Secure Data Structures

Sam A. Markelon

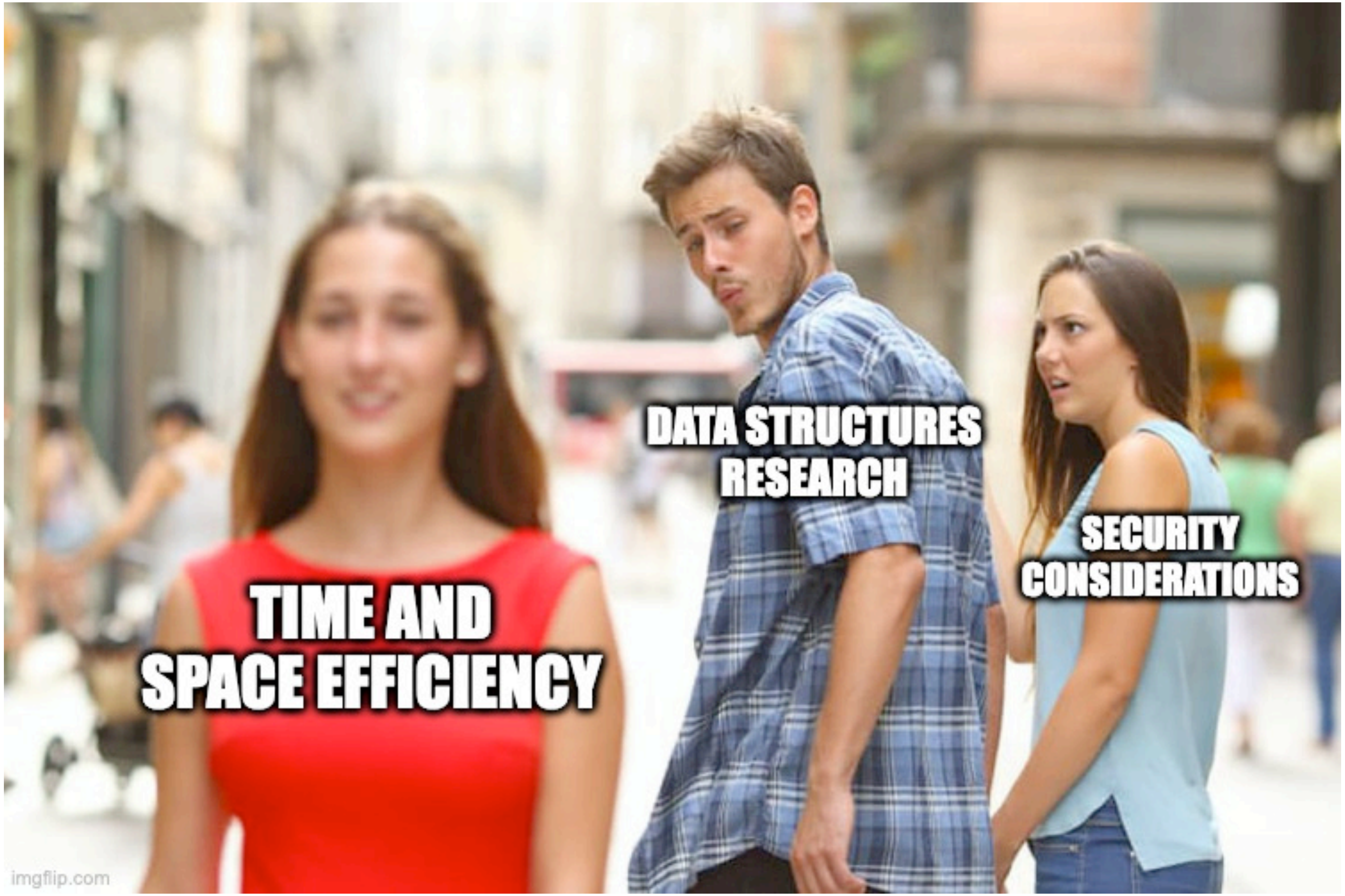
Thesis Proposal

December 4, 2024

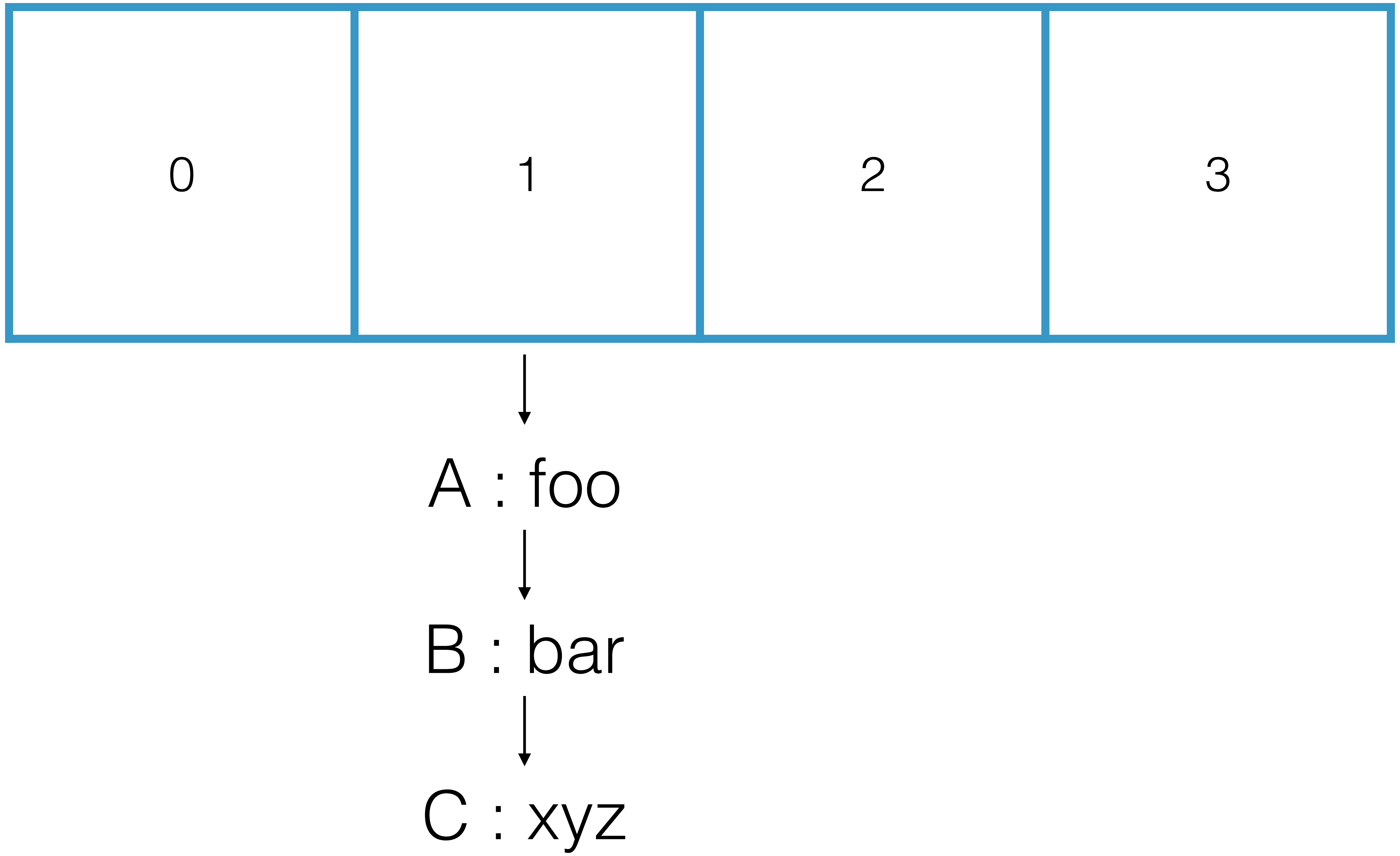
What are data structures?

Data structures define **representations** of possibly dynamic (multi)sets along with the **operations** that can be performed on the representation.

A Need for Speed (and Space)



Hash Flood DoS Attacks




```
hash(A) = 1
hash(B) = 1
hash(C) = 1
    .
    .
    .
```

Insertion of n elements ~ $O(n^2)$

Thesis Statement Summary

- Security of data structures is an afterthought
- Increasing use of data structures in adversarial environments
- Preliminary security results for data structures are overwhelmingly negative
- **Let's use the provable security framework**
    - Formal evaluation
    - Define attack models and goals
    - Design new provably secure structures
    - Analyze security and performance tradeoffs

Probabilistic Data Structures

Compactly represent (a stream of) data

and

provide **approximate answers** to queries about the data

- Frequency estimation
  How many times does x occur in the stream?
  Count-min sketch, Heavy-keeper

- Membership queries
  Is x in the set?
  Bloom filter, Cuckoo filter

- Cardinality estimation
  How many distinct elements in the set?
  HyperLogLog, KMV estimator

Probabilistic Data Structures

Compactly represent
(a stream of) data

and

provide **approximate answers** to
queries about the
data

- Bound on the **response error**
  - False positive rate for BF
  - Over-estimation bound for CMS
- Bound is strictly **non-adaptive**
  - Data does not depend on internal $$ of structure

Probabilistic Data Structures

Frequency estimation PDS

find the most sold stocks in the past minute

Frequency estimation PDS

identify possible DoS threats (network-monitoring

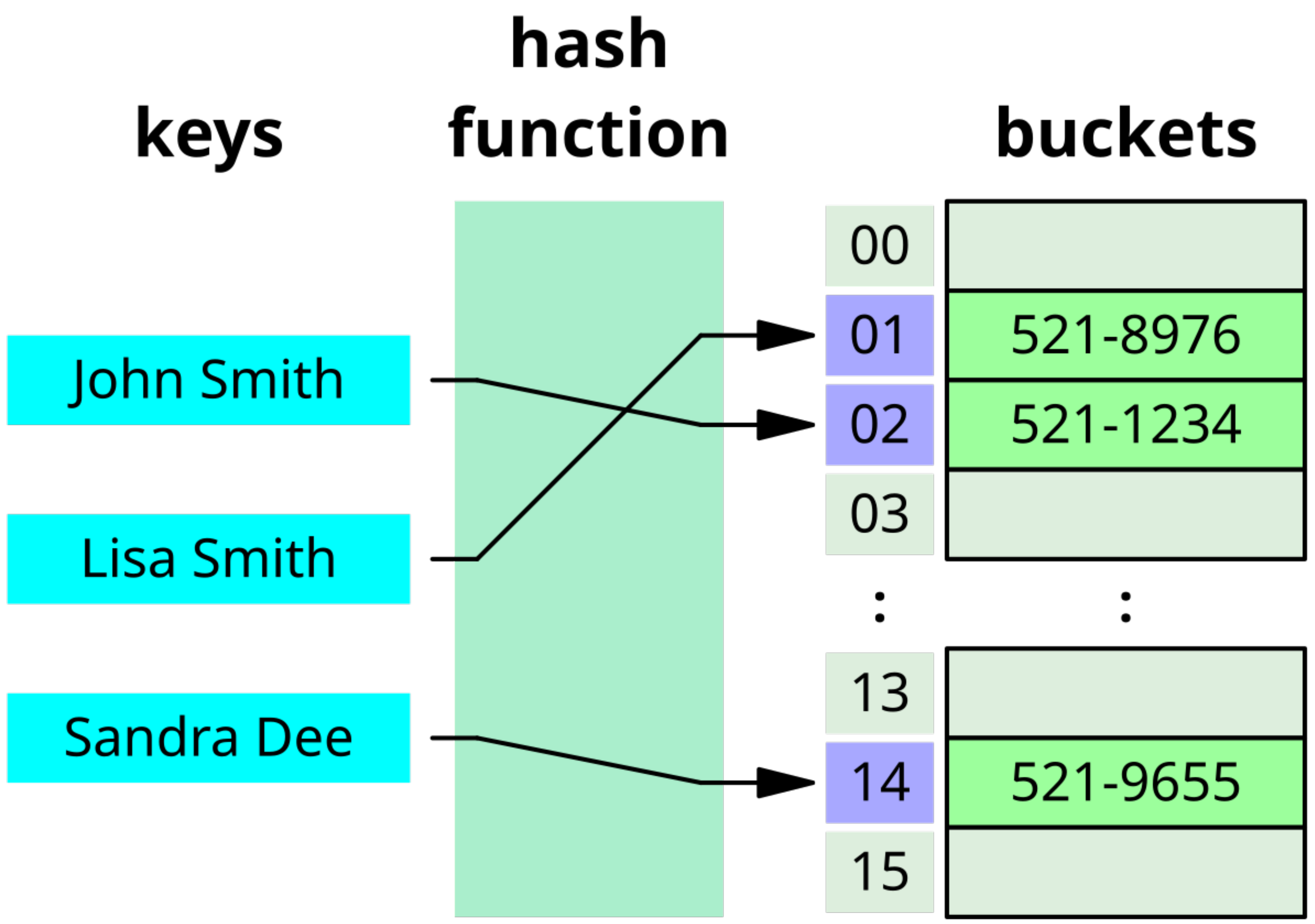
Membership query PDS

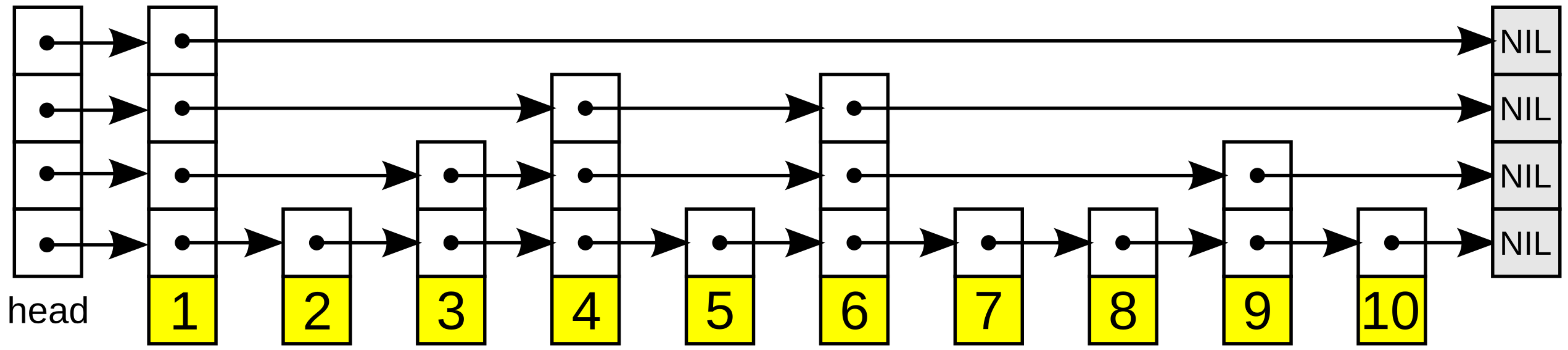
check certificate revocations in TLS/SSL

Cardinality estimation PDS

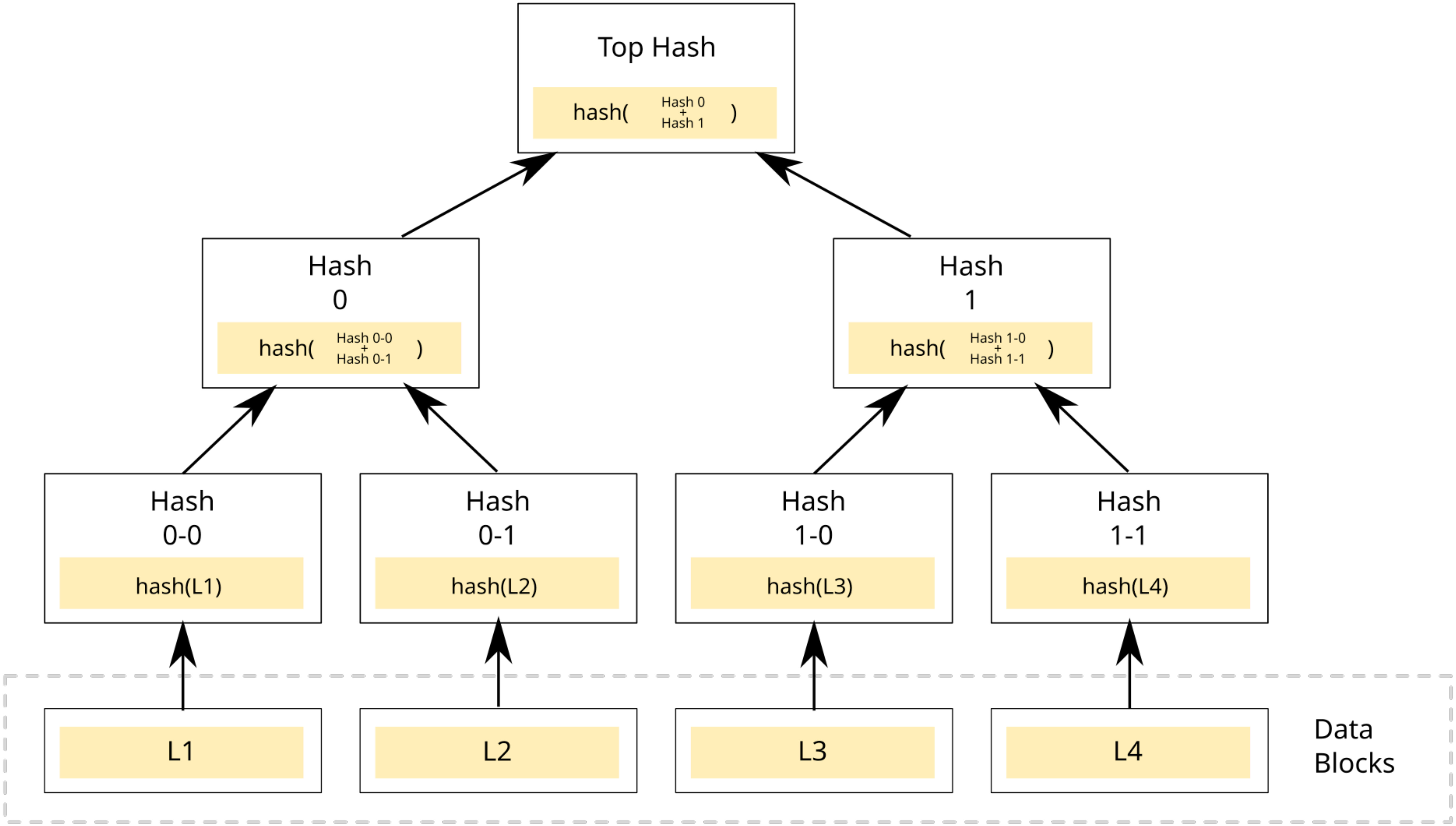
count the number of distinct users on a given service

Skipping Data Structures





Verifiable Data Structures



Thesis Work

Compact Frequency Estimators in Adversarial Environments

CCS ‘23

Probabilistic Data Structures in the Wild: A Security Analysis of Redis

Submitted: CODASPY ‘25

Skipping Data Structures in Adversarial Environments

In progress: CCS ‘25

A Formal Treatment of Key Transparency Systems with Scalability Improvements

Submitted: S&P ‘25

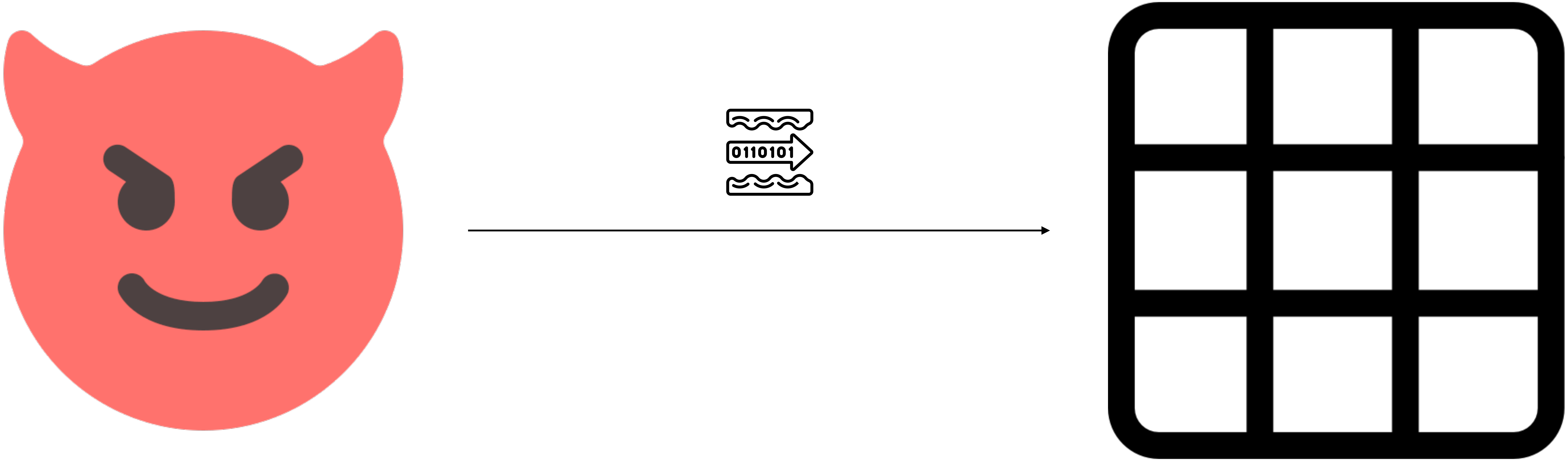
Compact, Private, and Verifiable Data Structures

In progress: TBD ’25

Compact Frequency Estimators in Adversarial Environments

Sam A. Markelon, Mia Filić, and Thomas Shrimpton
(CCS '23)

Adversarial Correctness of CFE



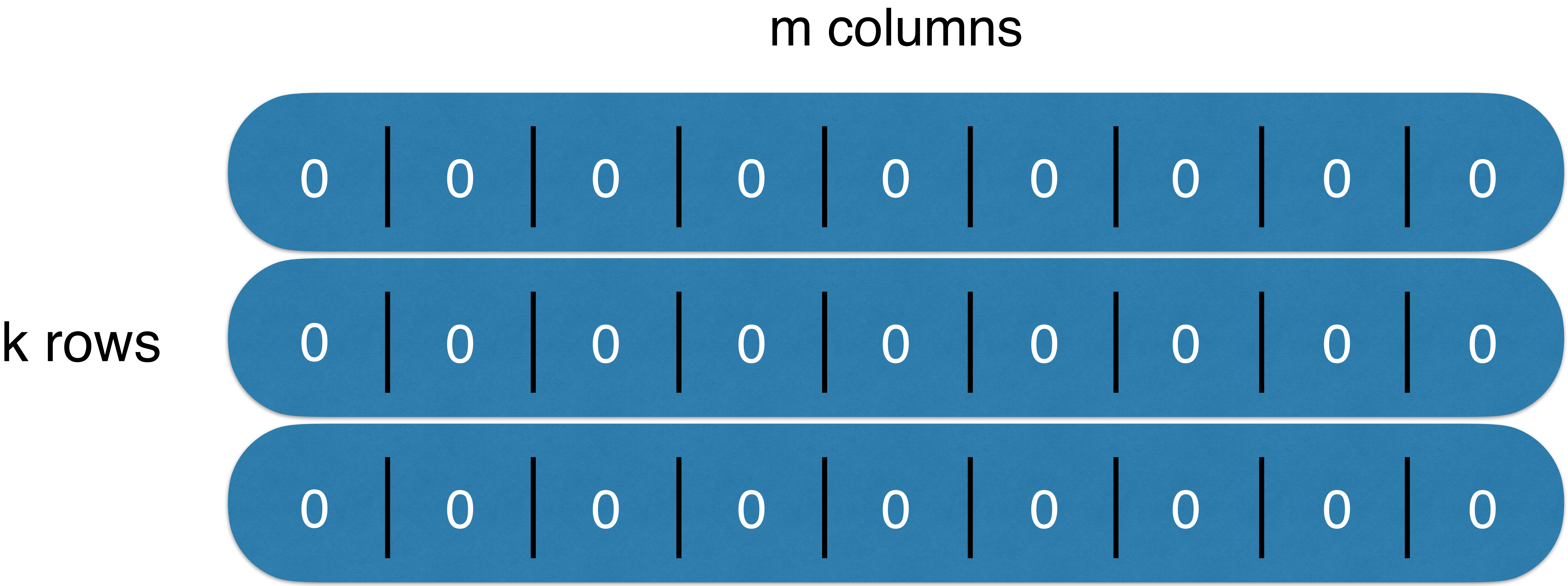
Adversarial Correctness of PDS

[AuthorsYear]	Structures	Security Proof Style
[NY15]	Bloom filter	Game based
[CPS19]	Bloom Filter Counting Filter Count-min Sketch	Game based
[PR22]	HyperLogLog	Simulation
[FPUV22]	Bloom Filter Cuckoo Filter	Simulation (privacy notions!)
[MFS23]	Count-min Sketch HeavyKeeper	Game based*

Standard hash functions:
Large correctness errors

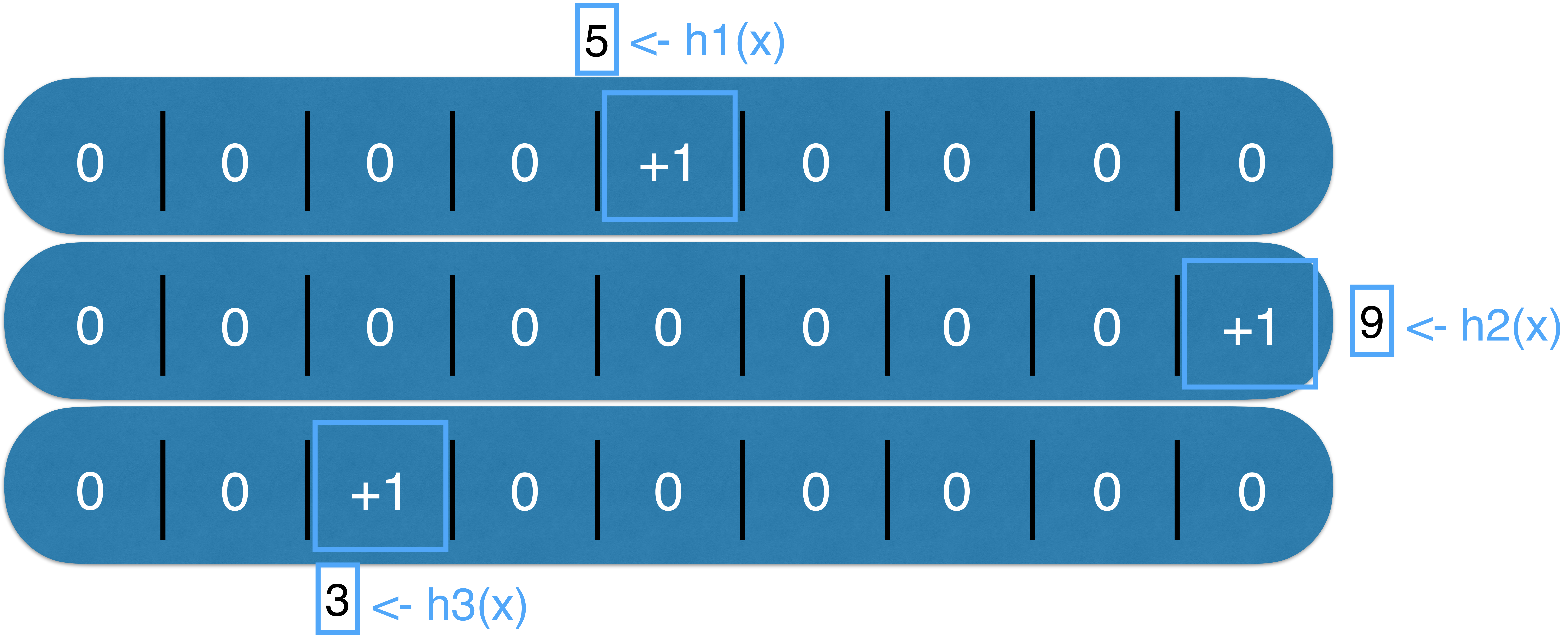
Swap to a keyed primitive:
Adversarial robust structures*

Count-min Sketch (CMS)

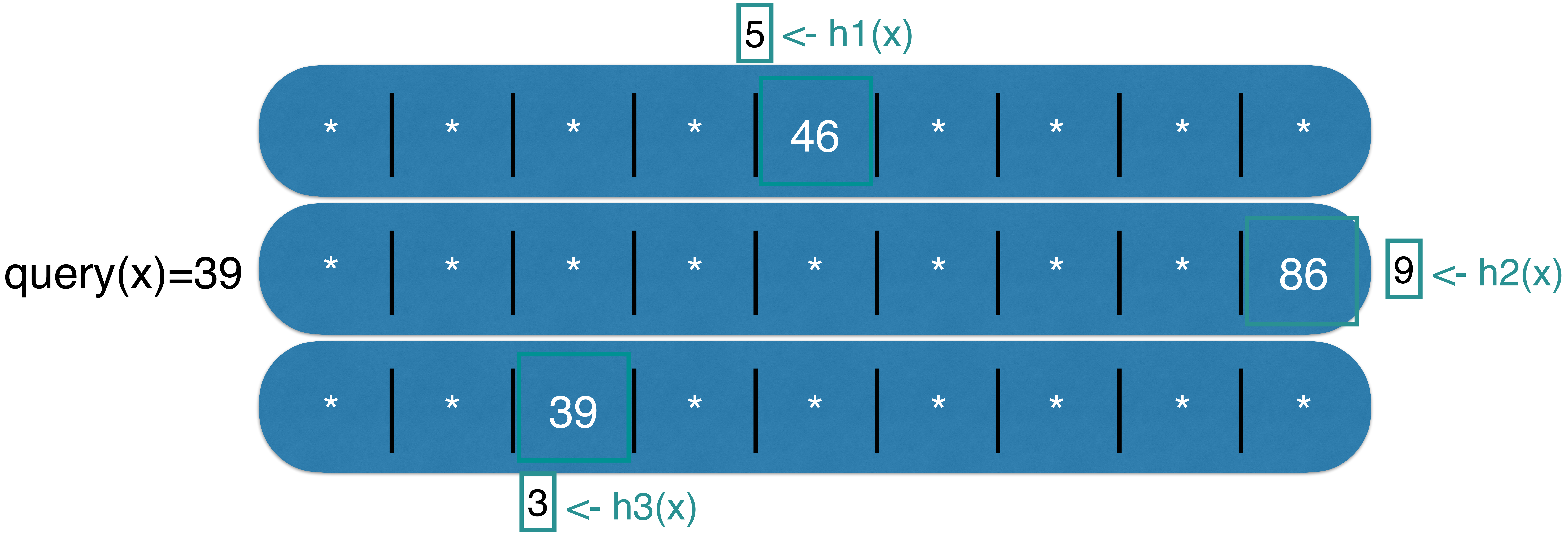


CMS Insert

insert(x)



CMS Query



CMS Properties

- Only overestimates
- "Honest Setting" guarantee
- Adversarial setting?

CFE Error Model (simplified)

x

CMS[🔑]:
insert
query

reveal

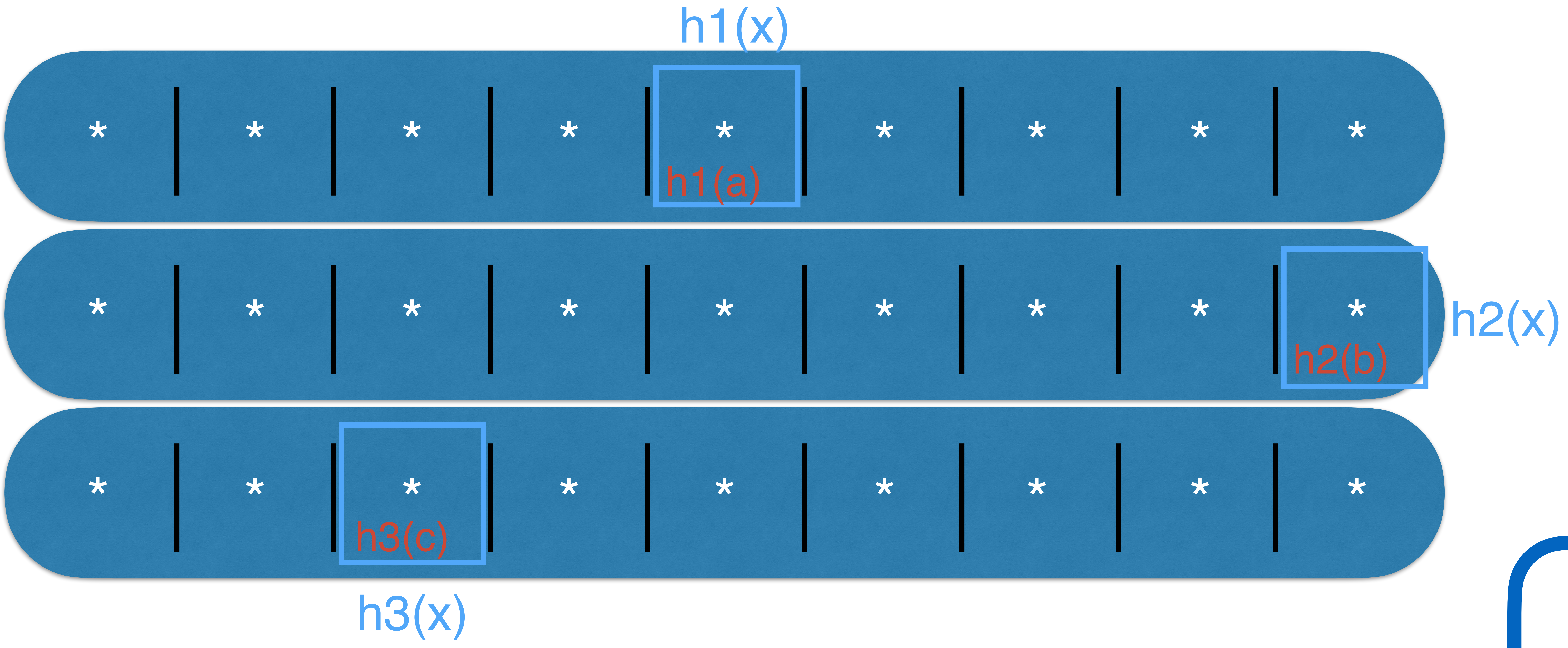
hash

Maximise
CMS error

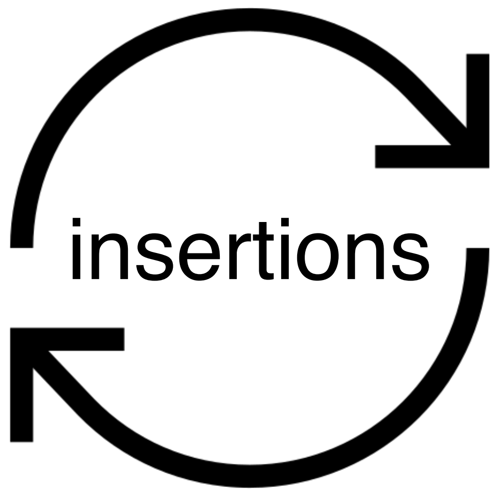
query(x) >> true_frequency(x)

CMS "Public Hash" Attack

x



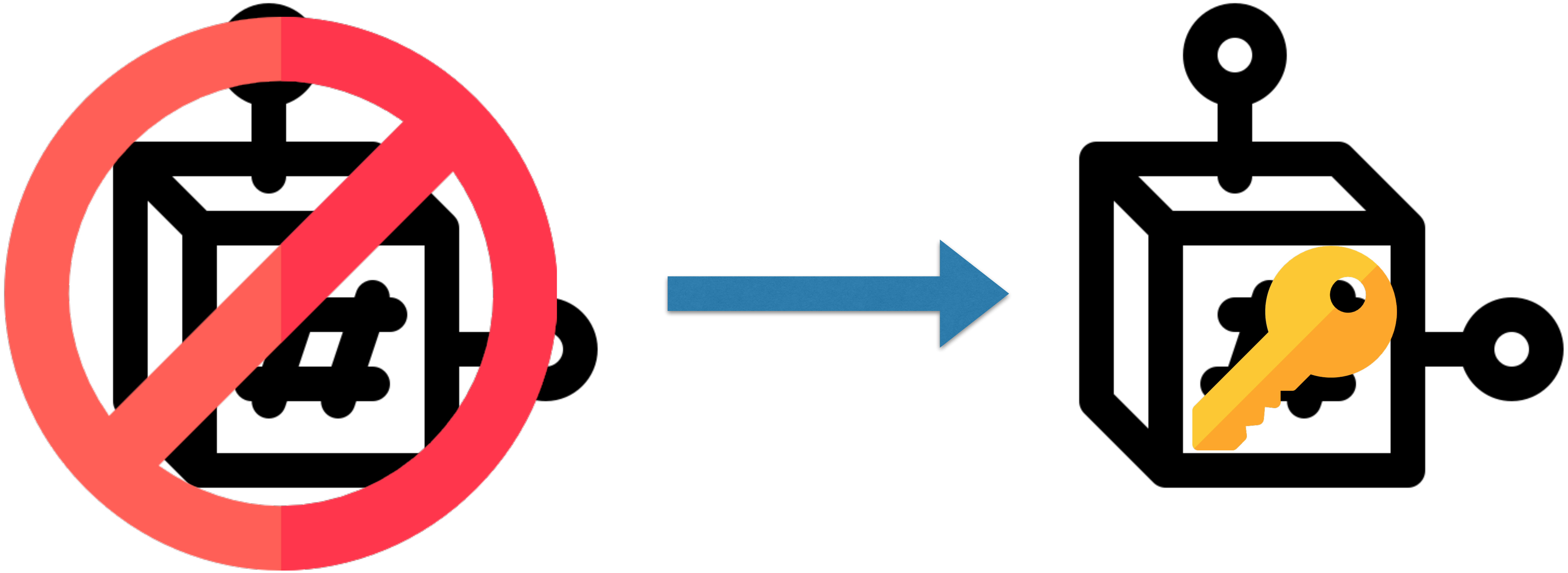
Cover set



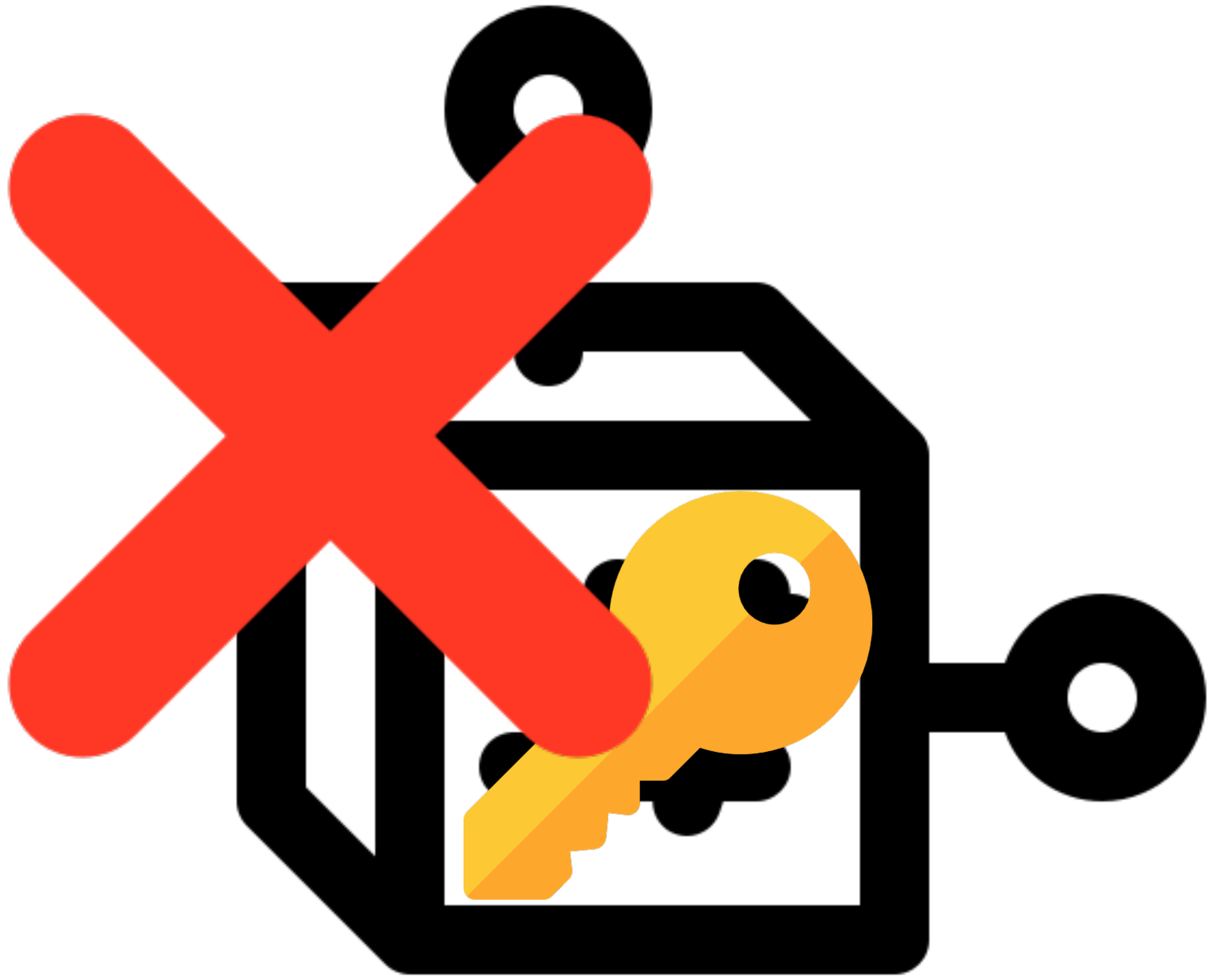
Err = insertions/k

Cover set = {a, b, c}

CMS Attacks Mitigations



CMS Attacks Mitigations



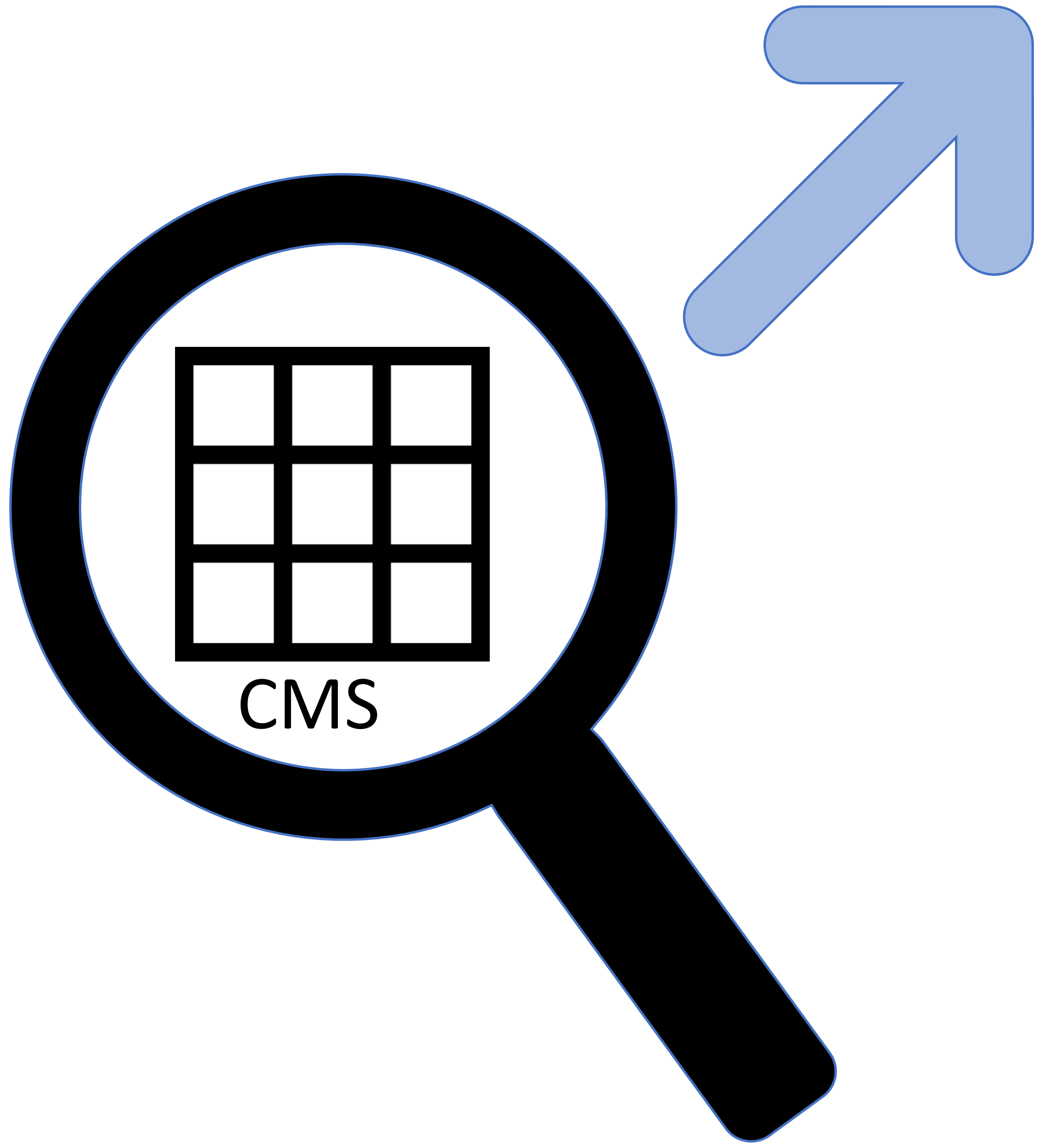
Still attacks when using a PRF and blackboxed structure!

Existing CFEs are not adversarially robust!

Motivating a more robust CFE

$$\text{cnt} = n_x + \sum_{y \in V_x^i} n_y$$

CMS minimizes the "collision noise"

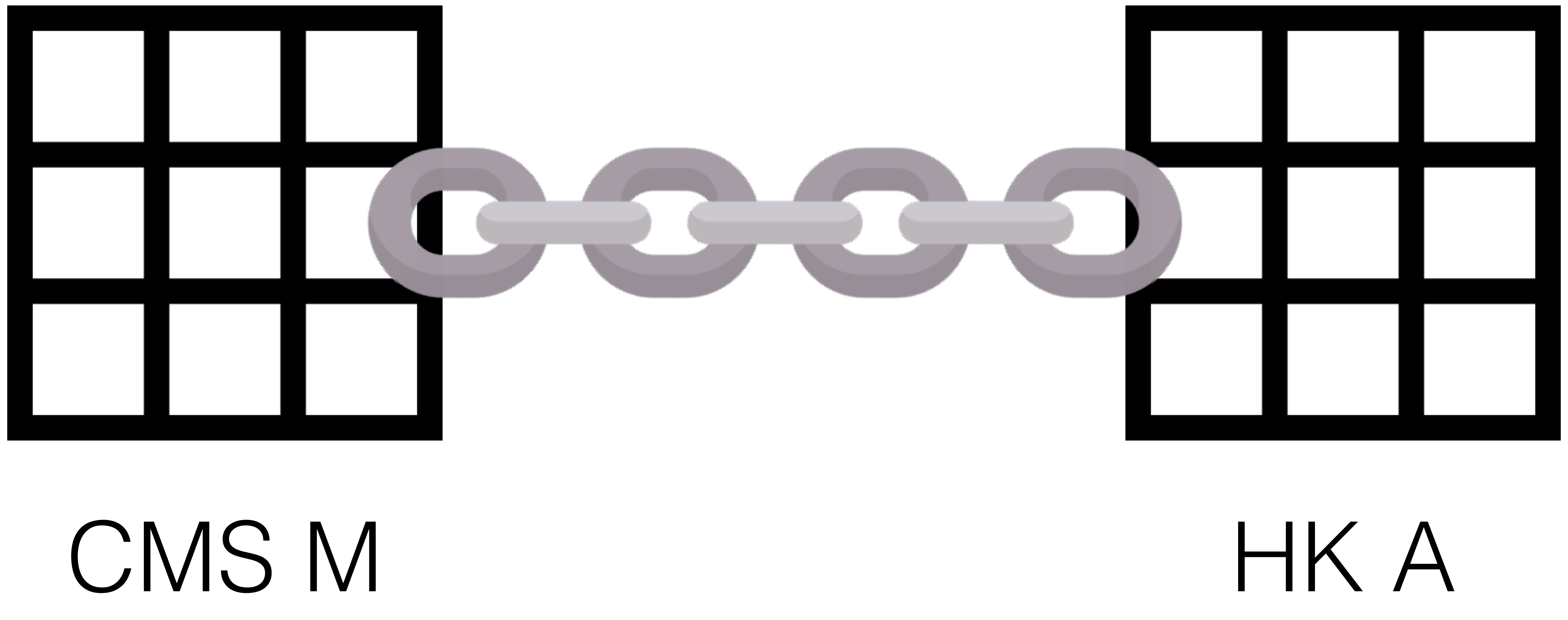


Can we do better? Yes!

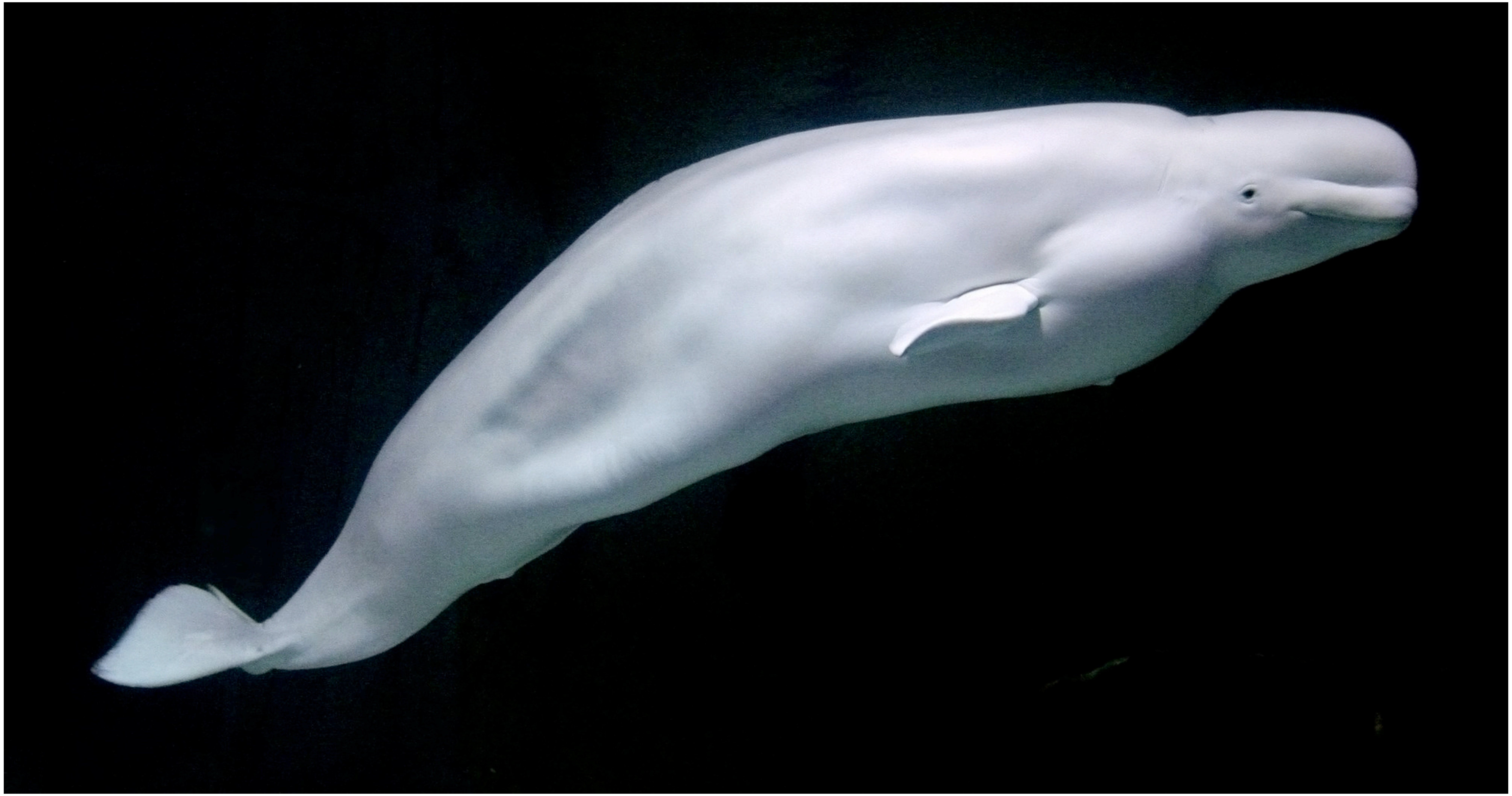
Idea: Use information from an auxiliary sketch!

Count-Keeper

- Hybrid between a CMS and HK
- Detects (does prevent) attacks
  - Flagging mechanism
  - Attacks are less damaging
- Works well in practice
  - Honest setting performance



Adversarially Robust CFE?



By Javier Yaya Tur (CAC, S. A.), CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=23971602

Probabilistic Data Structures in the Wild: A Security Analysis of Redis

Mia Filić, Jonas Hofmann, **Sam A. Markelon**, Kenneth G. Paterson, Anupama Unnikrishnan*

(Submitted to CODASPY '25)

* Alphabetical Ordering Used

Redis and RedisBloom

RedisBloom Details

- Open Source
- Widely Used
- Six PDS (We examine four)
  - Bloom Filters, Cuckoo filters, Count-min Sketch, Top-K (HeavyKeeper)
  - HyperLogLog, T-Digest
- Use MurmurHash2 with fixed seeds

Redis Security Model

"…it's totally insecure to let **untrusted clients access the system**, please protect it from the outside world yourself"

"an **attacker might insert data** into Redis that **triggers pathological (worst case) algorithm complexity on data structures** implemented inside Redis internals"

Our Attacks

- Ten different attacks against the four PDS we consider
  - **One against CMS and three against HK**
- MurmurHash2 family has fast inversion algorithms!
  - Target hash h and seed s, can generate arbitrarily many x s.t. h = hash(s,x).
  - Due to ASCII formatting constraints need to try ~16 inversions to find a collision
  - **Upshot for CFE: Find cover sets very fast!**

CMS Overestimation Attack

ϵ, δ (m, k)	Ours	[24]
$2.7 \times 10^{-3}, 1.8 \times 10^{-2}$ $(1024, 4)$	66.85	8533.32
$6.6 \times 10^{-4}, 1.8 \times 10^{-2}$ $(4096, 4)$	61.11	34133.36
$2.7 \times 10^{-3}, 3.4 \times 10^{-4}$ $(1024, 8)$	124.22	22264.72
$6.6 \times 10^{-4}, 3.4 \times 10^{-4}$ $(4096, 8)$	128.8	89058.72

Table 1: Experimental number (average over 100 trials) of equivalent *MurmurHash2* calls needed to find a cover for a random target x. We compare the average to the expected number of *MurmurHash2* calls needed in the attack of [24], namely kmH_k.

Implement attack from CCS '23 paper far more efficiently!

HK Attacks

- Very efficiently cause frequent elements to “disappear” (CCS ’23)
- Overestimation attacks due to being able to efficiently find fingerprint collisions
- DoS the entire structure
  - Pre-compute elements that map to every counter in the structure
  - Insert them ~100 times each in succession
  - Any subsequent insertions are never recorded

Countermeasures for RedisBloom

- PRF switch for Bloom filter and Cuckoo Filter
- Recall — no provably secure CFE
  - Suggestion: use Count-Keeper with a PRF

Skipping Data Structures in Adversarial Environments

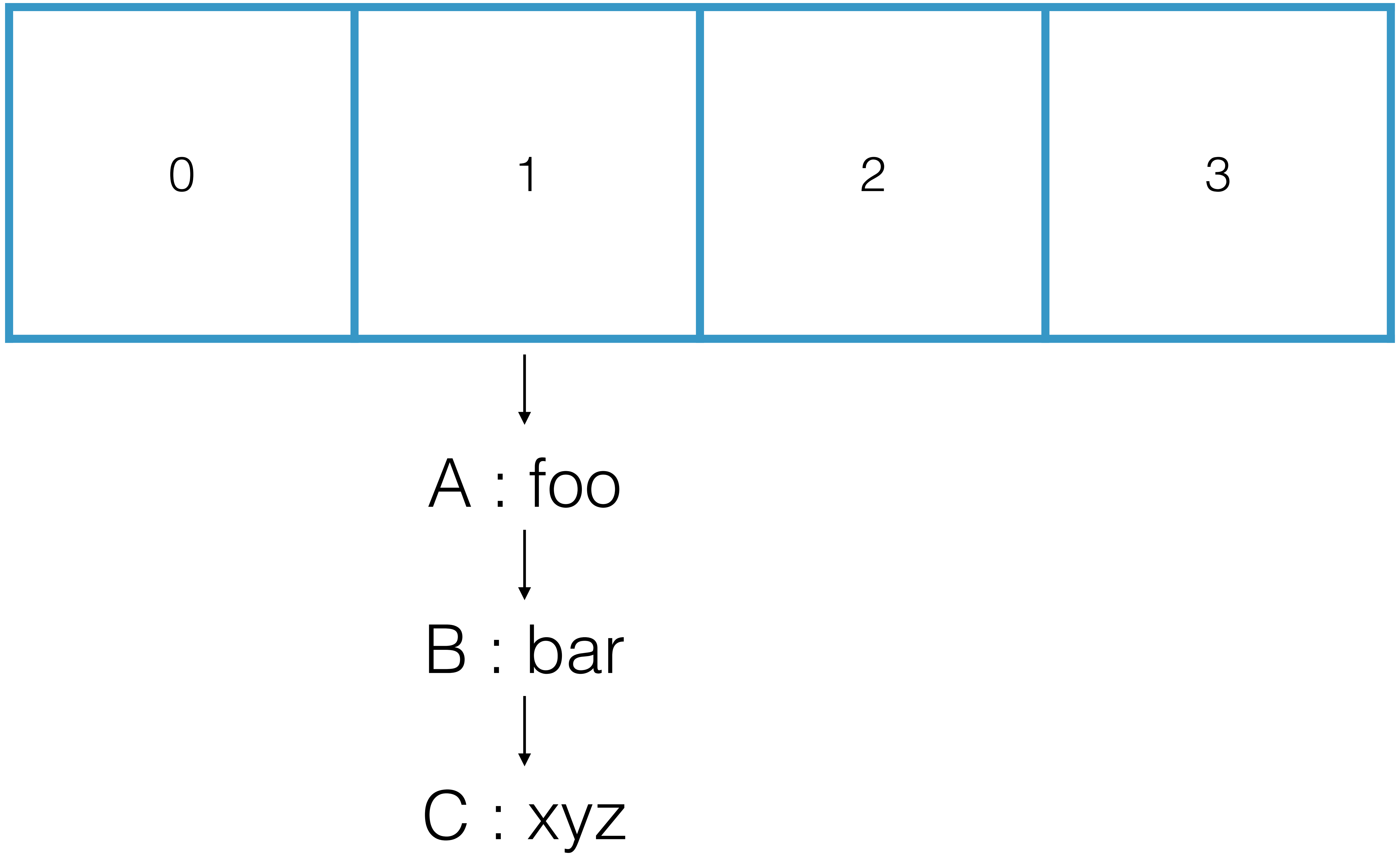
Moritz Huppert, **Sam A. Markelon**, Marc Fischlin*

(In progress. Target: CCS '25)

* Alphabetical Ordering Used

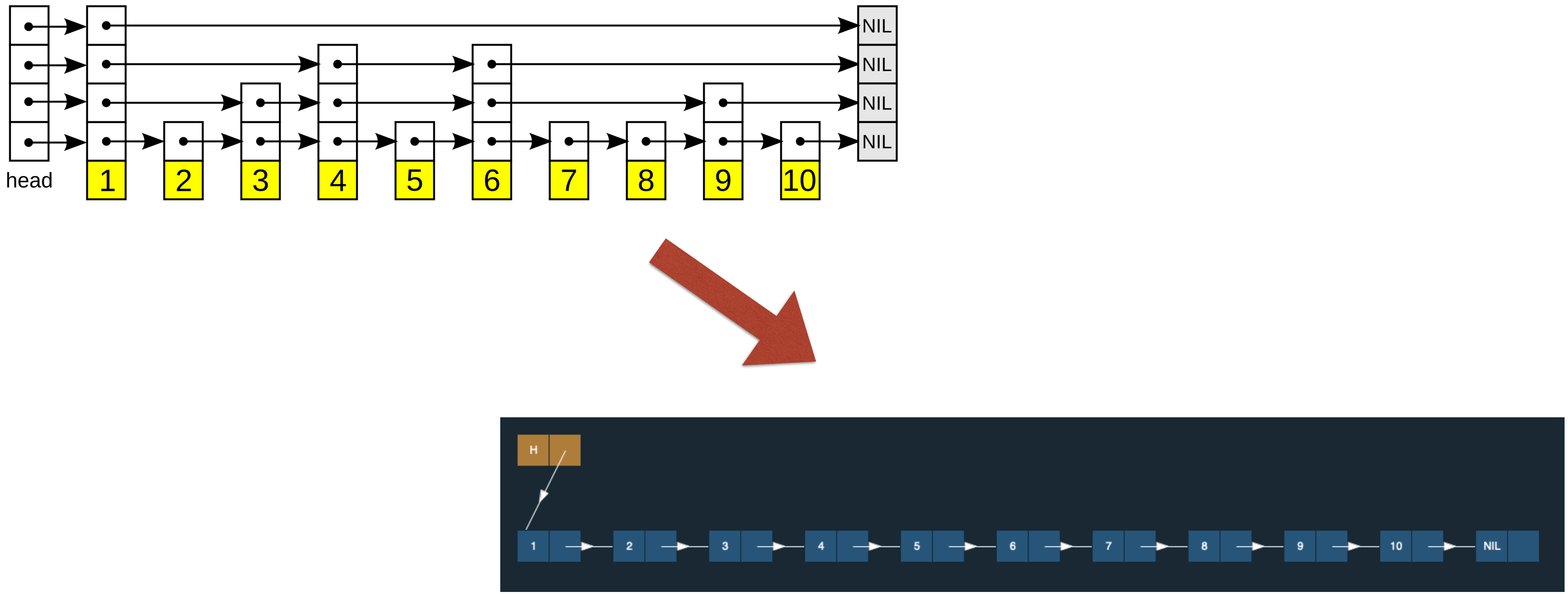
Recall: Hash Flood DoS Attacks

```
hash(A) = 1
hash(B) = 1
hash(C) = 1
      .
      .
      .
```



Insertion of n elements ~ $O(n^2)$

Similar Attacks Against Skip Lists



Motivating a Security Model

- A plethora of attack papers against hash tables and skip lists
  - No real attempt to formalize a security model
  - Some countermeasures explored
  - Some of these exploit timing side channels
- Consider the strongest adversary
  - Can perform any sequence of operations (wrt to some budget)
  - Has access to the internals of the structure at all times

Conserve Target Properties of the DS

- Want to conserve fast search operation
  - Entirely determined by the representation
- Known "non-adaptive" bounds
  - Maximum bucket population for HT
  - Maximum search path length for skip list
- Adversary wins in our game if the measured property after their execution **exceeds** the non-adaptive bound by more than some limit

HT Maximum Bucket Population: $\phi(D, \mathsf{repr})$

$e \leftarrow 0$
for $i \leftarrow 1$ **to** m
  $\ell \leftarrow \mathsf{length}(T[i])$
  if $\ell > e$
    $e \leftarrow \ell$
return e

Figure 2: The HT Maximum Bucket Population function $\phi : \mathcal{D} \times \{0,1\}^* \rightarrow \mathbb{R}$.

Towards Robust Structures

- No deletions
  - Replicate functionality by marking elements deleted
- No choosing how or where elements are inserted

Initial Results

- **Robust Hash Table**
  - Swap hash functions for a PRF and do not allow deletions
- **Robust Skip List**
  - Cannot use a PRF — destroys order!
  - Deterministic swapping mechanism that "heals" the structure and ND
- **Robust Treap**
  - Inherently robust with no deletions

Outstanding Work

- Generalize these results?
  - Skipping data structures as a sequence of iid random variables
- Formal proofs
  - Skip list and treap
- Analyze operational effects of our mitigations

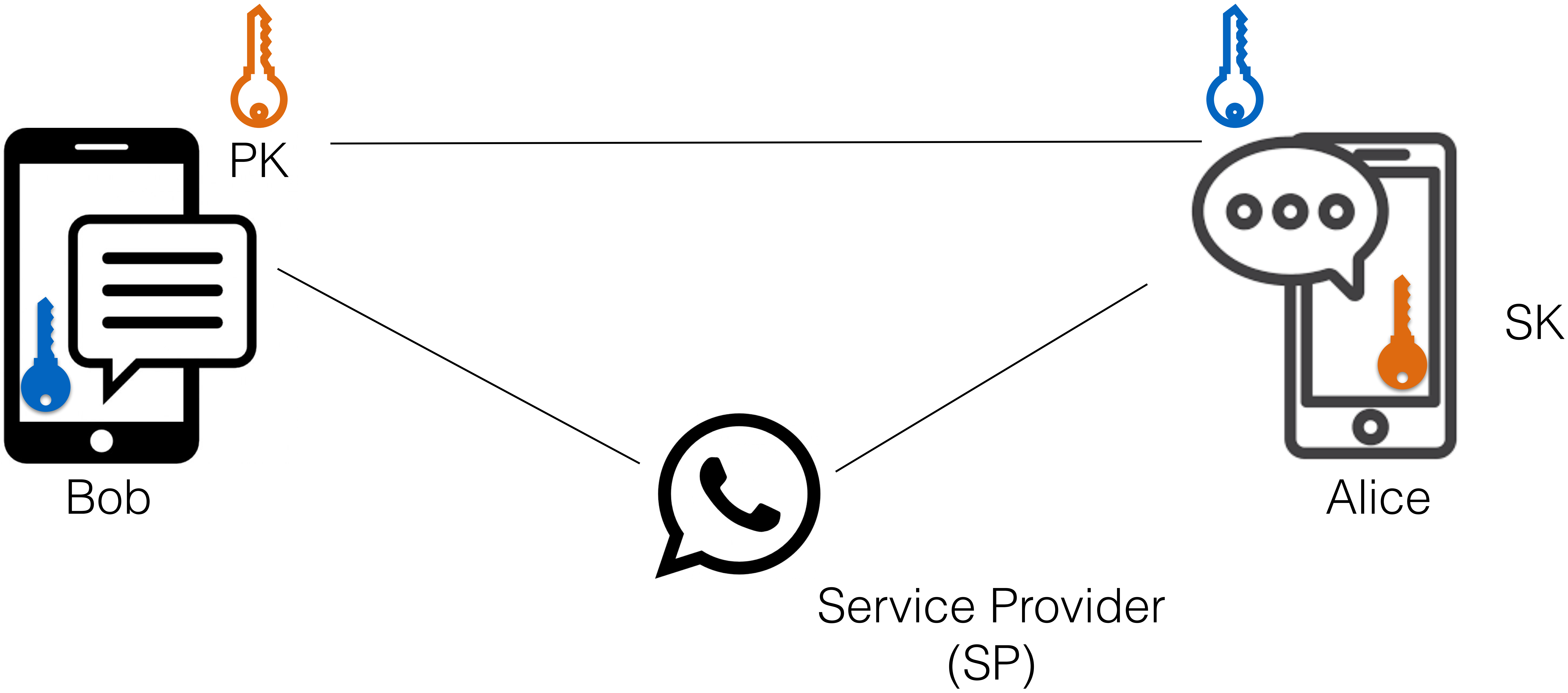
A Formal Treatment of Key Transparency Systems with Scalability Improvements

Nicholas Brandt, Mia Filić, **Sam A. Markelon***

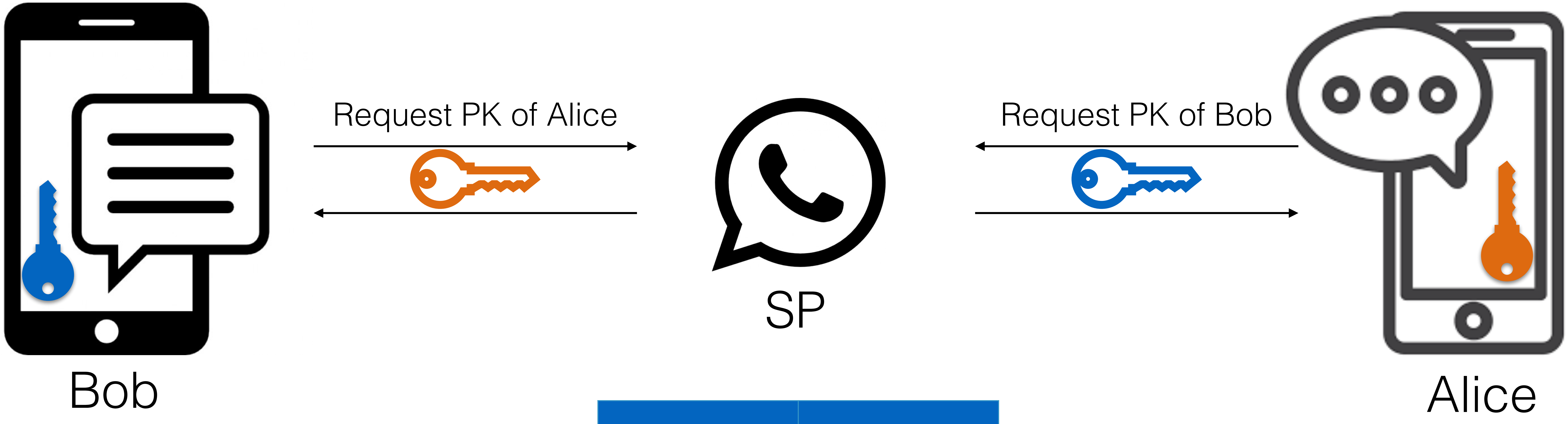
(Submitted: S&P '25)

* Alphabetical Ordering Used

E2E Encrypted Messaging

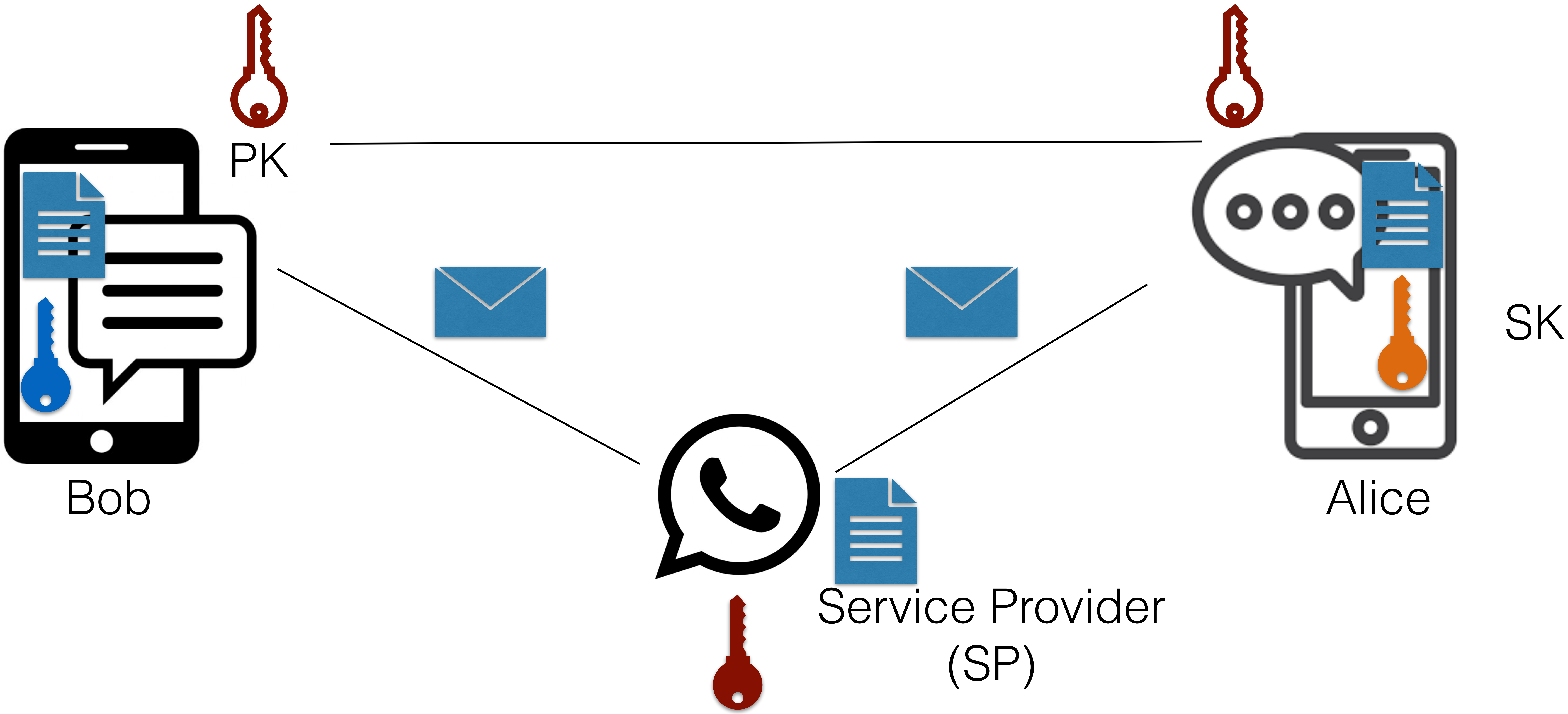


SP Maintained Key Directory

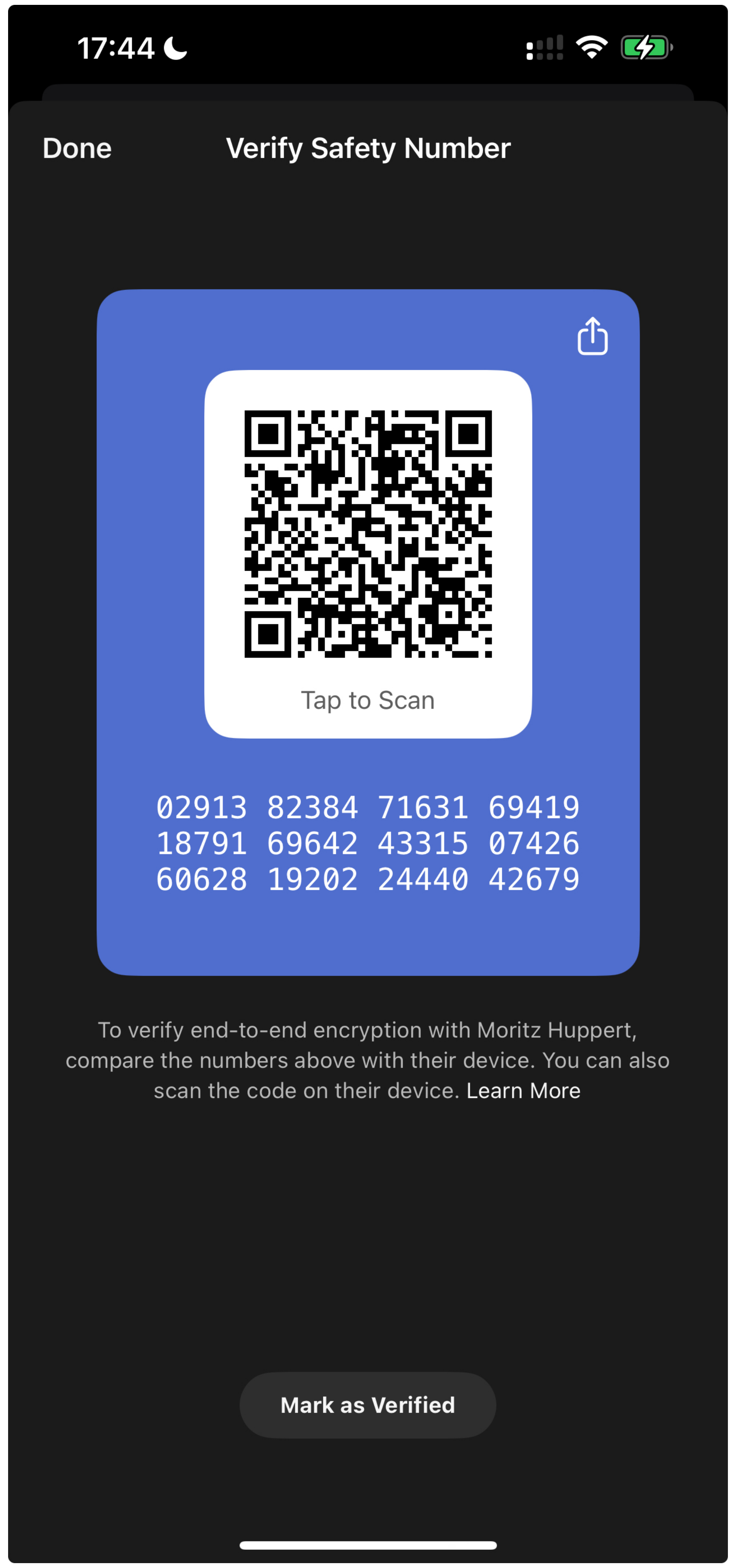


User ID	PK
Alice	PK_Alice
Bob	PK_Bob
…	…

Undetectable MitM Attack!



Traditional Approaches



CA

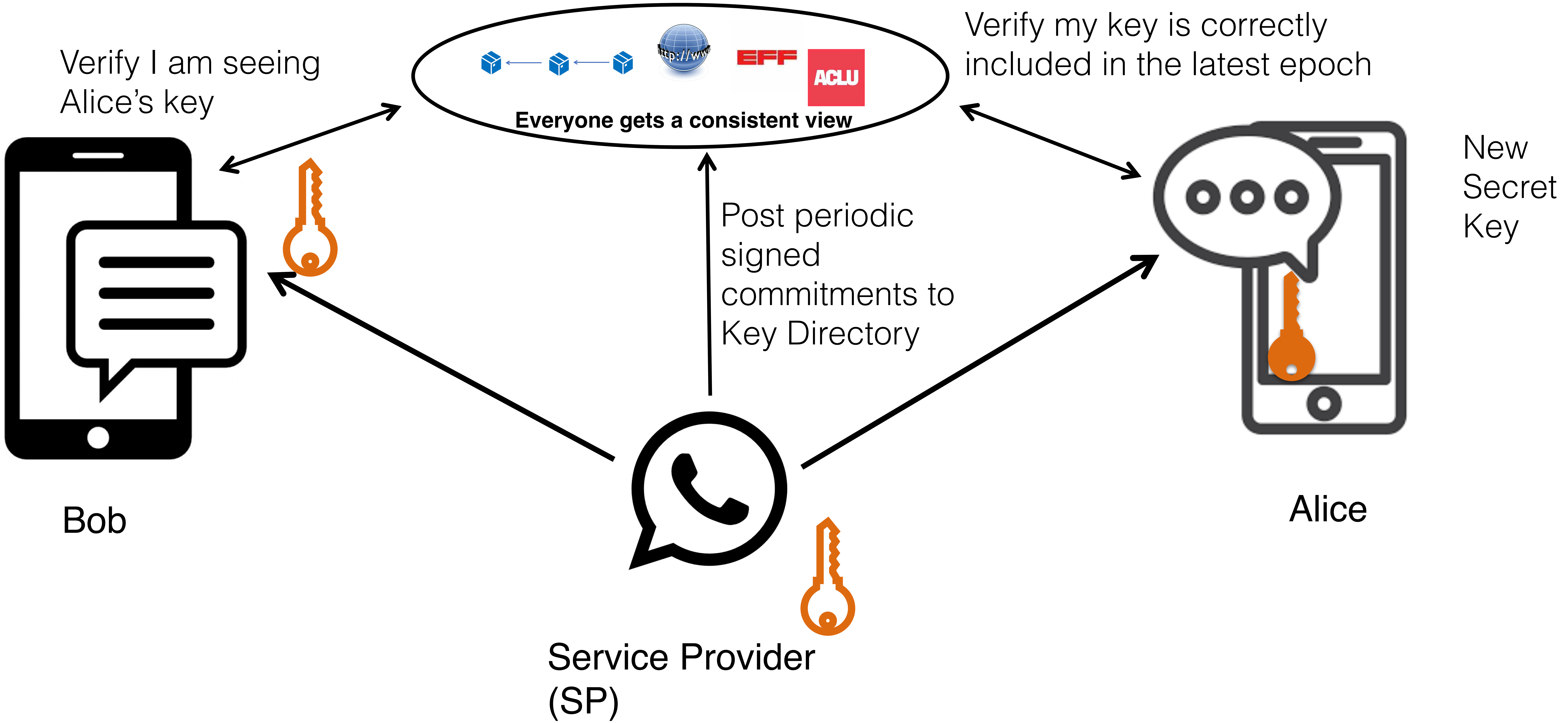
Enter Key Transparency!

- Require service provider to **commit** to key directory
- Key query responses come with a **proof** of correctness wrt commitment
- User can **monitor** their own key

Goals:

1. Seamless user experience; operations occur in background
2. Users do **not** have to manage **long-term secrets***
3. **Efficient** and easy to implement*; built on simple crypto primitives

Key Transparency Operation




KEY TRANSPARENCY
SO HOT RIGHT NOW
imgflip.com

Academia, Industry, and IETF

- CONIKS [MBBFF15]
- SEEMLess [CDGM19]
- Parakeet [MKSGOLL23]
- ELEKTRA [LCGJKM24]
- OPTIKS [LCGLM24]
- …and more!

zoom

IETF®

KEYTRANS working group

Current State of KT Formalization

KT as an ideal functionality

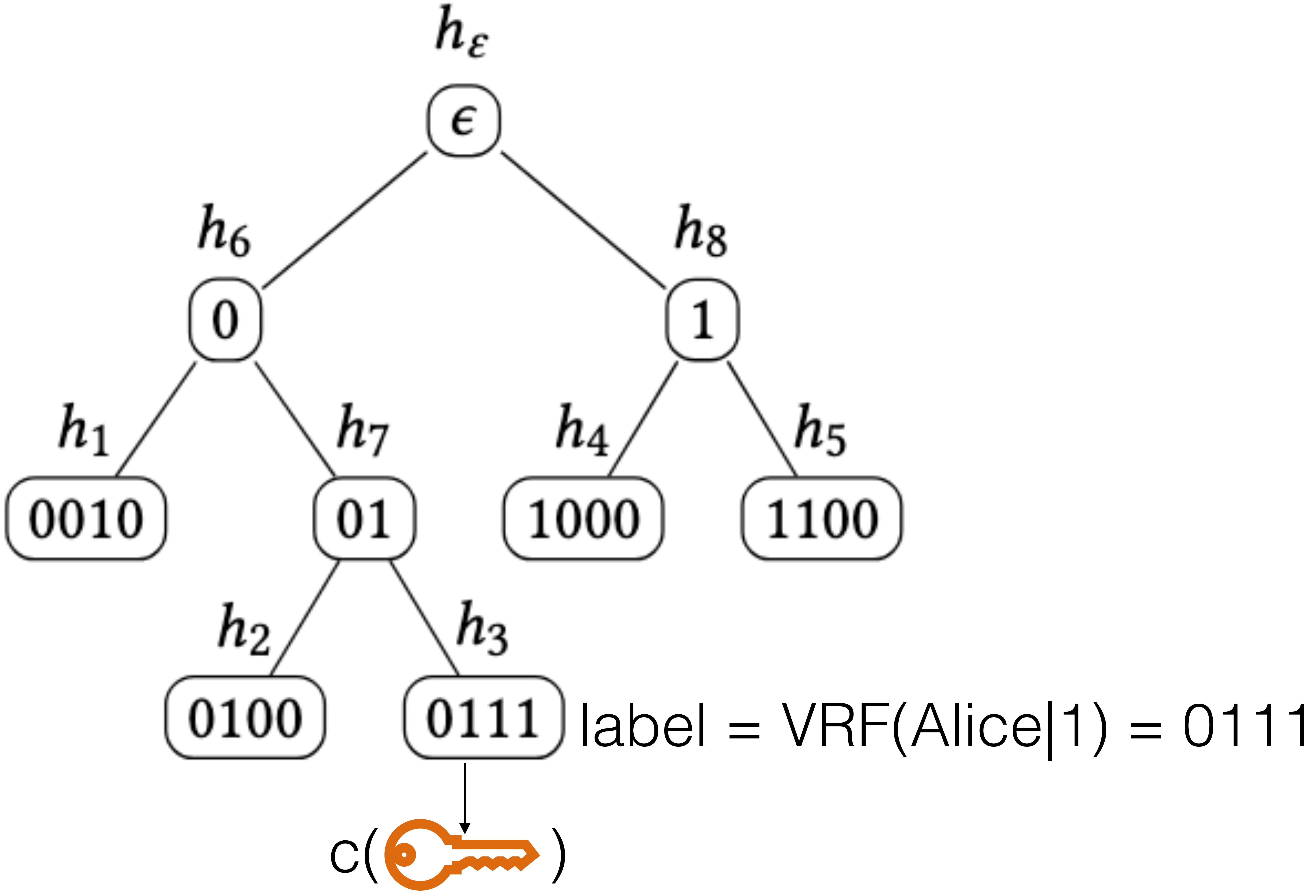
Functionality

Scheme

Protocol

KT Scheme Instantiation

aZKS



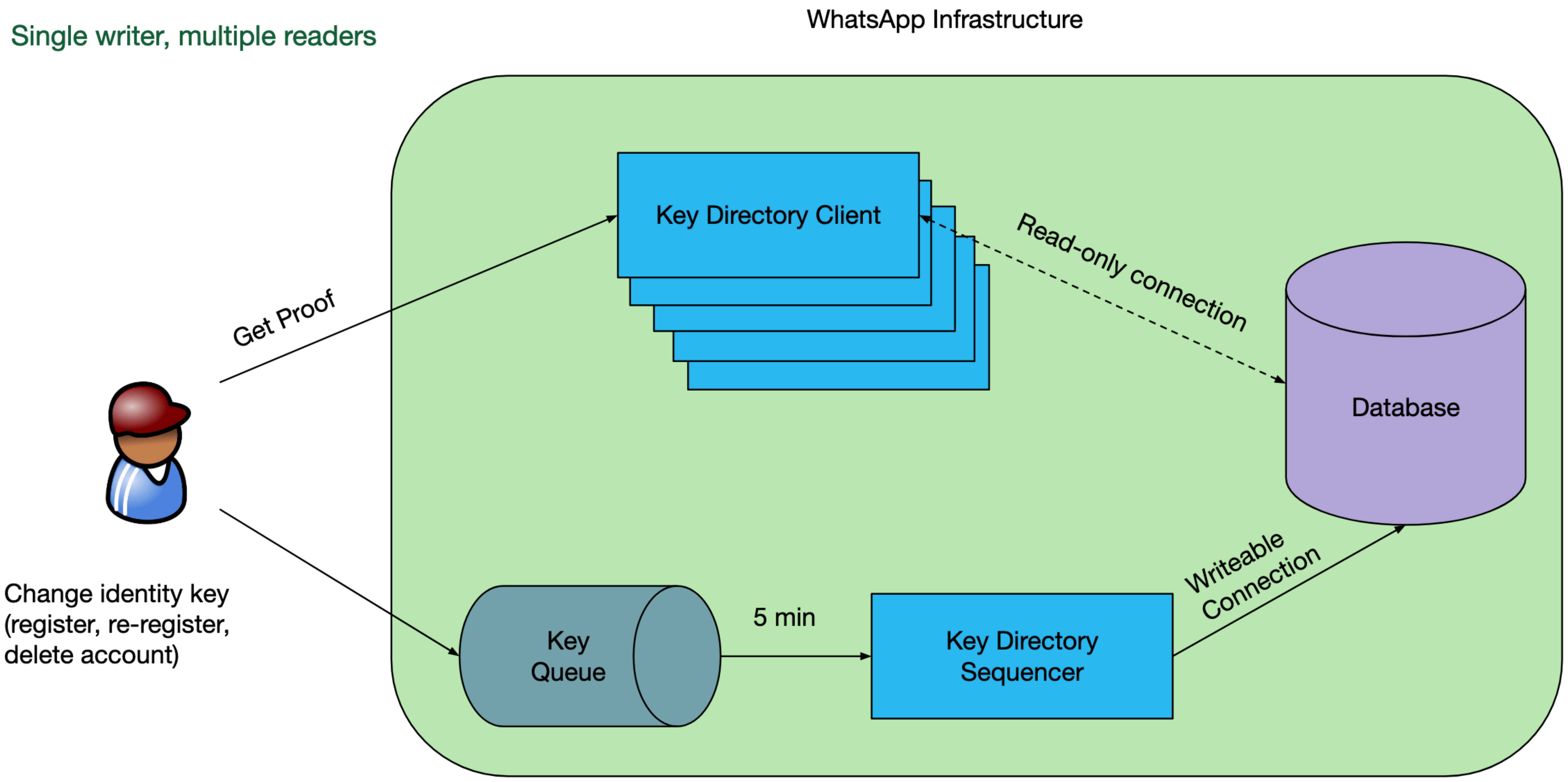
Before we get to our protocol…

Scalability?

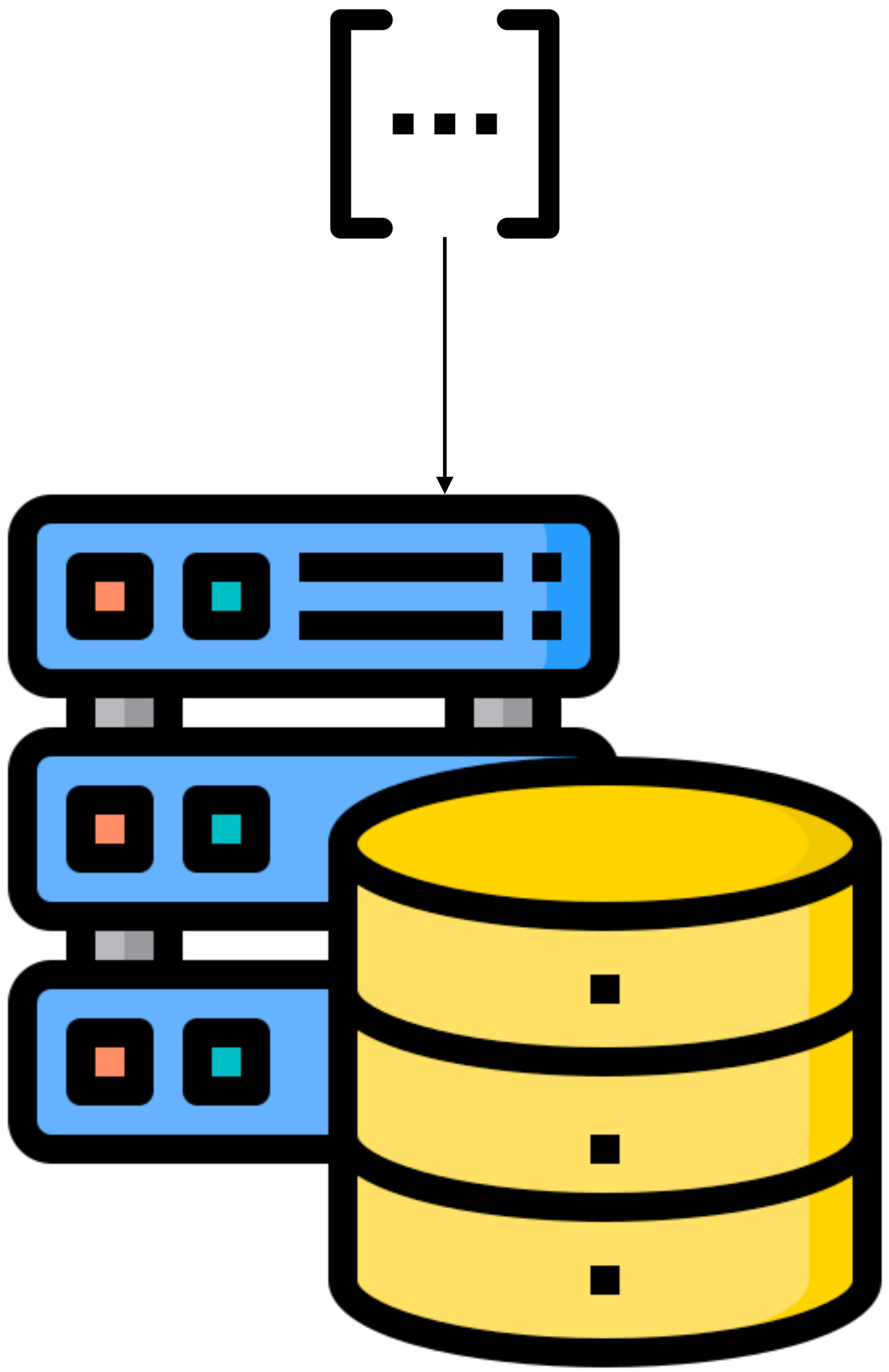
Distributed KT?

Infrastructure

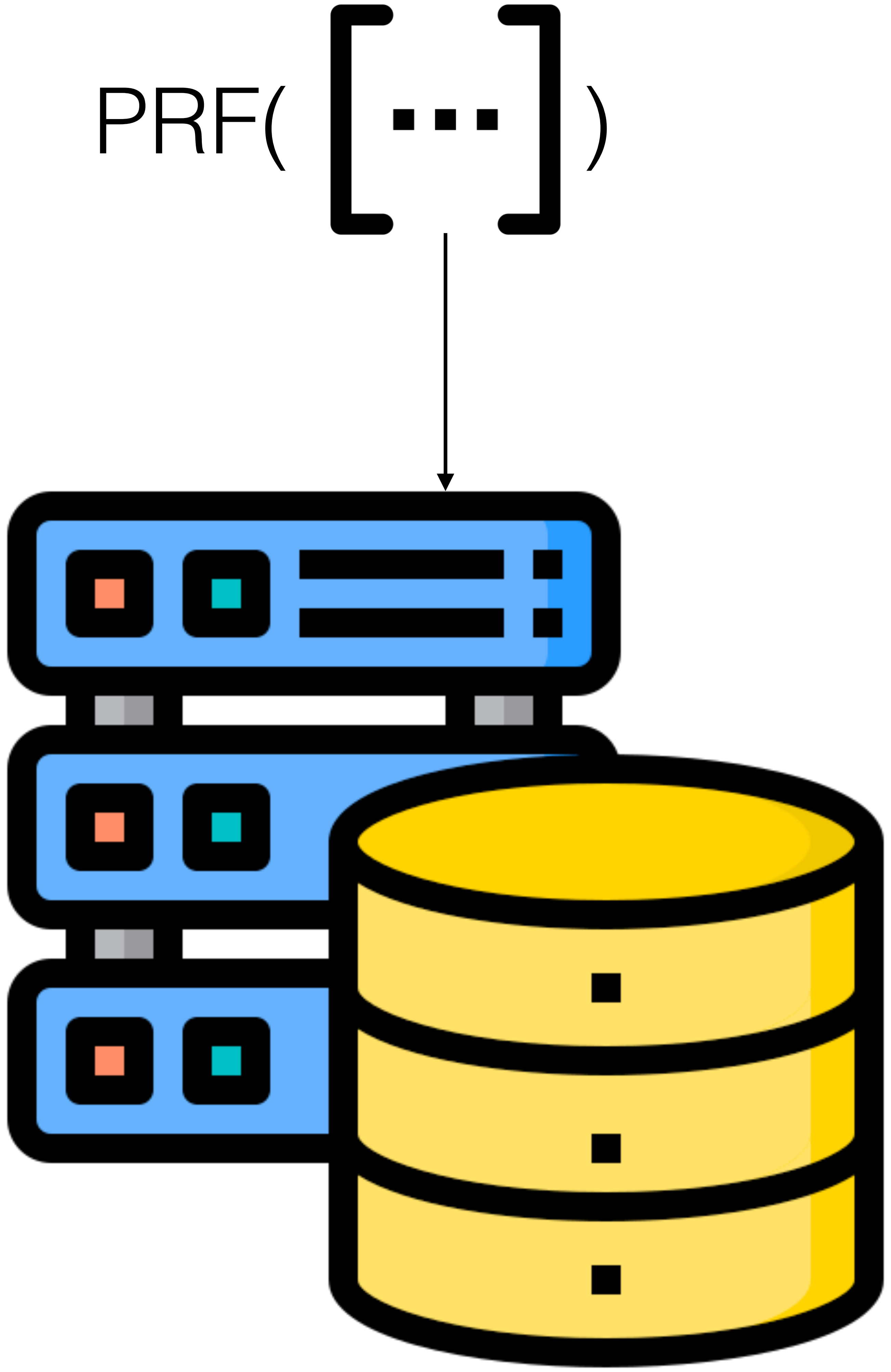
Single writer, multiple readers



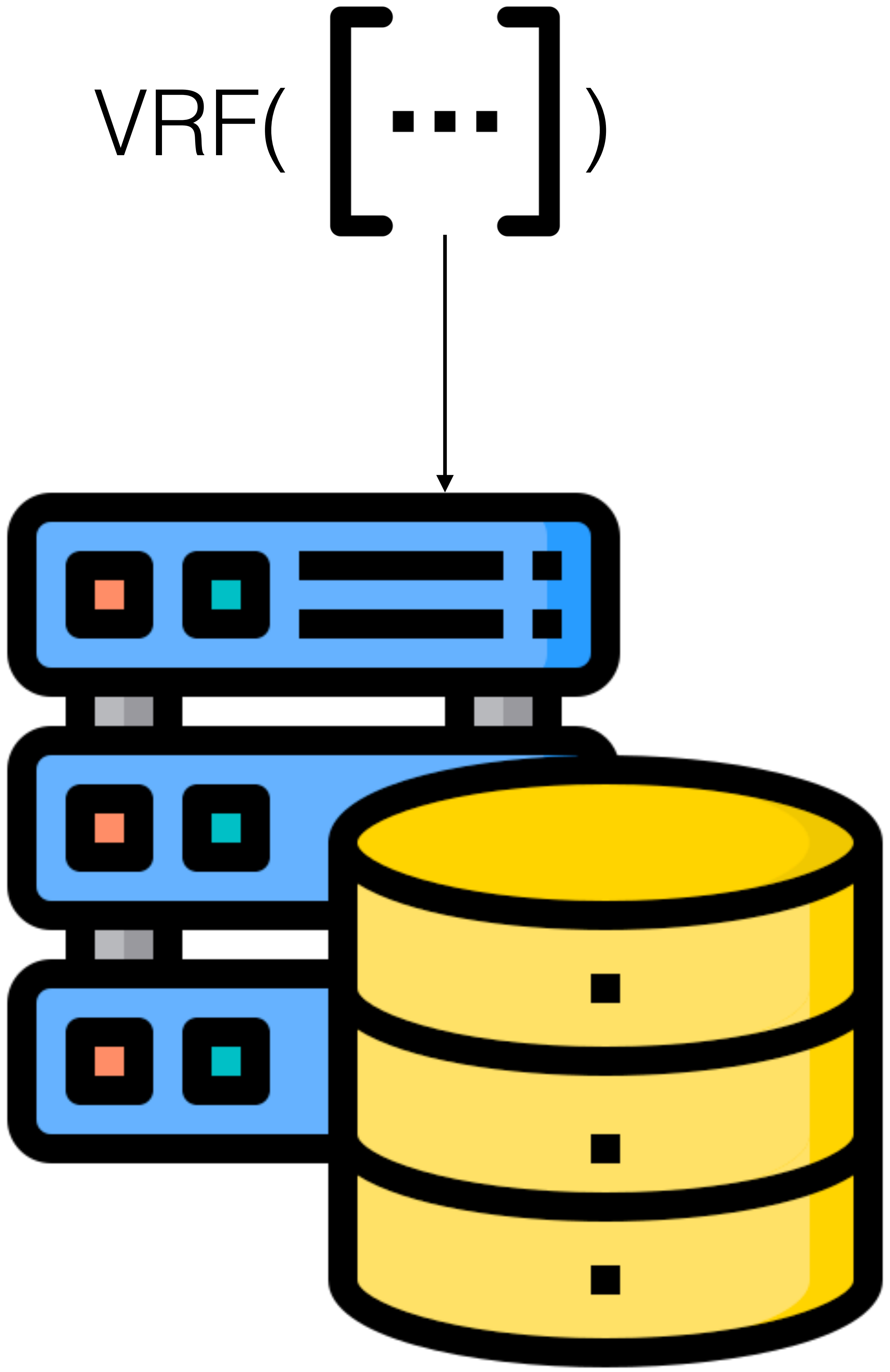
Per-Epoch Summary



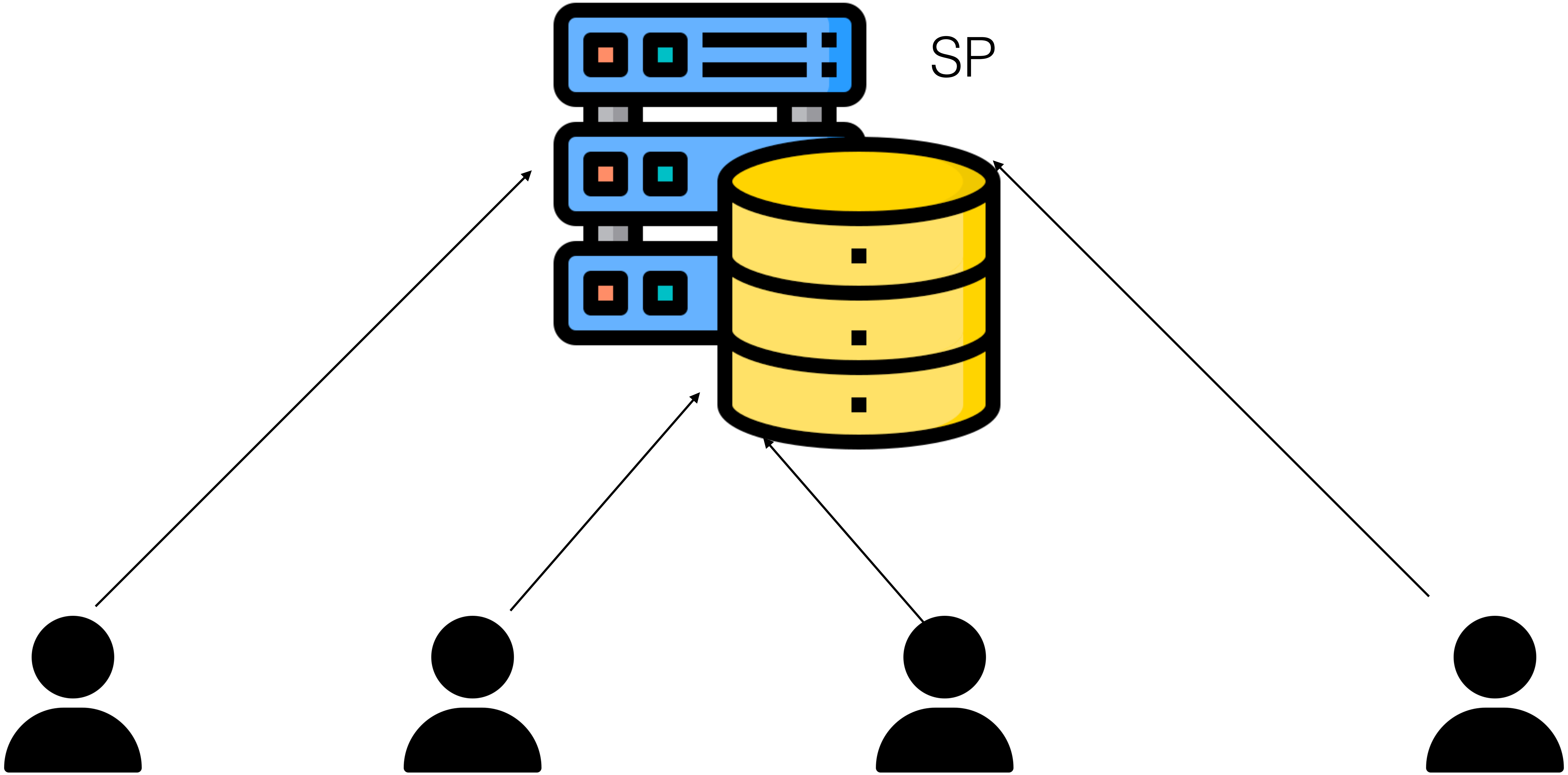
Per-Epoch Summary



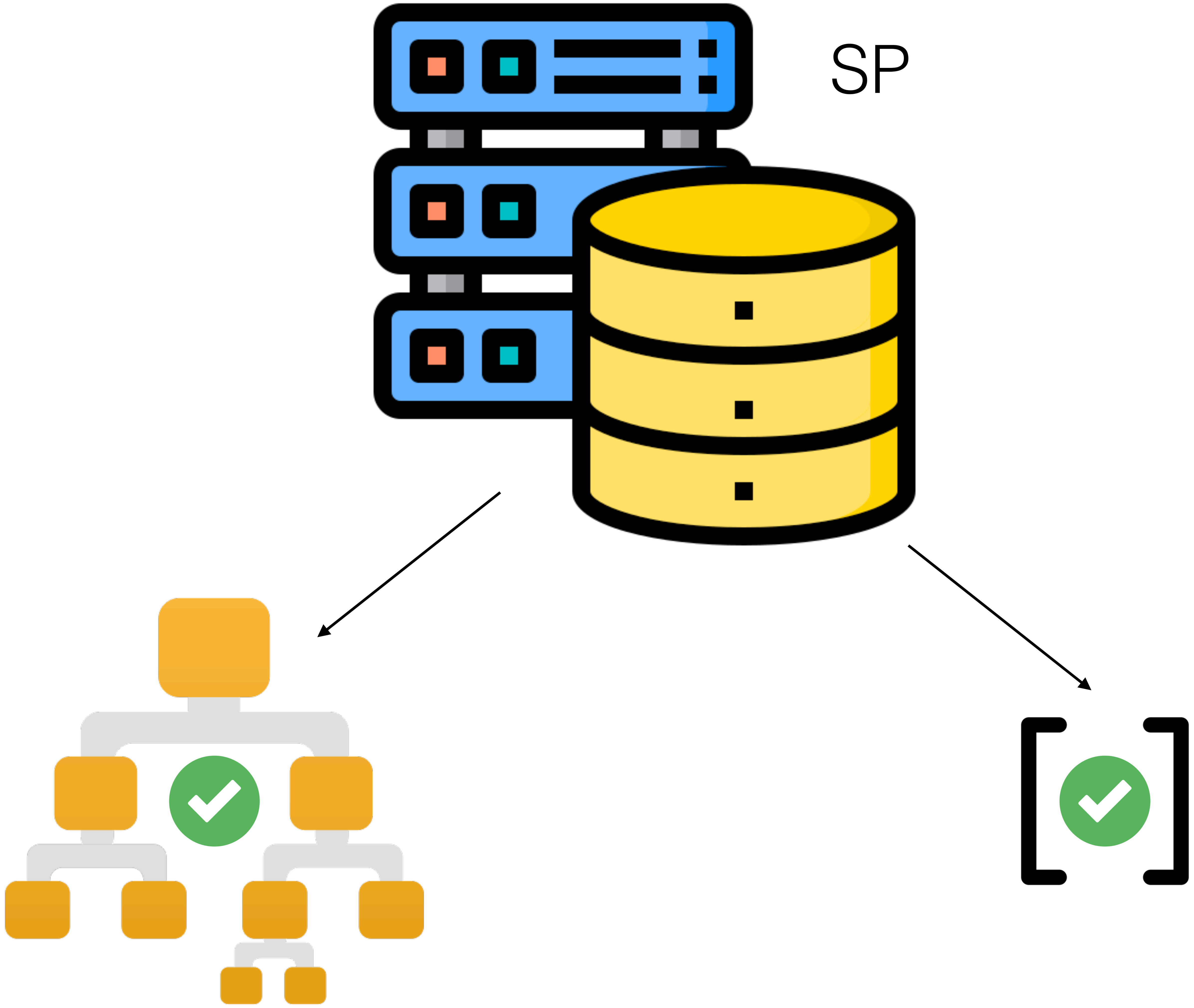
Per-Epoch Summary



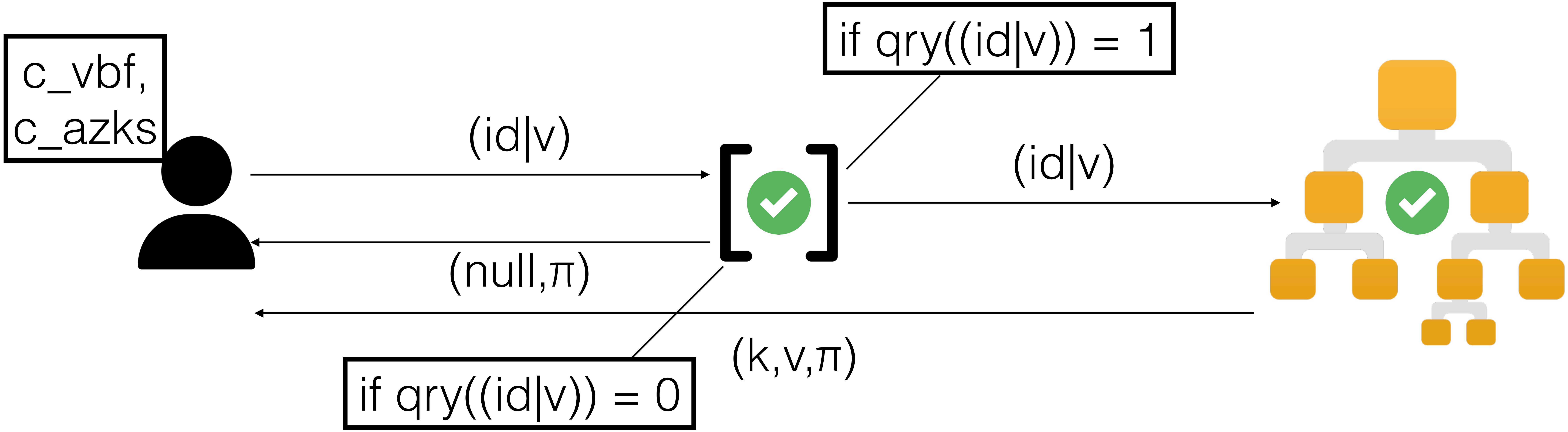
Our Protocol (Updates)



Our Protocol (Update Epoch)



Our Protocol (Querying)



Implementation and Experiments

- Build our protocol on top of Meta's AKD library
  - Modularity of our solution
- Building the VBF is practically free
- Querying the VBF is at one order of magnitude or more faster than aZKS
- Verifying VBF responses is 5x faster than aZKS
- *Very conservative speedup results*
- 64% reduction in query computation time for large scale deployments

Storage Comparison

- WhatsApp: 150,000 updates per epoch, a few billion keys stored
- Per-epoch VBF with FPR = 1%
  - 180 KB
- Compare with storing the entire aZKS state
  - SEEMLess: 27 TB
  - Parakeet is 2.2 TB

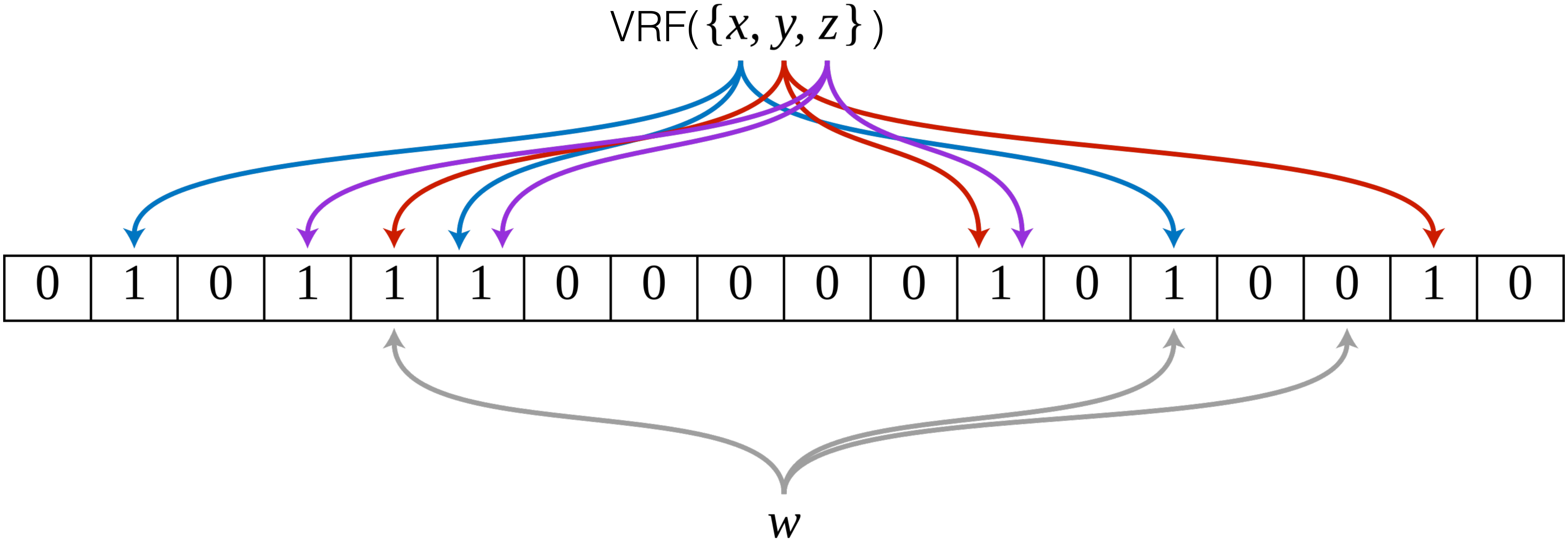
Compact, Private, and Verifiable Data Structures

Nicholas Brandt, Mia Filić, **Sam A. Markelon**, Thomas Shrimpton*

(In Progress. Target: TBD '25/'26)

* Alphabetical Ordering Used

Generalize and Formalize the VBF



By David Eppstein - self-made, originally for a talk at WADS 2007, Public Domain, https://commons.wikimedia.org/w/index.php?curid=2609777

Formalize a QVDS

- **Query Verifiable Data Structure**
- QVDS π = (GENKEY, REP, QRY, VFY)
- Satisfy completeness and verifiability properties

Making PDS Verifiable

- Generic transformation for a large class of PDS
  - Preprocess the inputs with a VRF!
- Properties
  - Generic completeness
  - Generic verifiability
  - Generic correctness
  - Generic privacy

Outstanding Work

- Concrete instantiations
- Updates
- Fancy query types
- Applications

Timeline

- **Completed and Submitted**
  - Compact Frequency Estimators in Adversarial Environments (CCS ’23)
  - Probabilistic Data Structures in the Wild: A Security Analysis of Redis (Submitted to CODASPY ’25)
  - A Formal Treatment of Key Transparency Systems with Scalability Improvements (Submitted to S&P ’25)
- **In Progress/Proposed**
  - Skipping Data Structures in Adversarial Environments (Target: CCS ’25 in April 2025)
  - Compact, Private, and Verifiable Data Structures (Target: TBD ’25/’26 in Spring or Summer 2025)
- **Thesis**
  - **Writing:** Spring and Summer 2025
  - **Defense:** Early Fall 2025

Publications

Compact Frequency Estimators in Adversarial Environments

CCS ‘23

Probabilistic Data Structures in the Wild: A Security Analysis of Redis

Submitted: CODASPY ‘25

Skipping Data Structures in Adversarial Environments

In progress: CCS ‘25

A Formal Treatment of Key Transparency Systems with Scalability Improvements

Submitted: S&P ‘25

Compact, Private, and Verifiable Data Structures

In progress: TBD ’25/‘26

The DecCert PKI: A Solution to Decentralized Identity Attestation and Zooko’s Triangle*

IEEE DAPPS ‘22

Leveraging Generative Models for Covert Messaging: Challenges and Tradeoffs for “Dead-Drop” Deployments

CODASPY ‘24

*Best paper award

Questions

?

Sam A. Markelon
smarkelon@ufl.edu