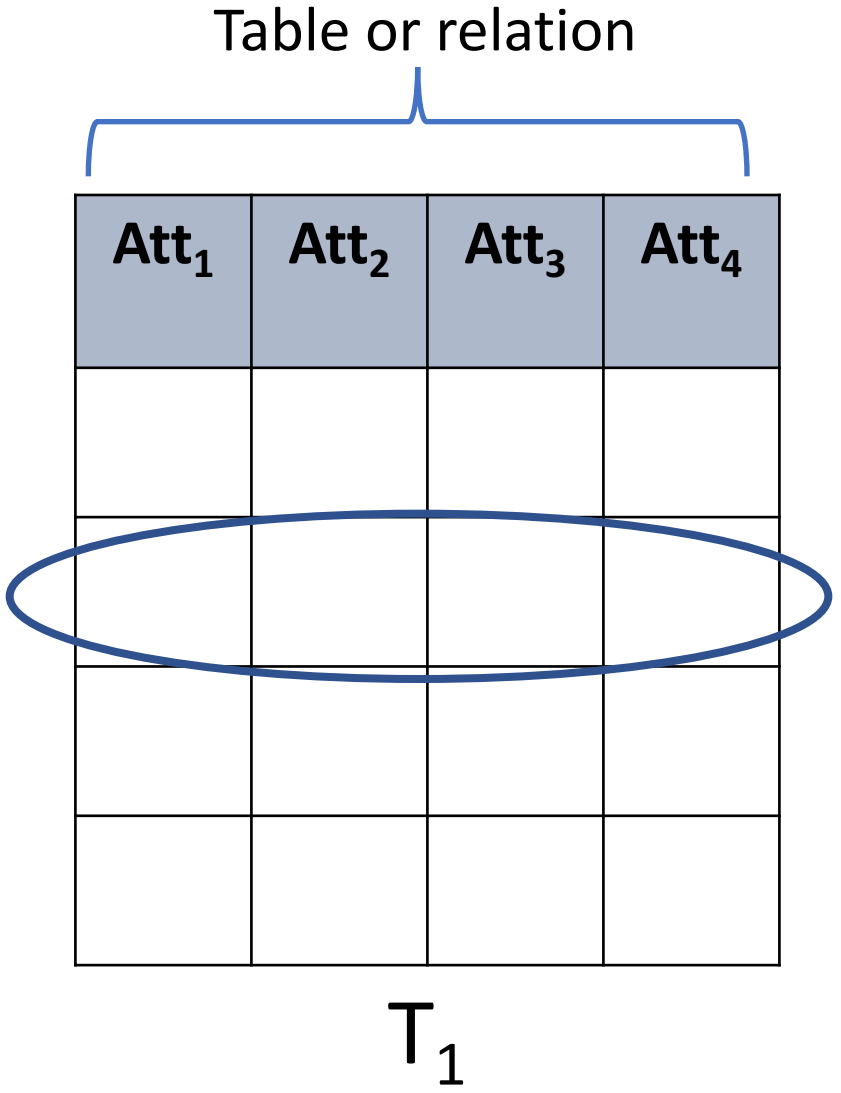
BROWN

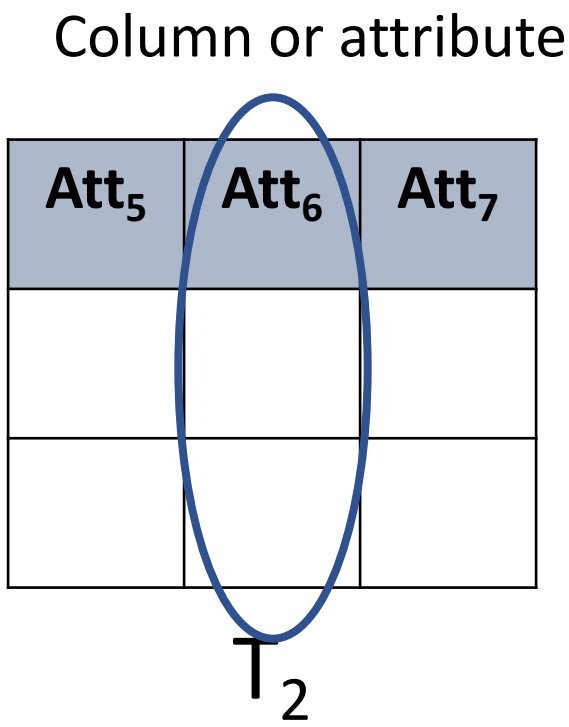
ENCRYPTED SYSTEMS LAB

# : What is a relational database?

# Relational DB

DB = Row or record



# Structured Query Language

- SQL is a language for querying relational DBs

$$\text{Select } attributes \text{ From } tables \text{ Where } condition,$$

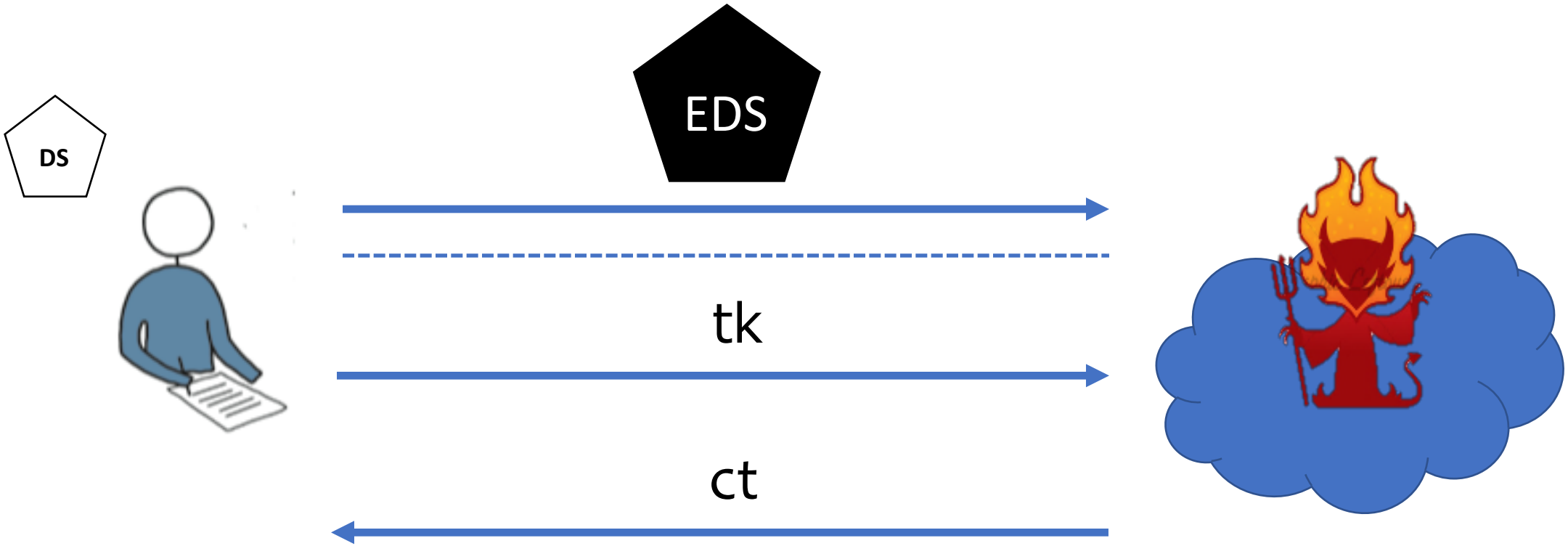
- Example:

**Select** (name, gender, height)
**From** ($T_2$, $T_8$)
**Where** (age = 36 AND zip = 10040 AND gender = F)

- SQL is the standard way to query a relational DB
  - Standard ANSI/ISO since 1986/1987

# What is Structured Encryption (STE)?

# Structured Encryption (STE) [CK10]

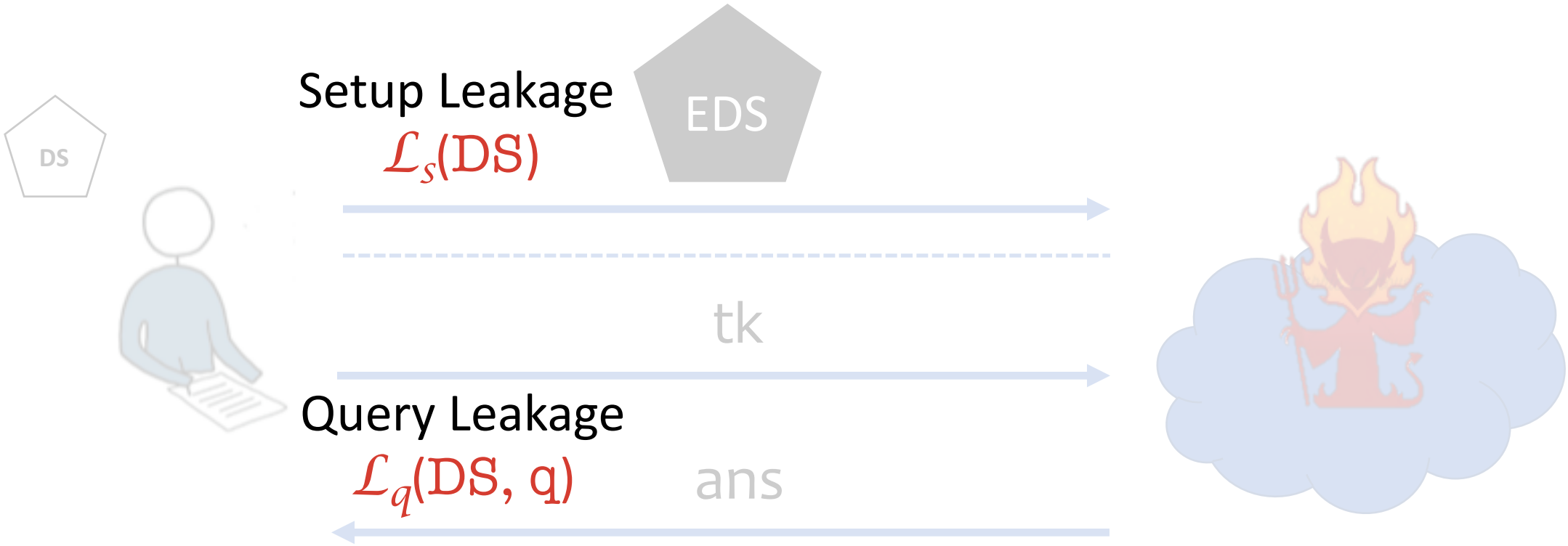



$\text{Setup}(1^k, \text{DS}) \longmapsto (\text{K}, \text{EDS})$

$\text{Token}(\text{K}, \text{q}) \longmapsto \text{tk}$

$\text{Query}(\text{EDS}, \text{tk}) \longmapsto \text{ct}$

# Structured Encryption (STE) [CK10]




$\mathrm{Setup}(1^k, \mathrm{DS}) \longmapsto (\mathrm{K}, \mathrm{EDS})$

$\mathrm{Token}(\mathrm{K}, \mathrm{q}) \longmapsto \mathrm{tk}$

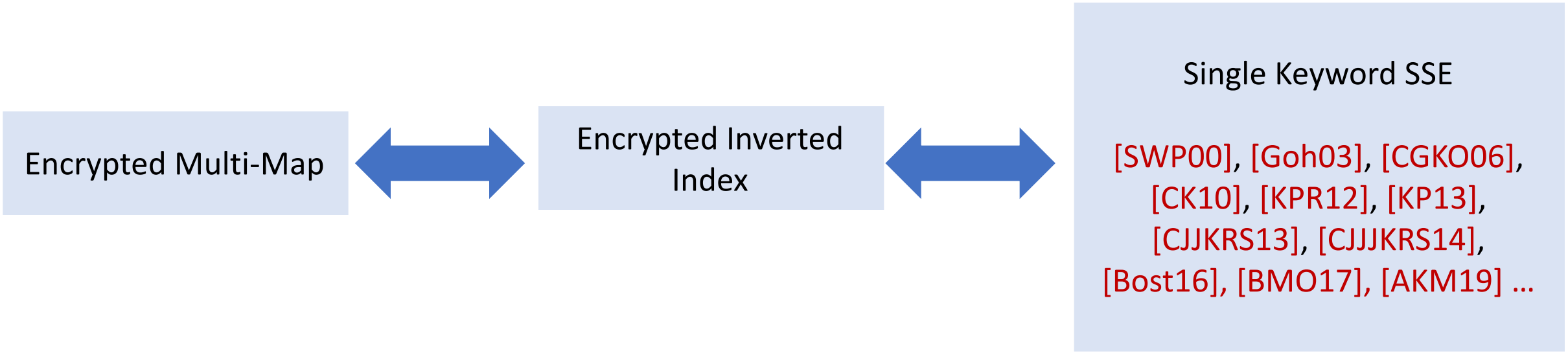
$\mathrm{Query}(\mathrm{EDS}, \mathrm{tk}) \longmapsto \mathrm{ans}$

# Structured Encryption (STE) [CK10]

We say that an STE is $(\mathcal{L}_S, \mathcal{L}_Q)$-secure if

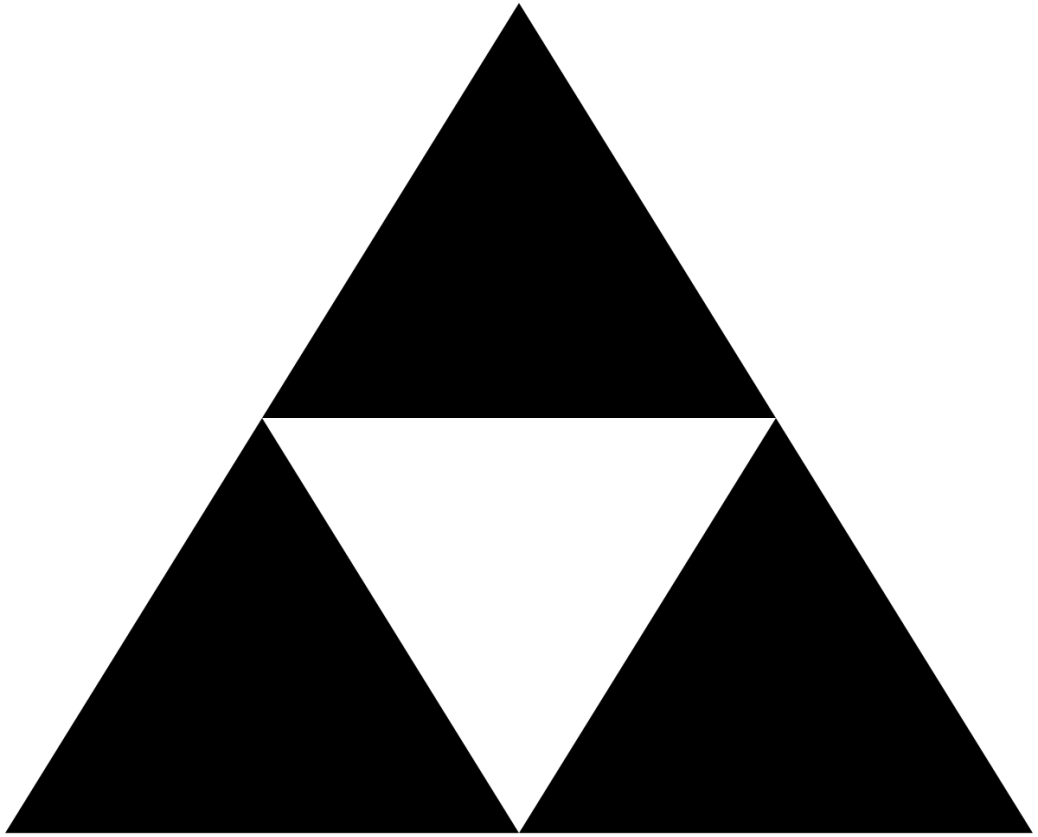
- It reveals no information about the structure beyond $\mathcal{L}_S$
- It reveals no information about the structure and queries beyond $\mathcal{L}_Q$

# Encrypted Multi-Maps [CK10]



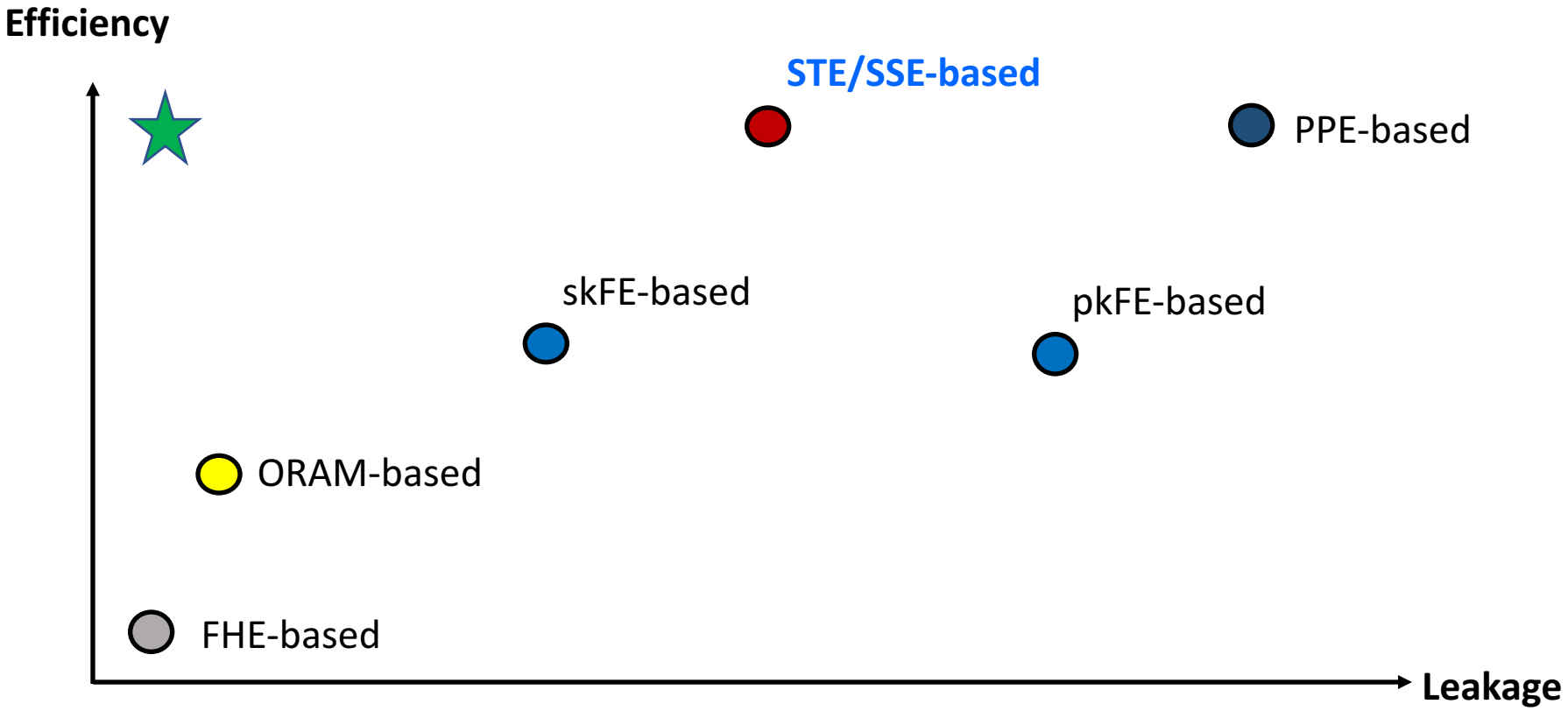
# : How can we encrypt a relational DB?

**Efficiency**

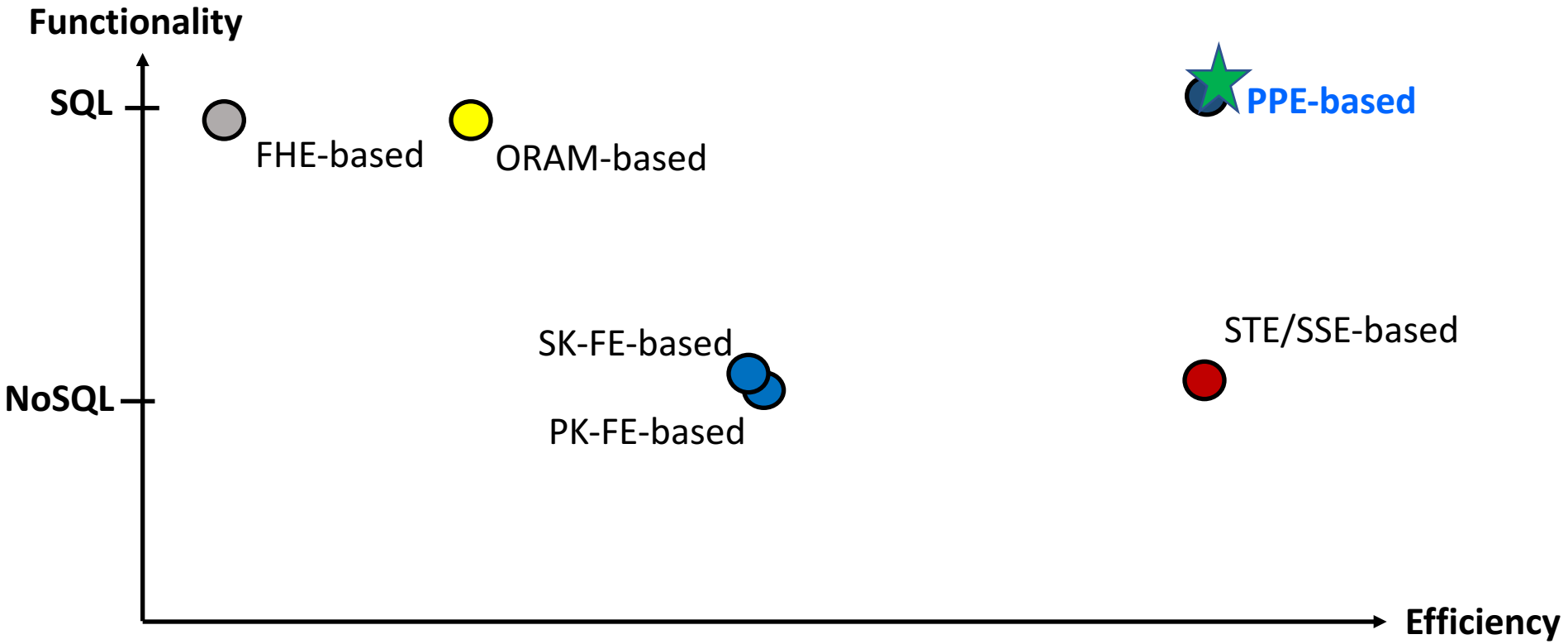


**Functionality** **Leakage**

# Tradeoffs: Efficiency vs. Security



# Tradeoffs: Functionality vs. Efficiency



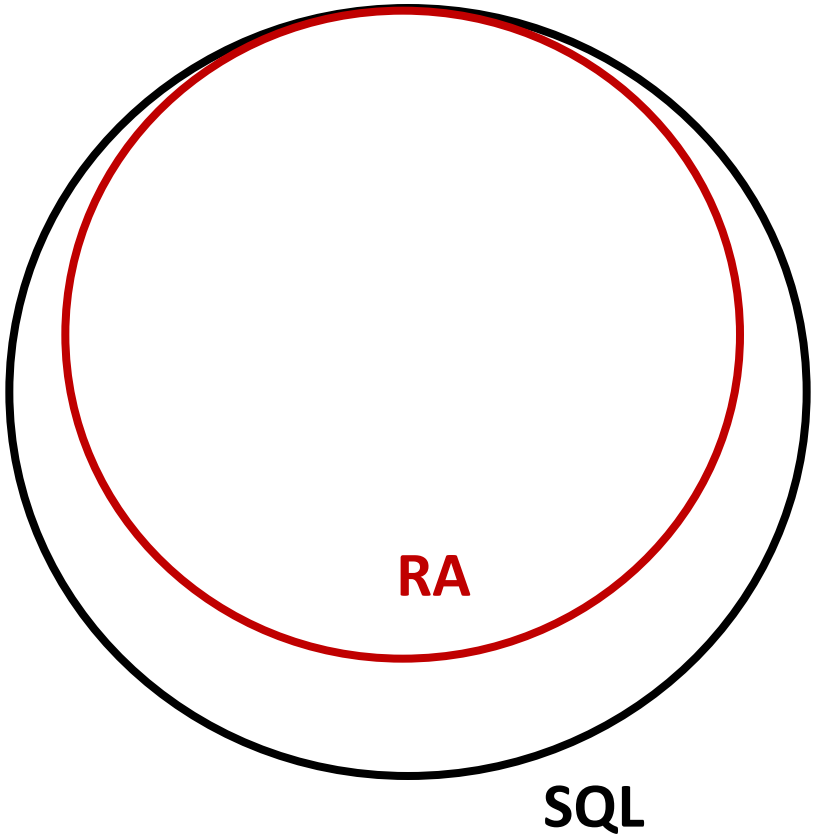
Q: Can we design an STE-based Relational EDB?

# Challenges

- No PPE so no plug-and-play solutions
- SQL is a declarative language
  - Where do we even start?
- SQL is complex
  - Combination of many basic query types
  - Most STE schemes handle a single type queries
- SQL is “constructive”
  - STE has been optimized for “lookup-type” queries

# Ch. #1: Declarative => Procedural

- Relational algebra [Codd70]
  - Set of operations on relations/tables
  - Union
  - Difference
  - Selection
  - Projection
  - Cross product
  - Join (many kinds)
  - …



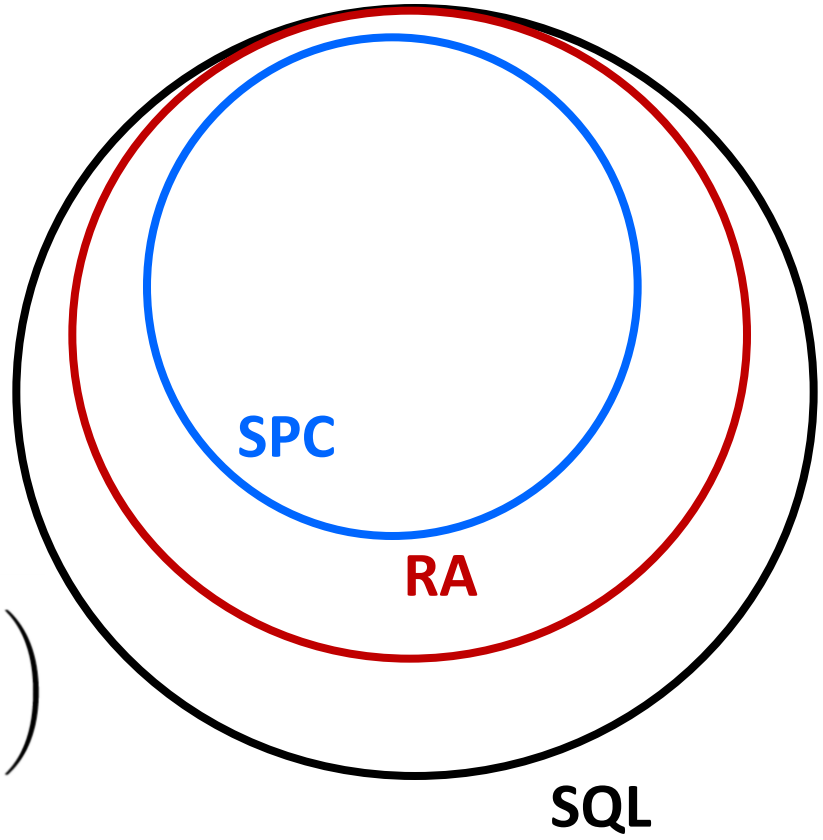
# Ch. #2: Complex => Simple

- SPC algebra [Chandra-Merlin77]
  - Selection, Projection, Cross product
  - Equivalent to *Conjunctive* SQL queries

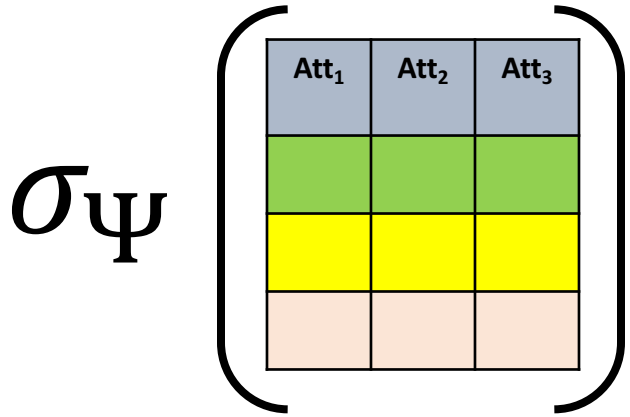
Select $attributes$ From $tables$ Where $(\mathsf{a}_1 = X_1 \wedge \cdots \wedge \mathsf{a}_\ell = X_\ell)$,

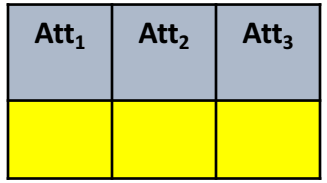
  - Any SPC query can be written in a Normal Form:

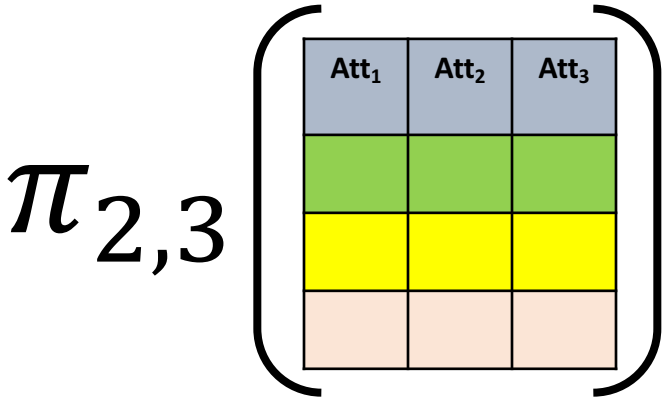
$$\pi_{\mathsf{a}_1,\cdots,\mathsf{a}_h}\left([a_1] \times \cdots [a_f] \times \sigma_\Psi(\mathsf{T}_{i_1} \times \cdots \times \mathsf{T}_{i_t})\right)$$

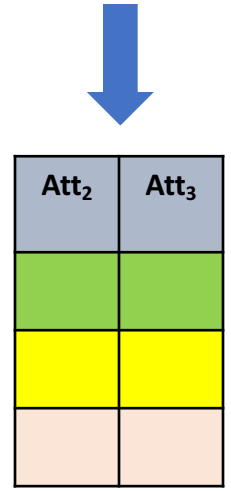


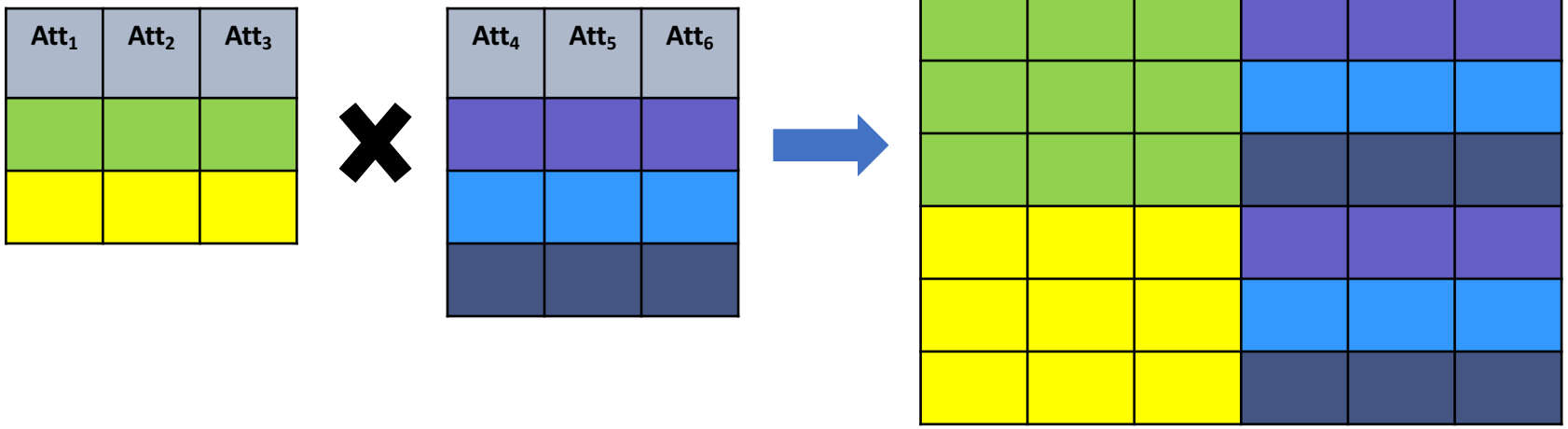
# Select, Project, Cross Product



# Our Goal

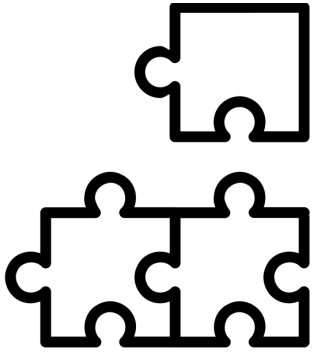
SQL => SPC => NF

$STE_K$ ( $Att_1$ $Att_2$ | $Att_1$ $Att_3$ )

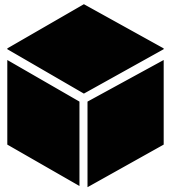
tk

$Enc_K$ ( $Att_2$ $Att_3$ )

# Our Results

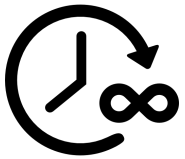


- **SPX**: Encrypted Relational Database
  - First STE scheme for relational DBs
  - Handles non-trivial subset of SQL
  - Sub-linear search and storage complexity (optimal under certain conditions)
  - from any single-keyword SSE



- **SPX⁺**: dynamic SPX
  - Only row addition and deletion
  - from any dynamic single-keyword SSE
  - Sub-linear search and storage complexity (optimal under certain conditions)

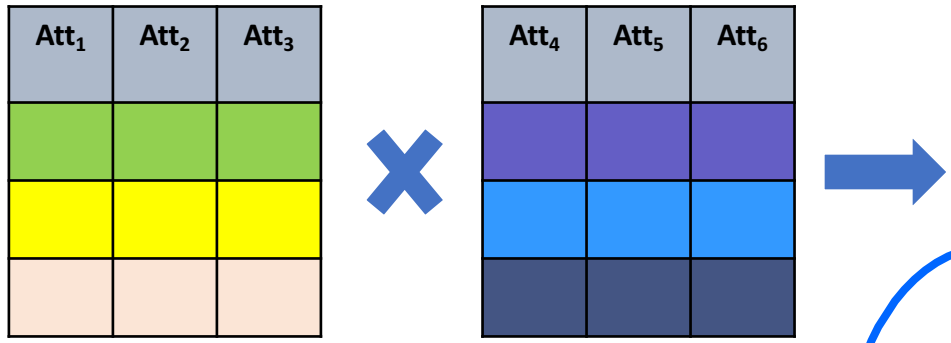
- **FP-SPX⁺**: forward-private dynamic SPX
  - poly-logarithmic overhead for updates



# A: Naïve STE-based Relational EDB

# Naïve SPC Algorithm

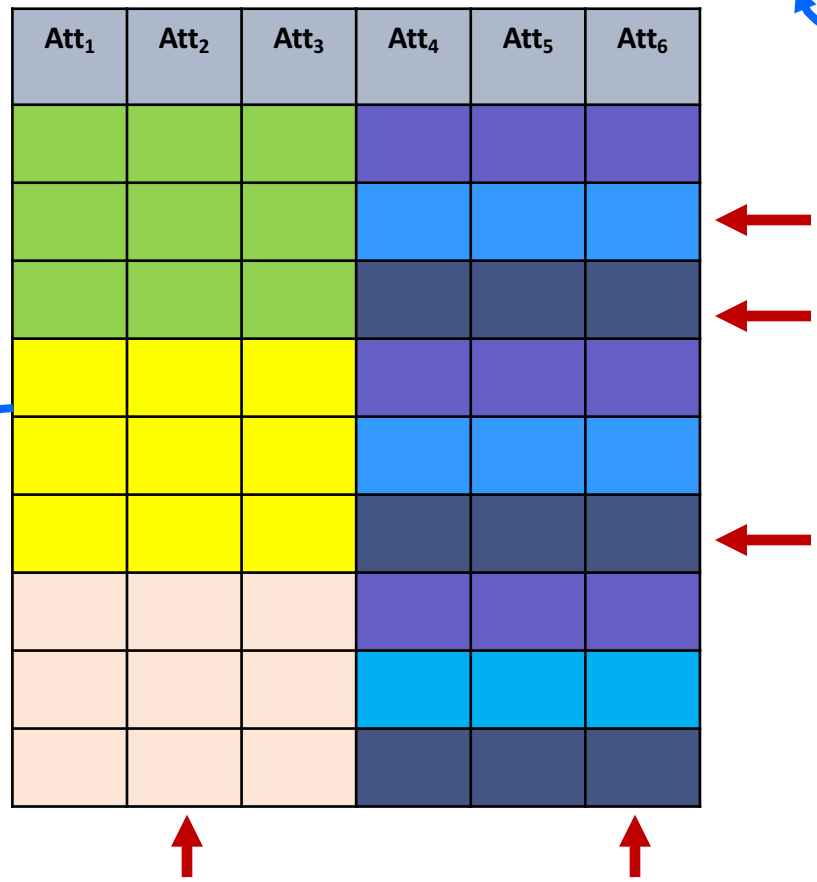
$$\pi_{\mathsf{a}_1,\cdots,\mathsf{a}_h}\left([a_1]\times\cdots[a_f]\times\sigma_{\Psi}(\mathsf{T}_{i_1}\times\cdots\times\mathsf{T}_{i_t})\right)$$

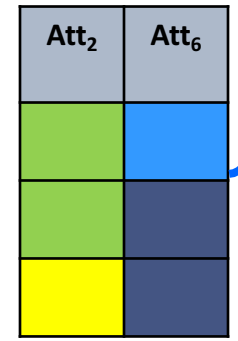






$$\mathcal{Q} = (m^{h'} \cdot \Pi_{i=1}^{\ell} \# R_i) \times h$$



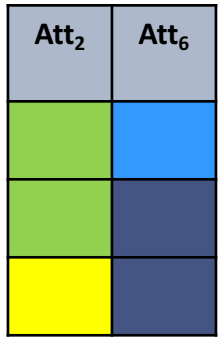




$$m^t \times \left(\sum_{i=1}^{t} \|\mathsf{T}_i\|\right)$$

# Sub-Linear SPC Algorithm

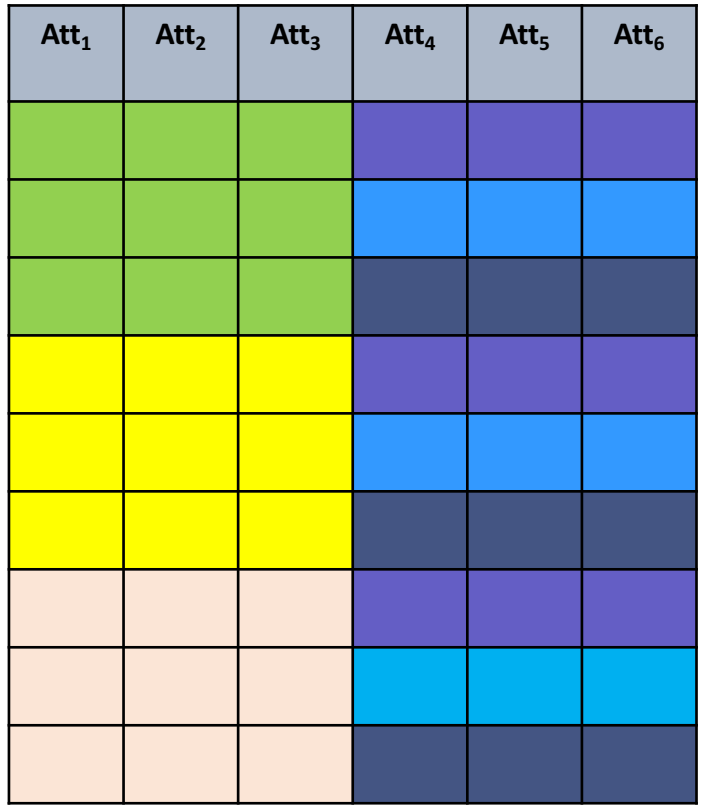
- Ideally linear in output size:

$$\mathcal{Q} = (m^{h'} \cdot \Pi_{i=1}^{\ell} \# R_i) \times h$$




- Less than cross product size:

$$m^t \times \left( \sum_{i=1}^{t} \| \mathsf{T}_i \| \right)$$

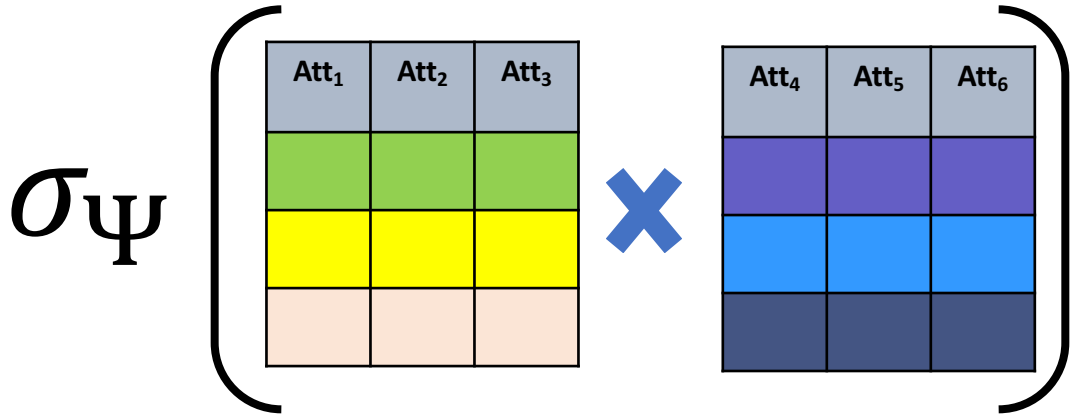


Q: Can we achieve sub-linear STE-based EDB?

# SPX Overview

- Step 1. Heuristic normal form (HNF) instead of the standard normal form
  - Avoid naïve Cartesian product by a “push select through product” method
- Step 2. New (plaintext) data structure that supports HNF
  - Different representations of the database to handle different SPC operators
- Step 3. Encrypted structure that supports HNF queries
  - Chaining technique with a better control of leakage
  - From any single-keyword SSE schemes

# Step 1: Heuristic Normal Form (1)

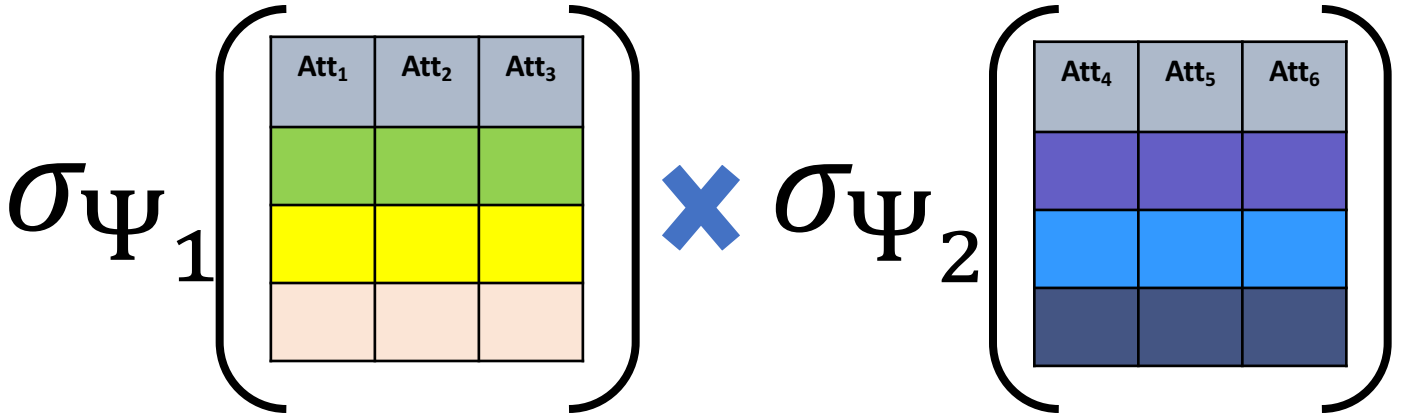



More complicated

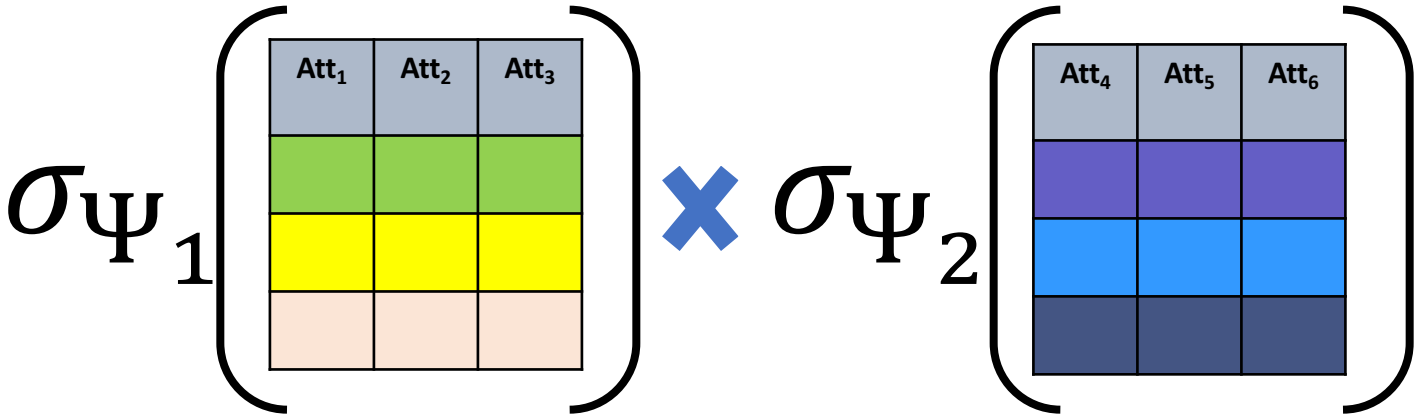
- Correlated/non-correlated
- Different types of select

$$\Psi = \Psi_1 \wedge \Psi_2$$

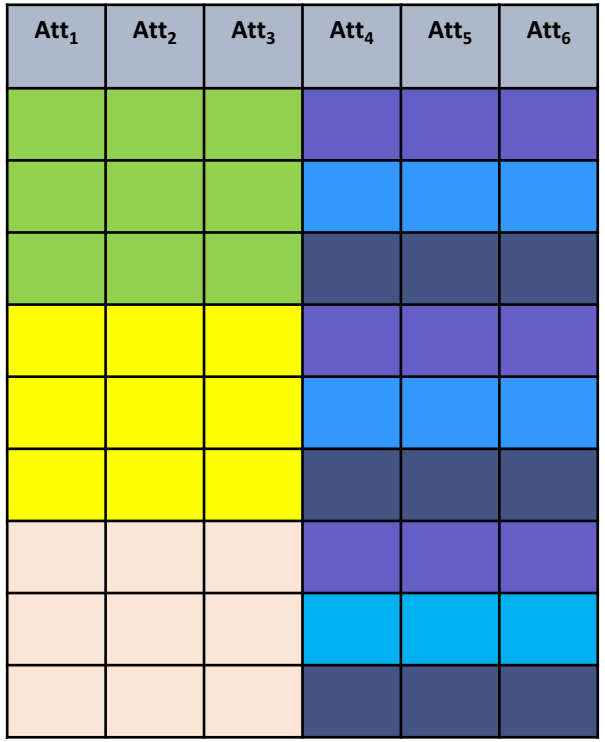
Push Select through Product

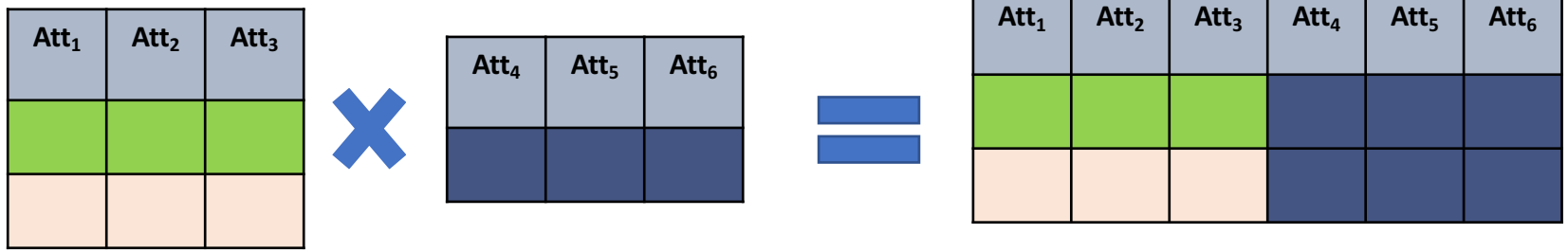


# Step 1: Heuristic Normal Form (2)



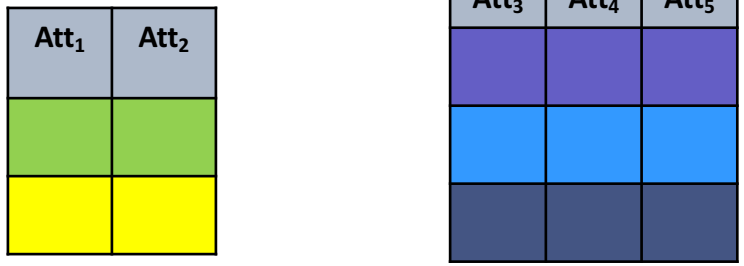

Size Overhead $\ll$



# Step 2: Database representations

DB =




$T_1$ $T_2$

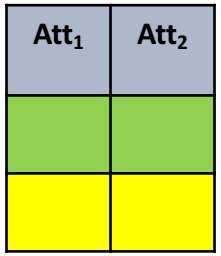
Row representation

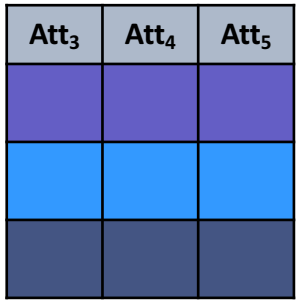
Column representation

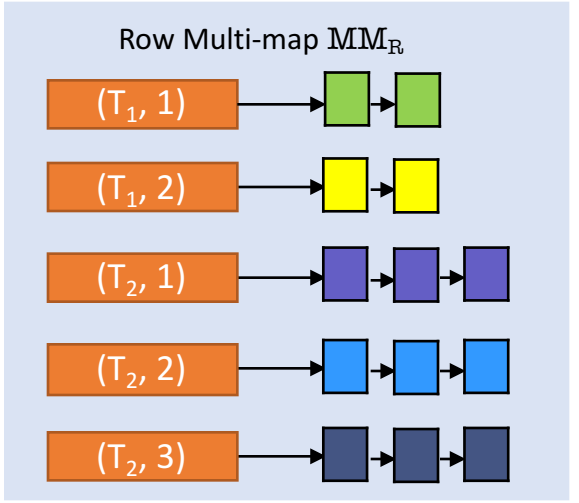
Value representation

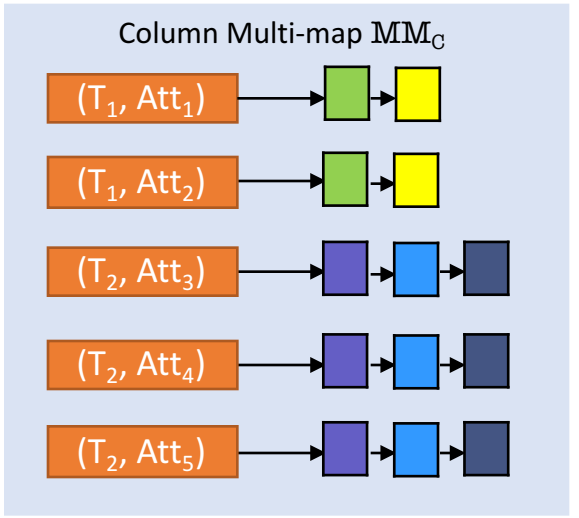
Cross-value representation

# Step 2: Row / Column representation



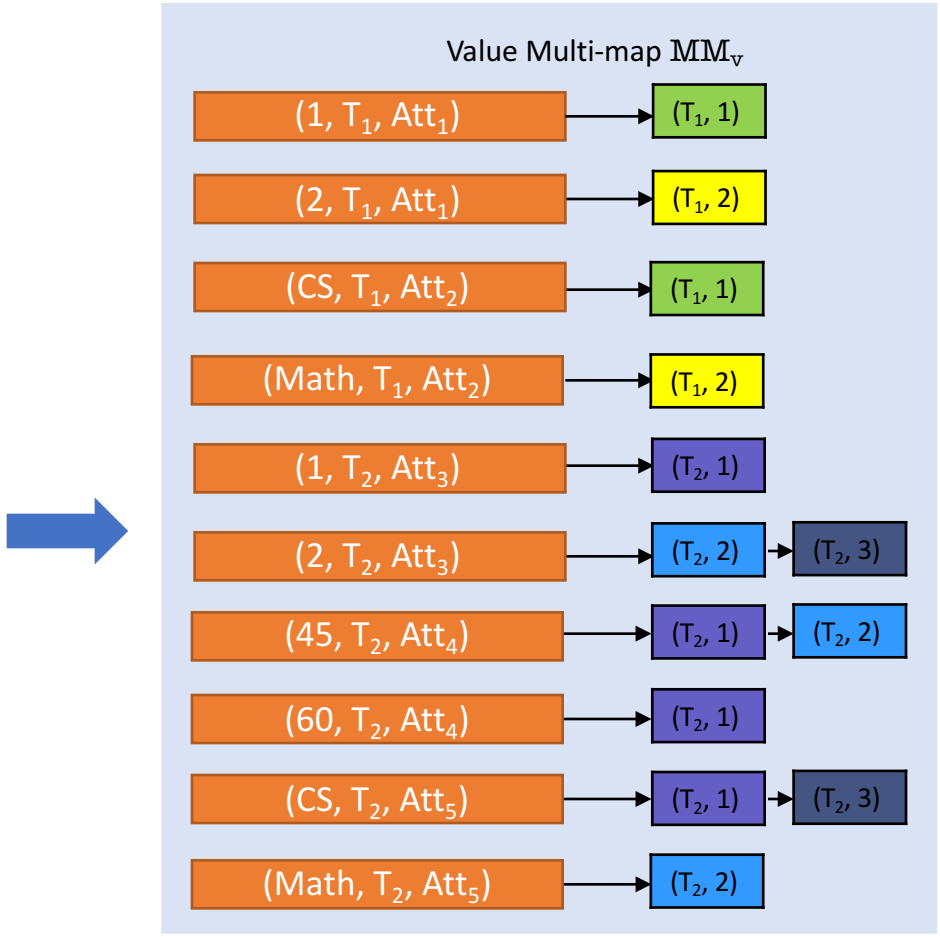



# Step 2: Value representation

| $Att_1$ | $Att_2$ |
|---|---|
| 1 | CS |
| 2 | Math |

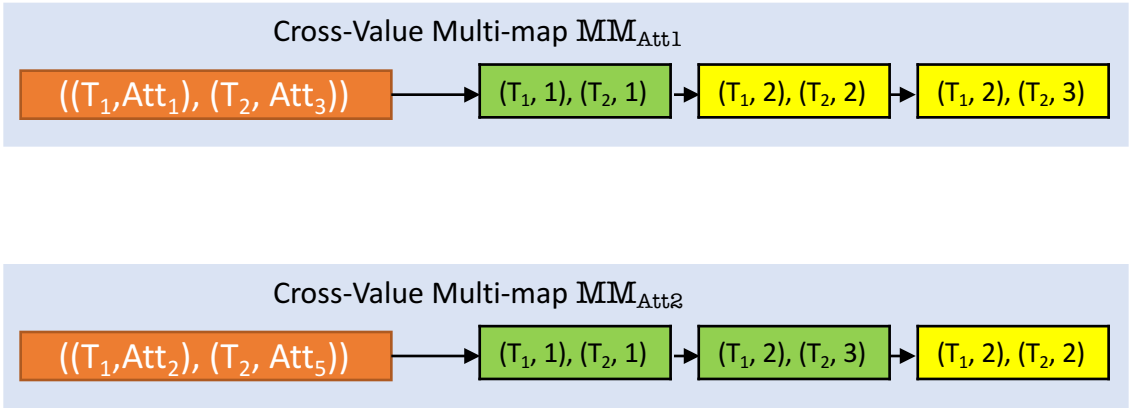
| $Att_3$ | $Att_4$ | $Att_5$ |
|---|---|---|
| 1 | 45 | CS |
| 2 | 45 | Math |
| 2 | 60 | CS |



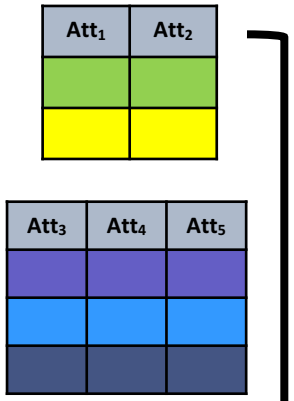
# Step 2: Cross-Value representation

| $Att_1$ | $Att_2$ |
|---|---|
| 1 | CS |
| 2 | Math |

| $Att_3$ | $Att_4$ | $Att_5$ |
|---|---|---|
| 1 | 45 | CS |
| 2 | 45 | Math |
| 2 | 60 | CS |

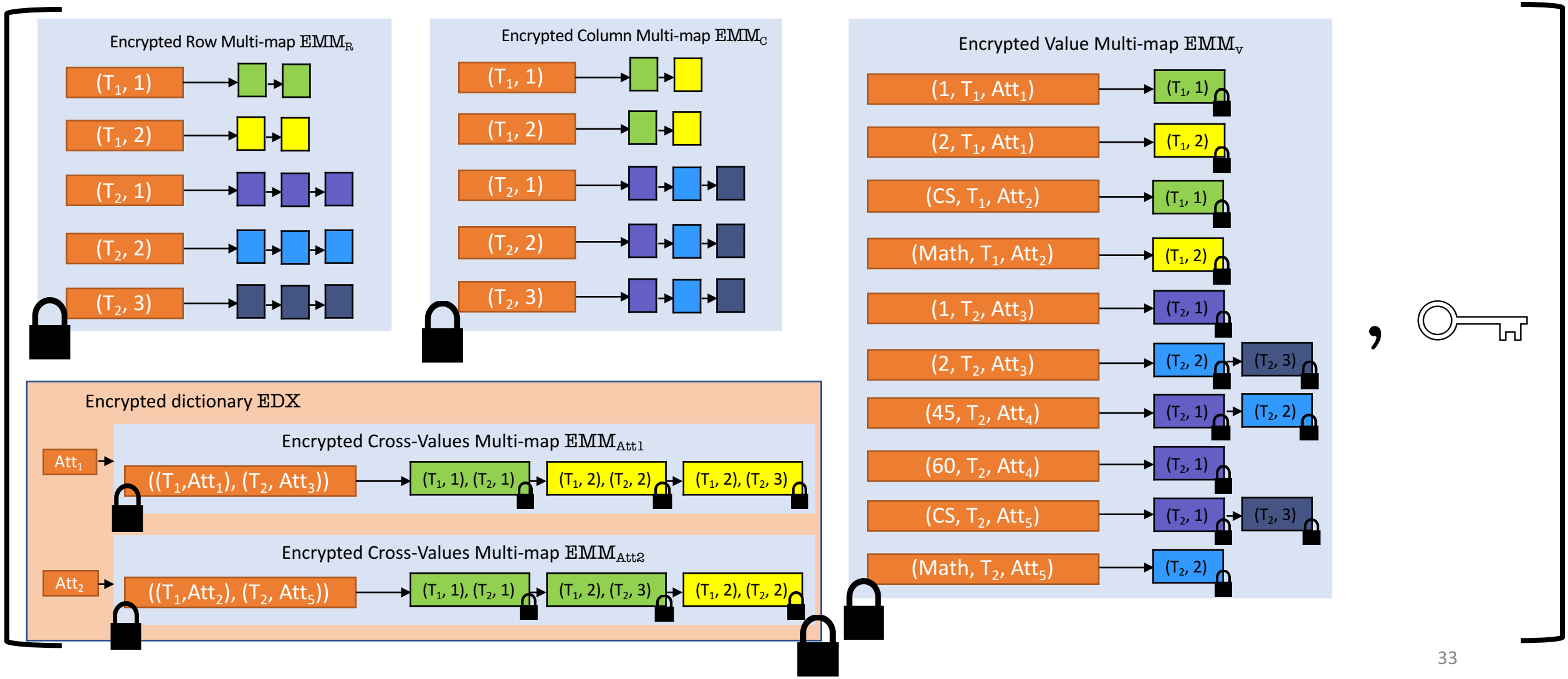


# Step 3: SPX Setup

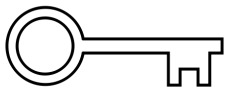
$\mathtt{Setup}^{\mathtt{SPX}}\left[1^k, \right.$ 

$\left.\right]$

# Step 3: SPX Setup



# Step 3: SPX Token (1)

$\text{Token}^{\text{SPX}}$ [  , **Select** $Att_3$ **From** $(T_1, T_2)$ **Where** $T_1.Att_2 = T_2.Att_5$ ]

# Step 3: SPX Token (2)

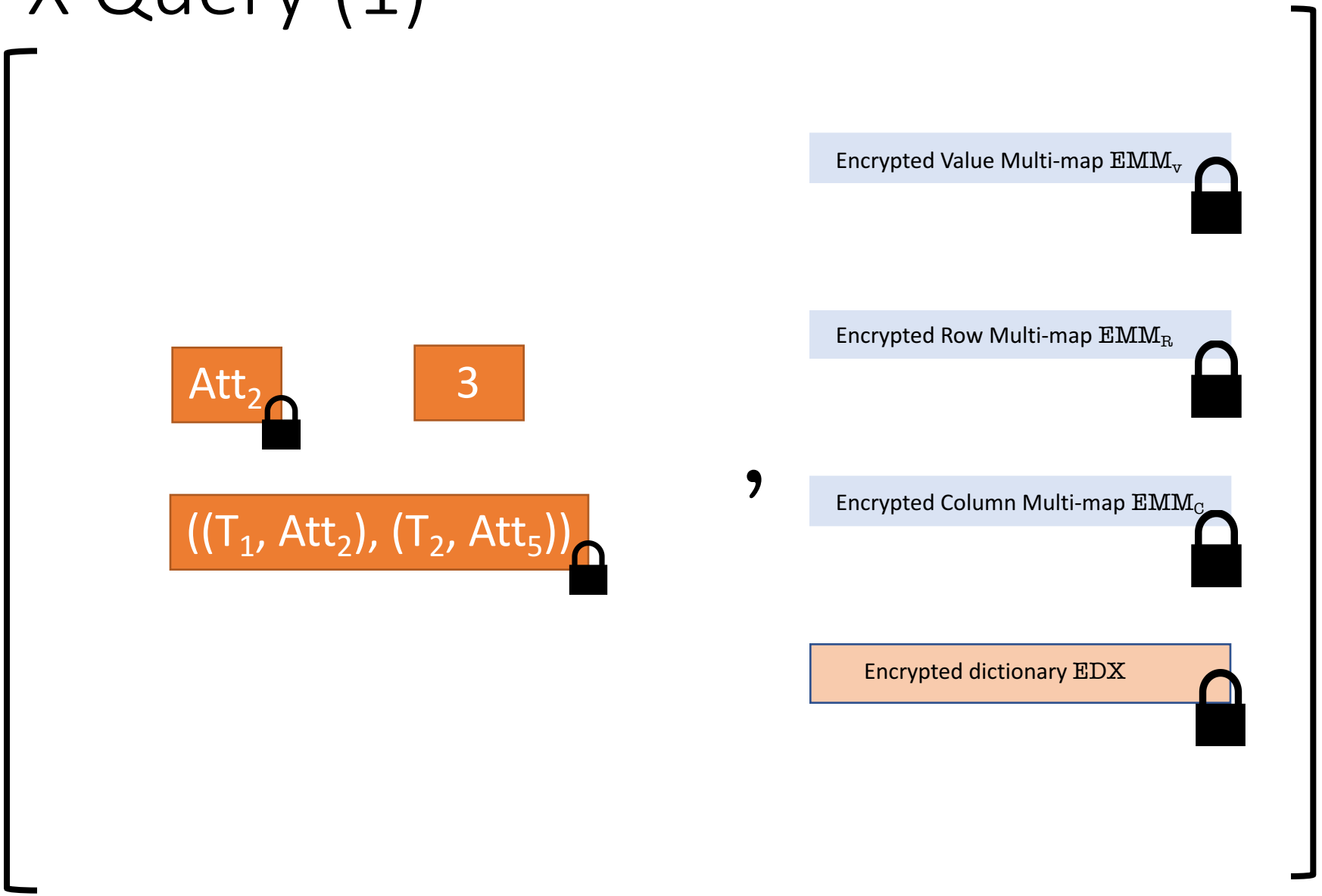
1. Rewrite SQL query to Normal Form

$$\pi_{\text{att}_3}\left(\sigma_{\text{att}_2=\text{att}_5}\left(\mathsf{T}_1 \times \mathsf{T}_2\right)\right)$$

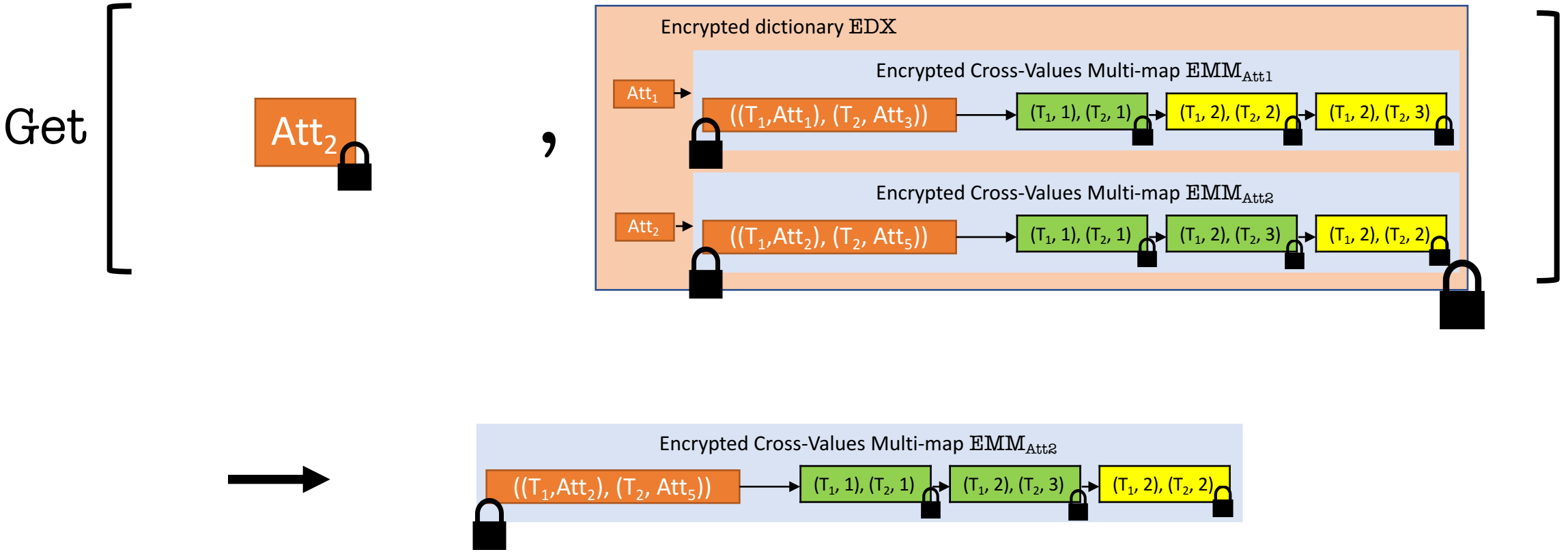
2. Rewrite Normal Form to Heuristic Normal Form
3. Generate the token

# Step 3: SPX Query (1)

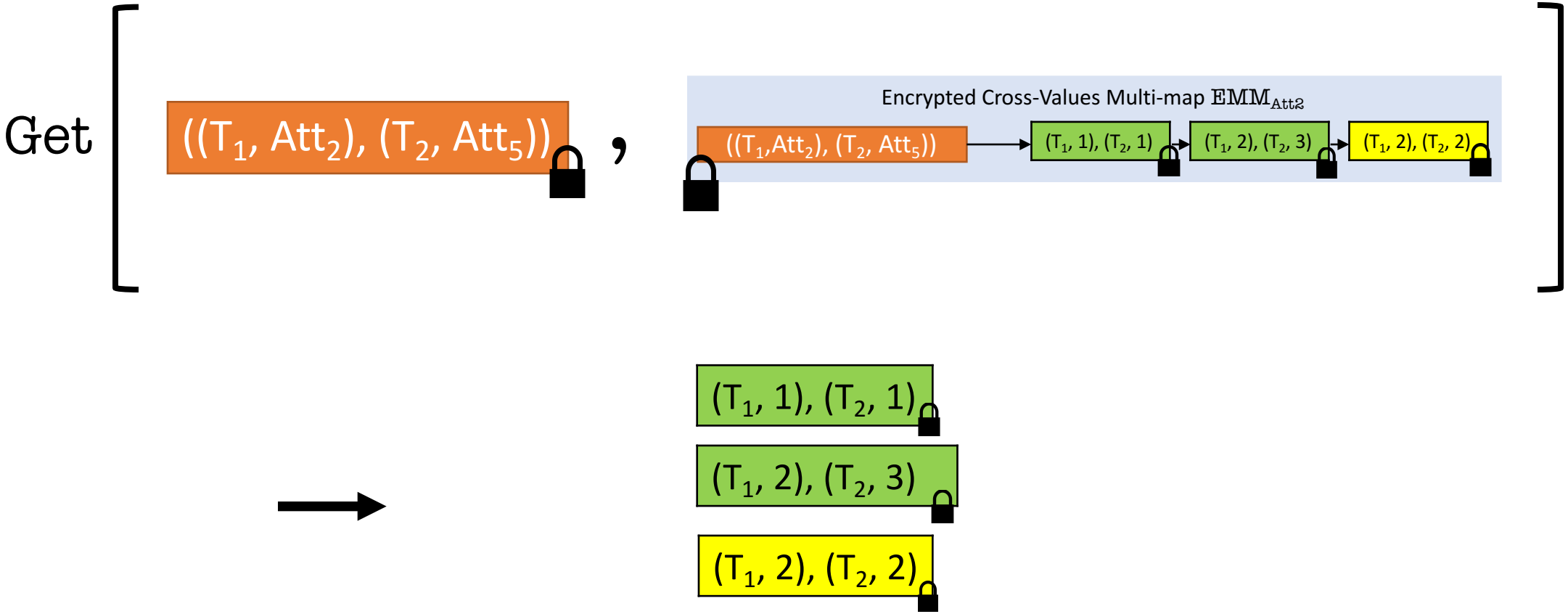
$\text{Query}^{\text{SPX}}$



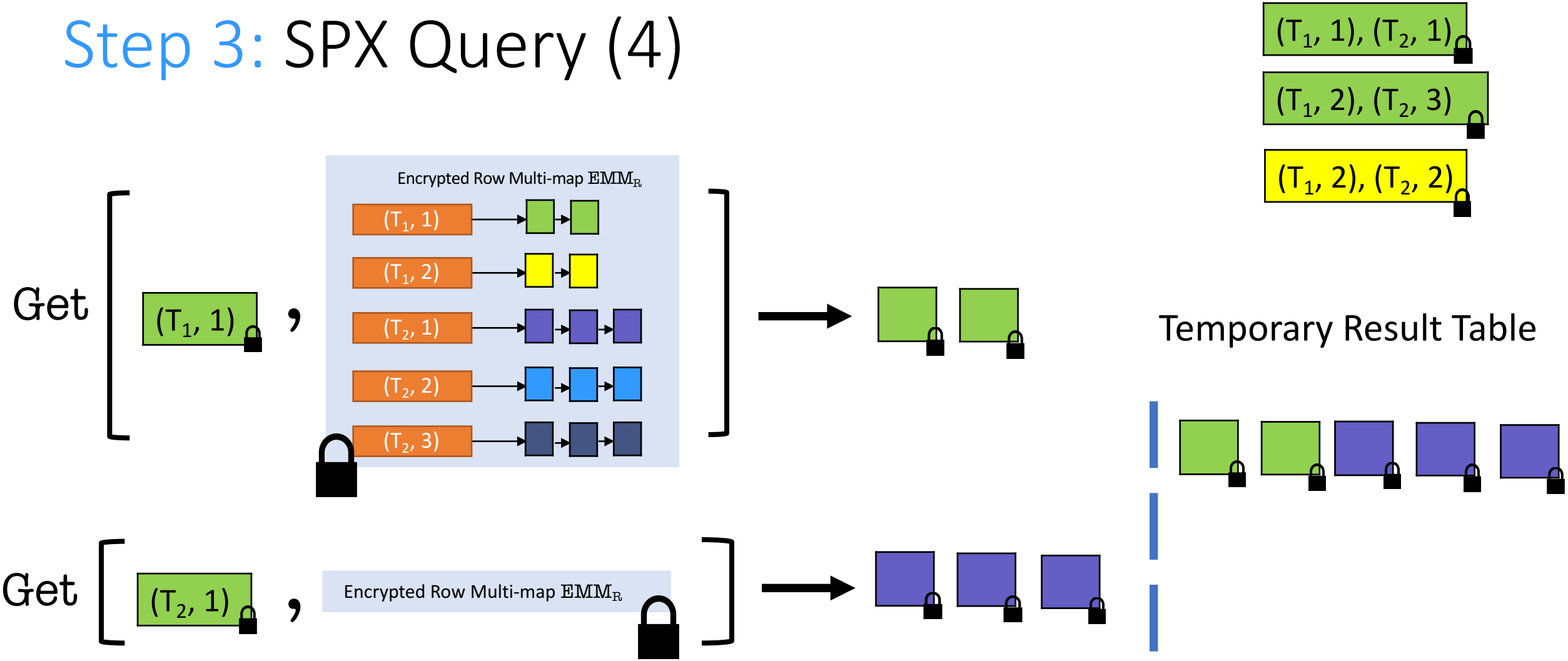
# Step 3: SPX Query (2)



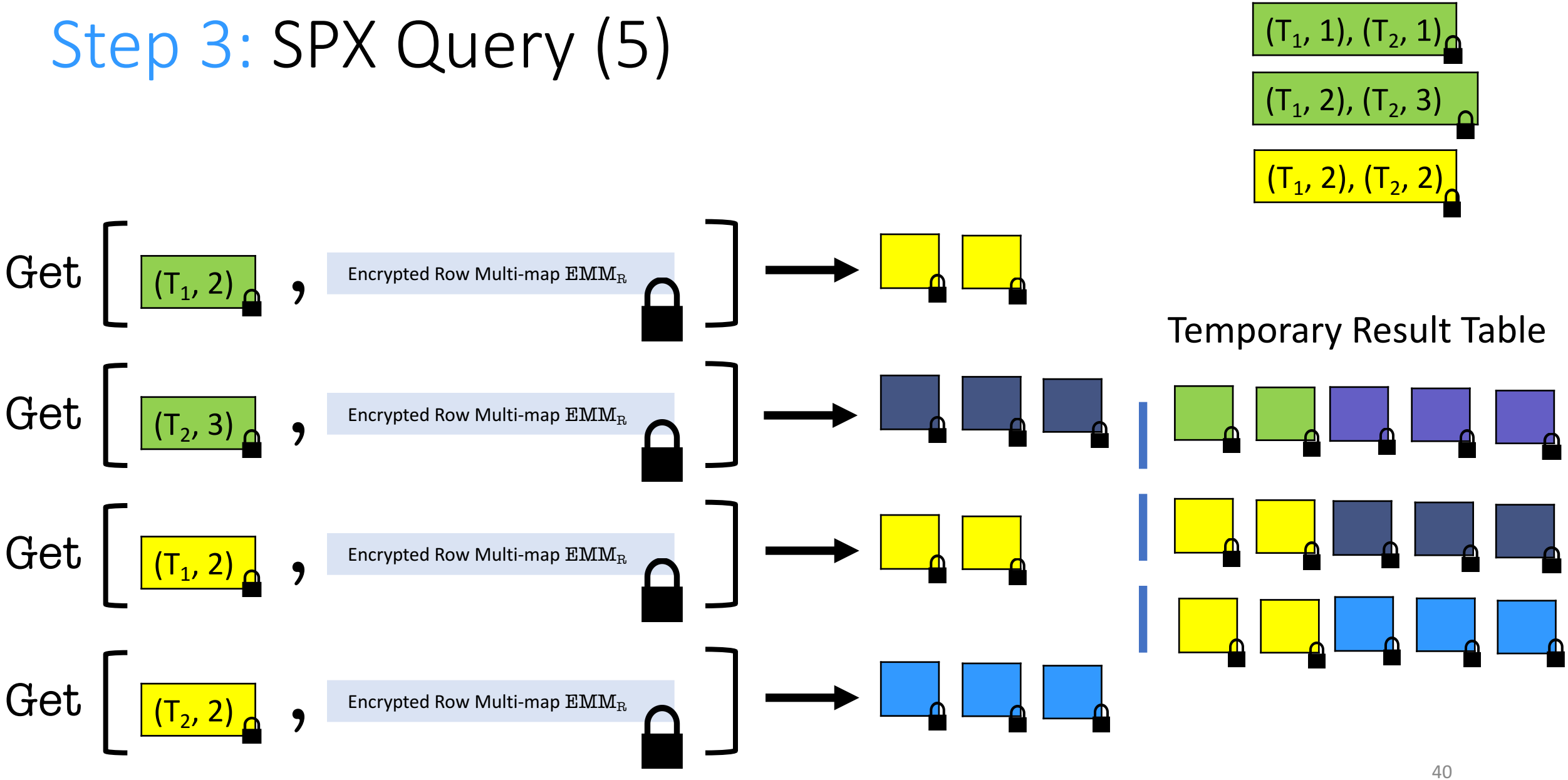
# Step 3: SPX Query (3)



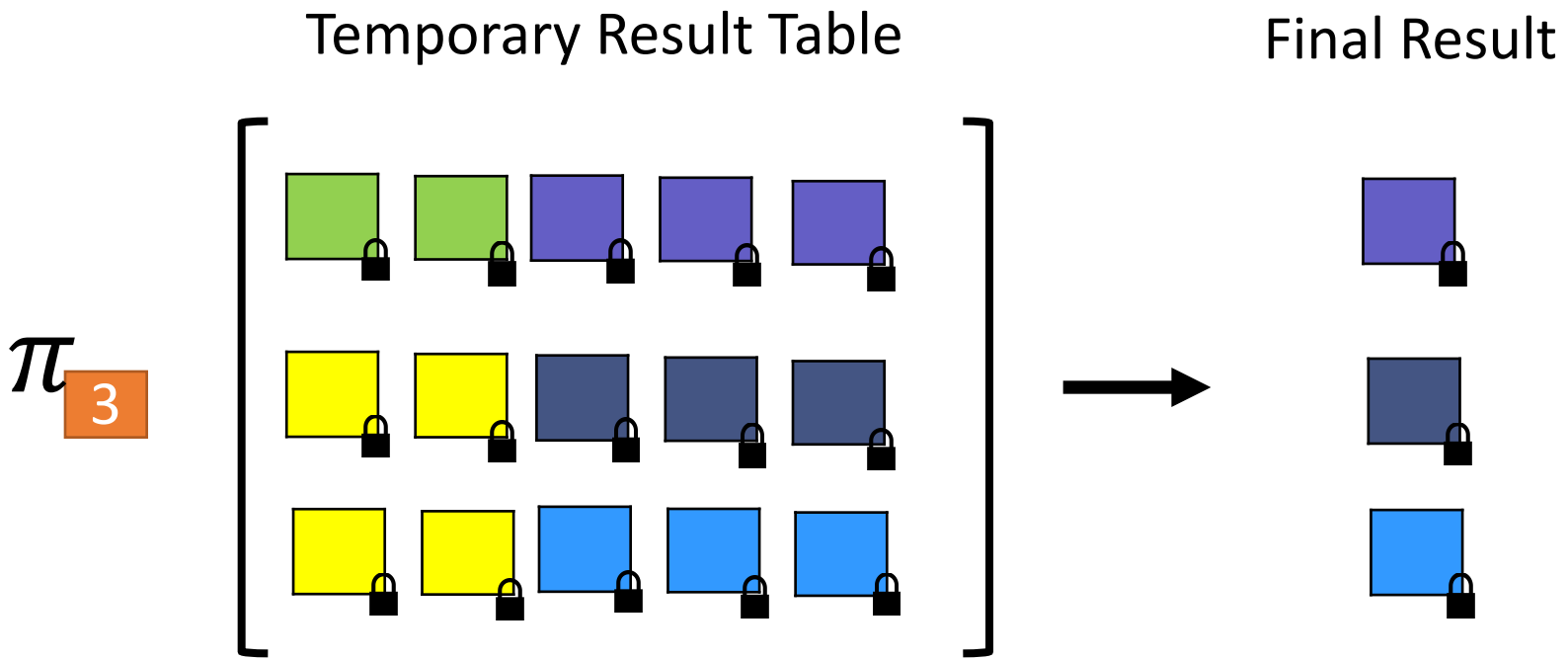
# Step 3: SPX Query (4)



# Step 3: SPX Query (5)

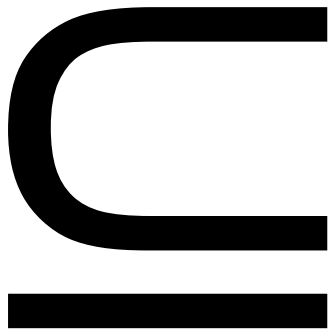


# Step 3: SPX Query (6)



# Leakage: SPX-OPT vs. PPE-based

- Query leakage of SPX
  - Cross product pattern
  - Projection pattern
  - Selection pattern



- Query leakage of PPE-based schemes
  - Cross product pattern
  - Projection pattern
  - Selection pattern
  - Frequency pattern
    - Persistent
    - Existing very strong attacks

# Modularity: SPX-Obliv vs. SPX-OPT

Query leakage of SPX-Obliv $\subseteq$ Query leakage of SPX-OPT

- When the EMMs are oblivious [GO96,SvDS+13,GMP16,KMO18]
- But comes with extra overhead

# SPX-OPT Asymptotics

- Worst-case query complexity with
  - Assuming optimal MM and DX encryption schemes [CK10,CJJ+14]
  - $h$ projected attributes
  - $t$ tables with size $s_i$ each
  - #R is the size of the result on a plaintext database

$$O\left(\frac{\#\mathsf{R}}{h} \cdot \sum_{i=1}^{t} s_i\right)$$

- Mild condition
  - If $h^{-1} \cdot \sum_{i=1}^{t} s_i$ is constant in #R , then query time is optimal

# SPX-OPT Asymptotics

- Communication complexity is optimal
- Storage depends on the data distribution

$$O(\#\mathsf{DB} + \sum_{\mathsf{att} \in \mathbb{S}} \#\mathsf{MM}_{\mathsf{att}})$$

- Concretely, the big-O hides a multiplicative factor of 3

# Takeaways and Future Work

- First STE-based encrypted relational database
- Sub-linear query time (optimal under certain conditions)
- First (forward-private) dynamic STE-based encrypted relational database
- Better leakage profile than PPE-based
- Future research questions:
  - Extend SPX to handle relational algebra
  - Remove interaction in $SPX^{+}$
  - Extend SPX to handle range sub-predicates
    - Design of new encrypted range solutions is required [KKNO16, LMP17]

# Thank you!

https://eprint.iacr.org/2016/453