HybrIDX: New Hybrid Index for Volume-hiding Range Queries in Data Outsourcing Services

Kui Ren, **Yu Guo**, Jiaqi Li, Xiaohua Jia, Cong Wang,
Yajin Zhou, Sheng Wang, Ning Cao, and Feifei Li

Trend of data outsourcing services

- **Digital data to reach 175 zettabytes by 2025**

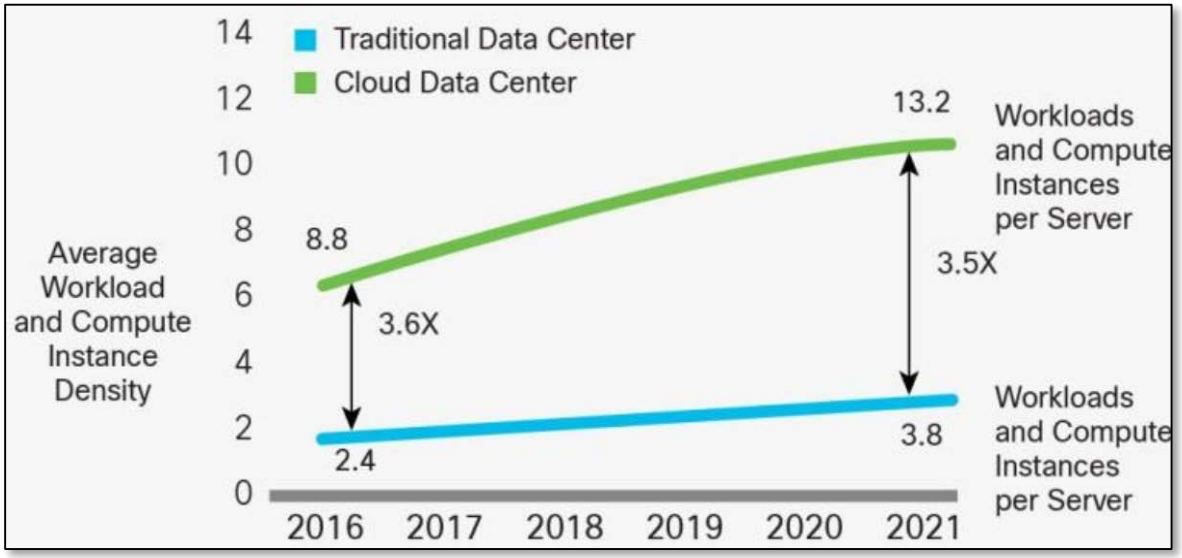
  **IDC Report, Executive Summary: Data Growth, Business Opportunities, and IT Imperatives, 2019.*

- **Data outsourcing demand remains strong**
  - Increasing adoption rate
  - Big data analytics in cloud

amazon web services™

Microsoft Azure

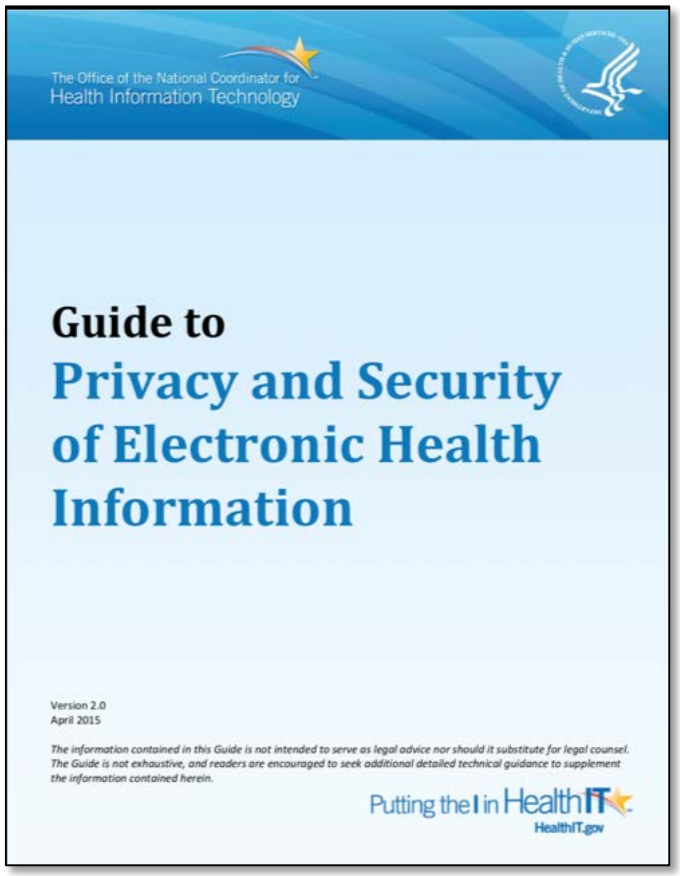
Alibaba Cloud



Source: Cisco Global Cloud Index: Forecast and Methodology, 2016-2021.

Why encrypted search?

- **Sensitive data demands encrypted storage**
  - General Data Protection Regulation (EU)
  - California Consumer Privacy Act



- **Search is ubiquitous**

"if your practice has a breach of encrypted data [...] it would not be considered a breach of unsecured data, and you would not have to report it"

-- Guide to privacy and security of electronic health information, 2015

Our effort

- **Volume*-hiding range queries over encrypted data**

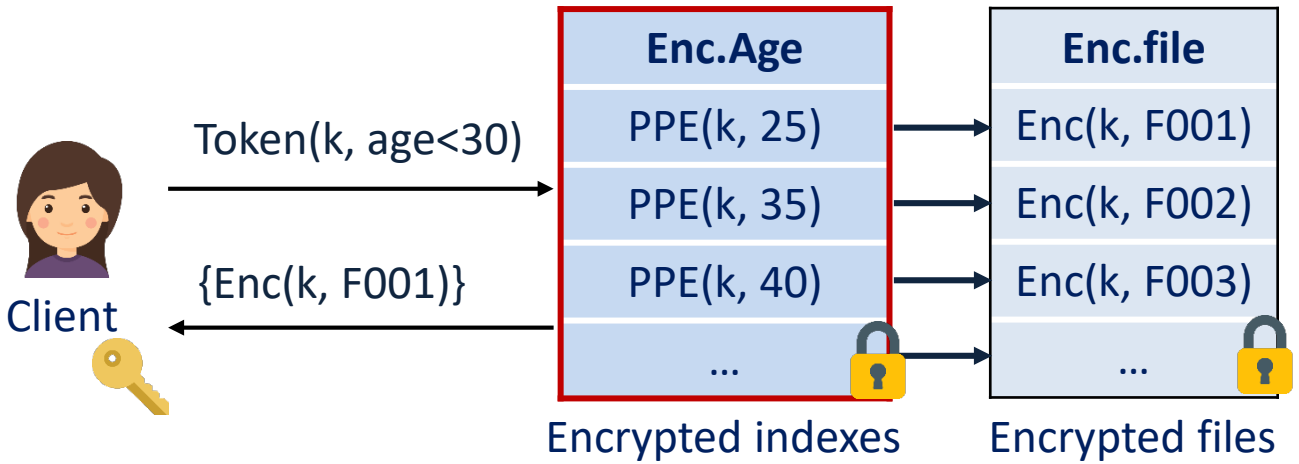
SQL: SELECT * FROM table_user WHERE age > 30

An example of range query SQL statement

- **Significantly reduced leakage profile**
  - Hiding the number of range query results (volume)
  - Obfuscating the results co-occurrence across different range queries

- **More resilience against recent attacks**
  - [F. B. Durak et al. CCS'16], [P. Grubbs et al. S&P'17]
  - [M.-S. Lacharit and B. Minaud S&P'18], [Z. Gui et al. SIGSAC'19] …

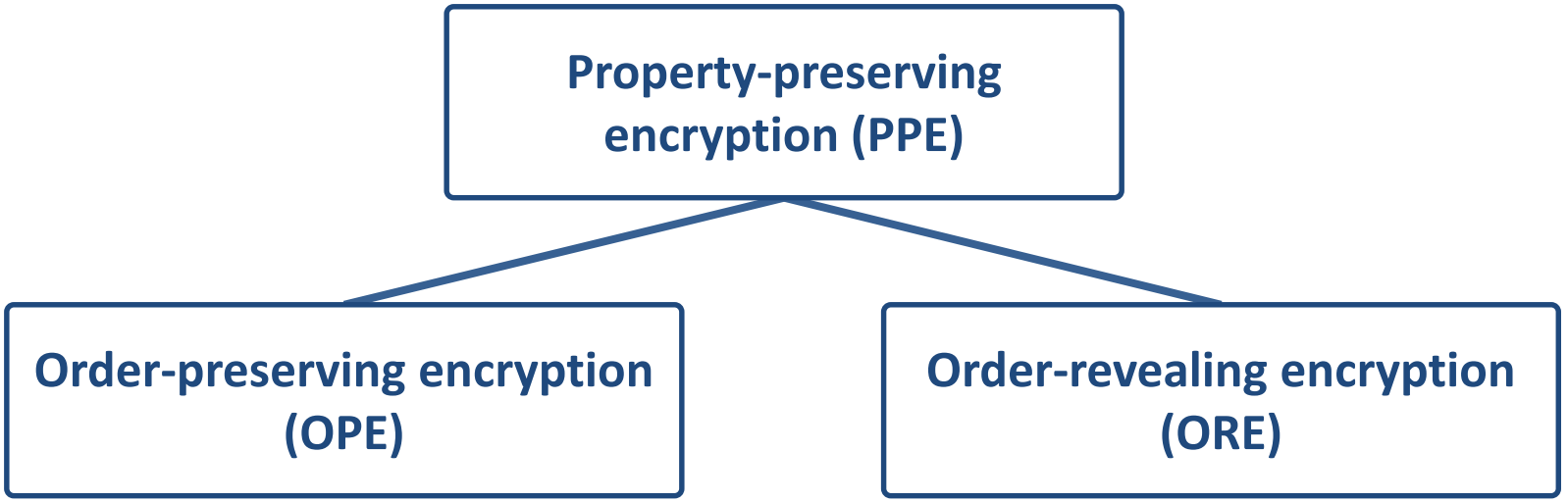
An example of encrypted range query



PPE: Some property-preserving encryption that allows range query

- **An encrypted index allows the server to conduct various query functionalities in the ciphertext domain**

Existing solutions



[Boldyreva et al. EUROCRYPT'09]
[Popa et al. SP'13]
[Kerschbaum et al. CCS'15] …

- Pros: sub-linear query complexity
- Cons: deterministic enc. that leaks frequency and order

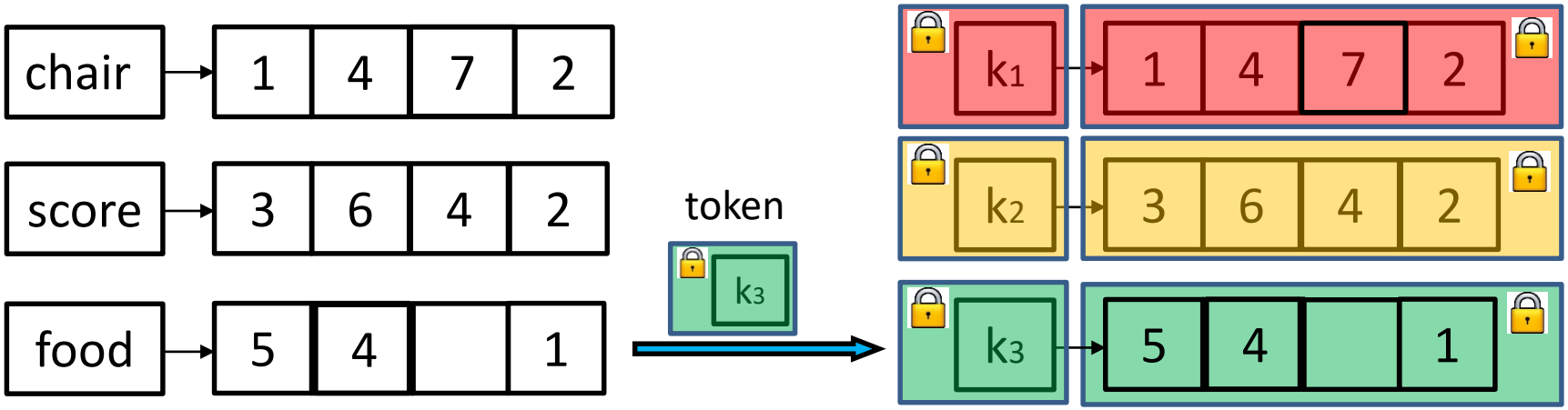
[Chenette et al. FSE'16]
[Lewi and Wu CCS'16] …

- Pros: protect frequency, and some degree of order-leakage
- Cons: linear query complexity

- **But their leakage profiles can still be abused**
  - Mainly from the result co-occurrence pattern and the volume

Simple counting attacks on volume

- Observation: when a query returns a unique number of files (volume), it can immediately be guessed! [Cash et al., CCS'15]



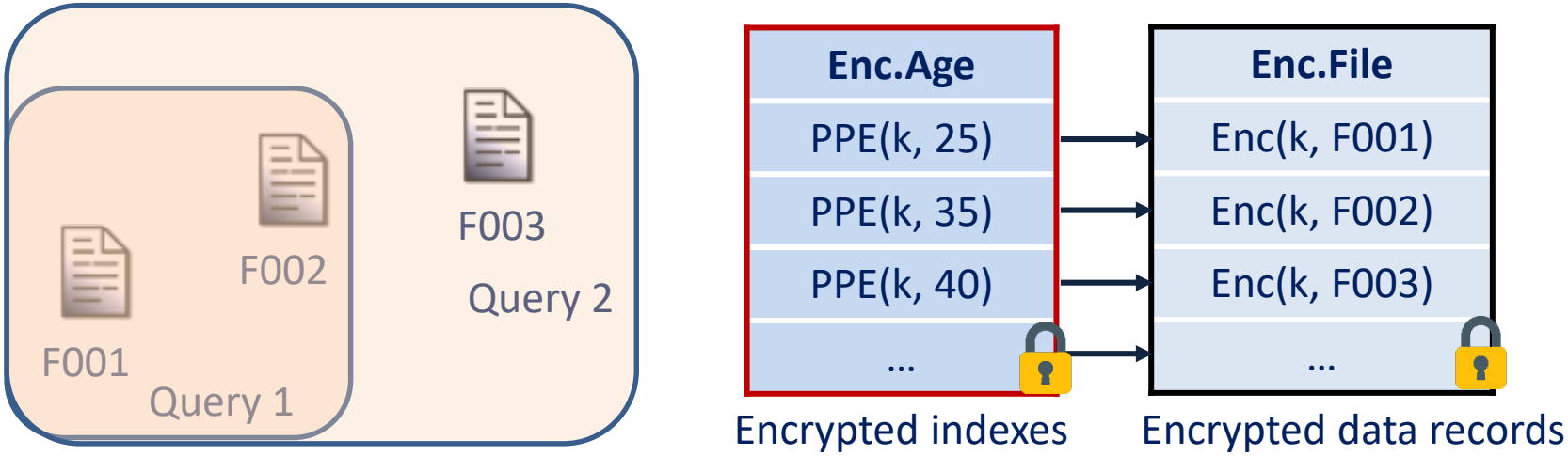
Index (Known by adversary)

3 files are matched, so the query keyword must be "food"!

Similar intuition can also be applied to range query

Attacks on result co-occurrence

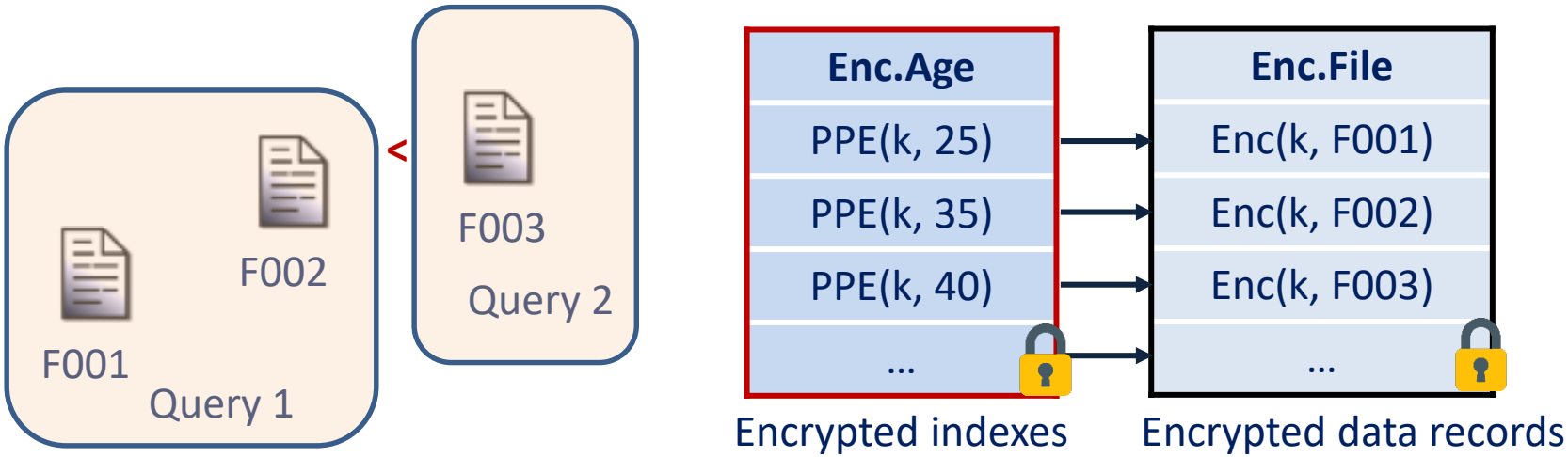
- Observation: infer order of values by observing the result co-occurrence in different range queries [Lacharit et al., S&P'18]



Q1 (age<?): {F001, F002} Q2 (age<?): {F001, F002, F003}

Attacks on result co-occurrence

- Observation: infer order of values by observing the result co-occurrence in different range queries [Lacharit et al., S&P'18]



Q1 (age<?): {F001, F002} Q2 (age<?): {F001, F002, F003}

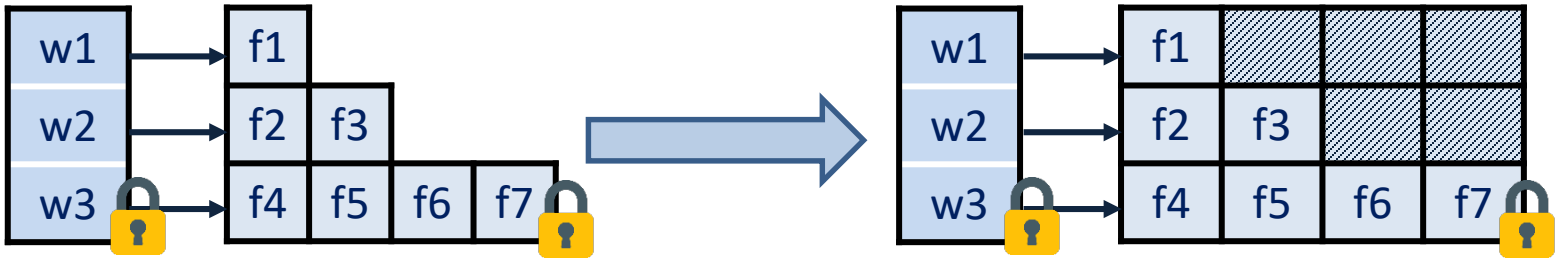
Goals and challenges

- Need to significantly suppress the leakages
  - More resilience against inference-attacks on encrypted range query

- Our plan:
  - Borrow volume-hiding structure from encrypted keyword search
  - Obfuscate the results co-occurrence among different queries
  - Still maintain range query search efficiency

Volume-hiding keyword search

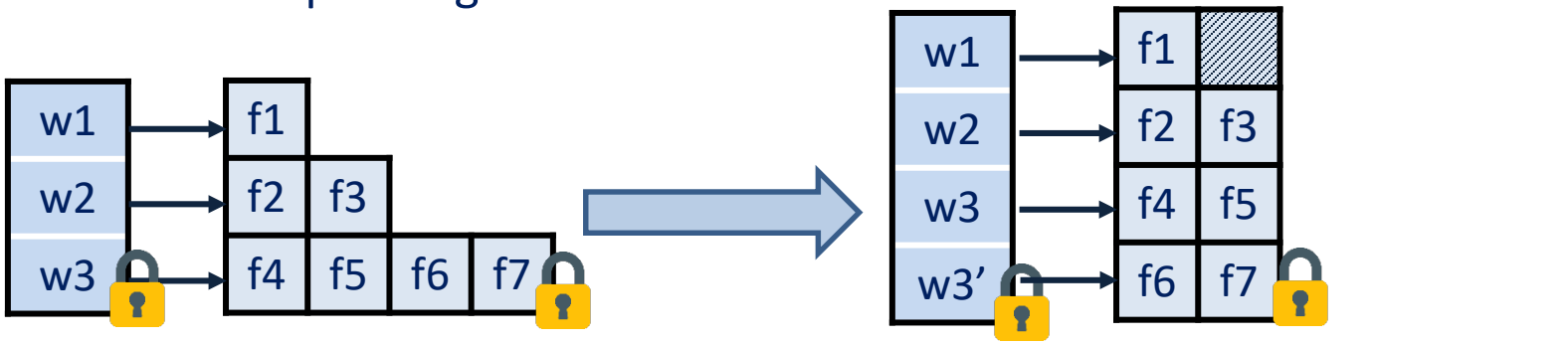
- Naïve padding over predefined search results:



Inverted indexes

Naïve padding

- Bucketization-based padding:
  - Reduced padding overhead

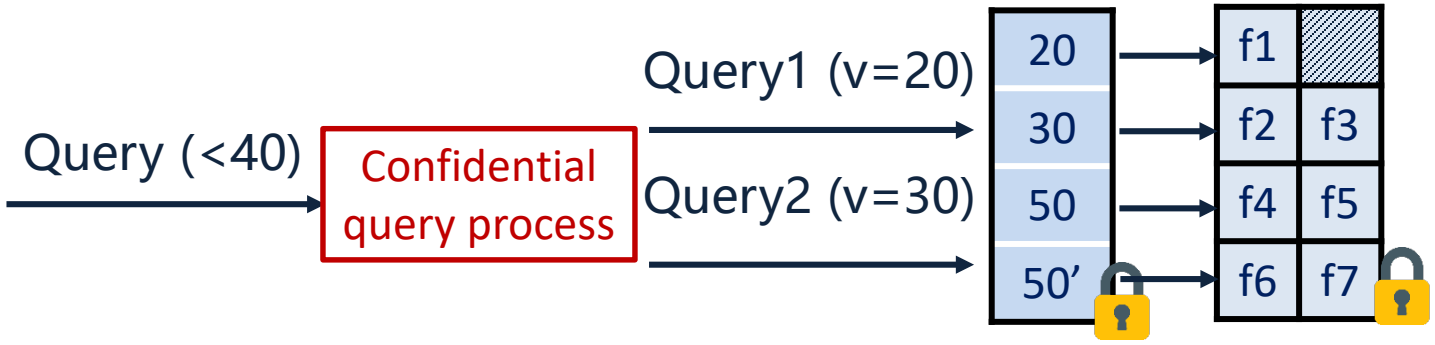


Inverted indexes

Desired volume-hiding structure

Towards volume-hiding range query

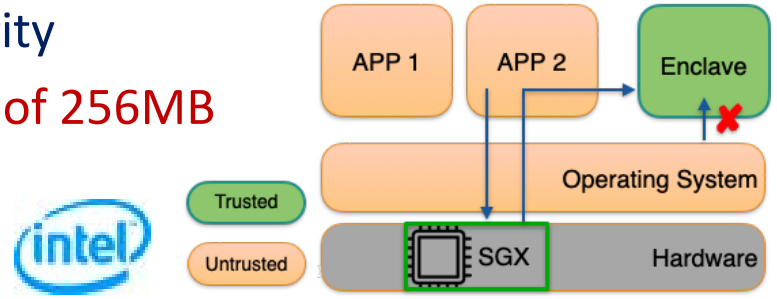
- But range query cannot be pre-defined
  - Unable to forecast all range-matched results
  - The maximum volume can be the entire dataset
- Treat each value in the query range as “keyword”
  - Convert range query into multiple “keyword” search (aka sub-queries)



- A hybrid design: volume-hiding structure + TEE (SGX)

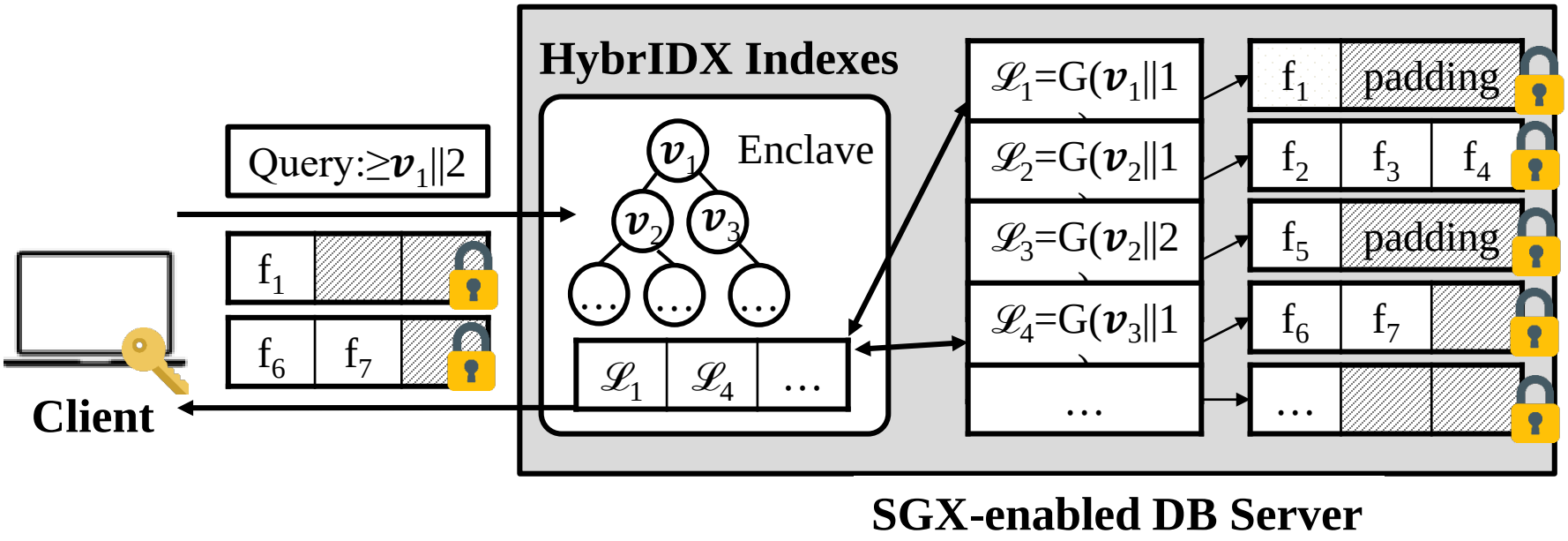
Why not put everything inside TEE?

- We focus on Intel SGX
  - Hardware-enabled trusted execution environment (Enclave)
  - Provide confidentiality and integrity
  - Limited by the current maximum of 256MB



- We only use TEE for two aspects:
  - Confidential range query processing (sub-query conversion)
  - Secure result caching for co-occurrence pattern obfuscation

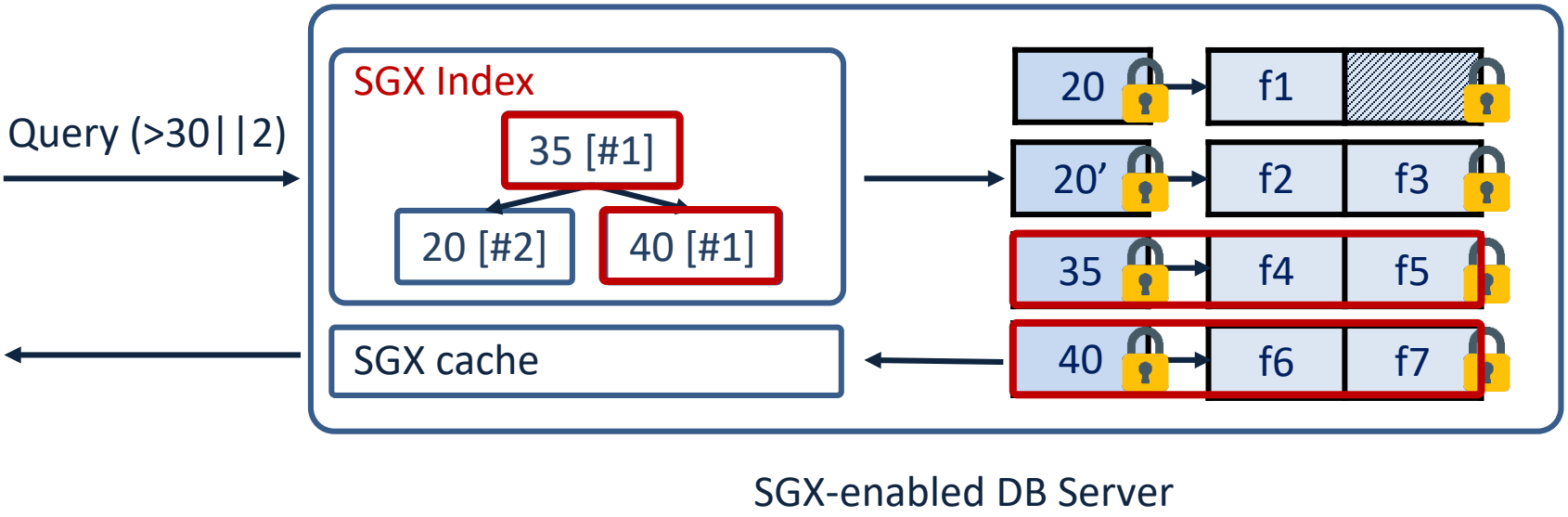
HybrIDX architecture



- In enclave: A tree-based range index and a trusted cache (fixed size)
- External: An encrypted volume-hiding structure, with file blocks and padding

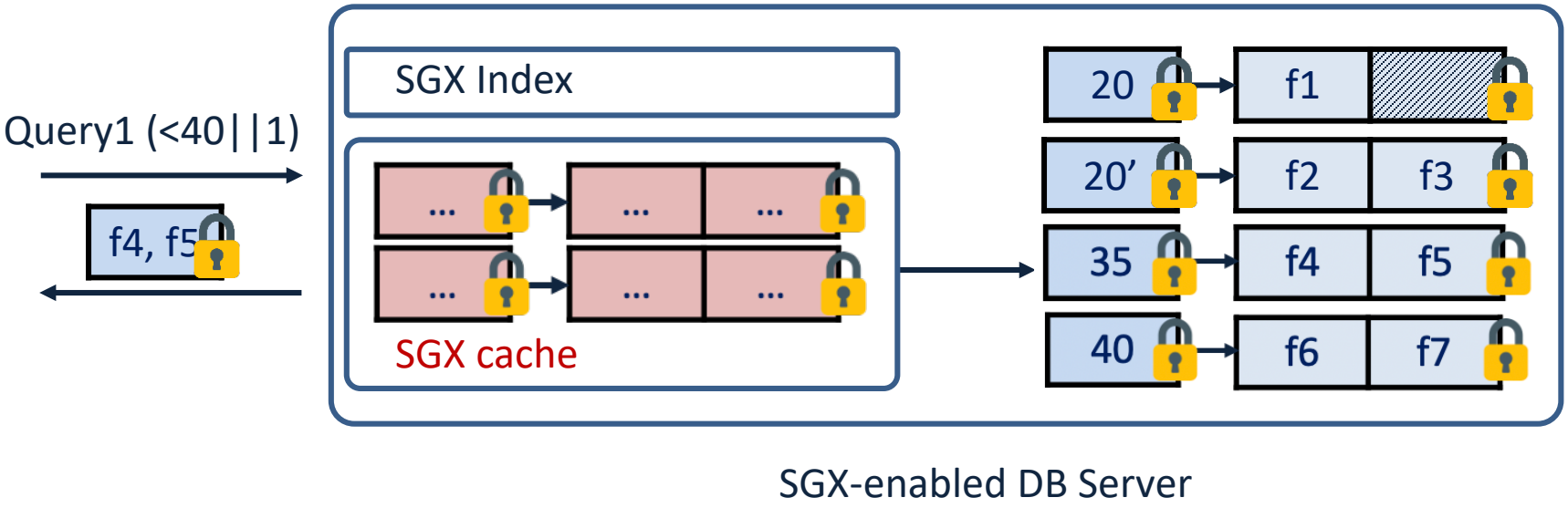
HybrIDX: query in action

- Prior query results are cached inside enclave
  - Subsequent query is processed with cache
  - Trigger cache swapping and shuffling when needed



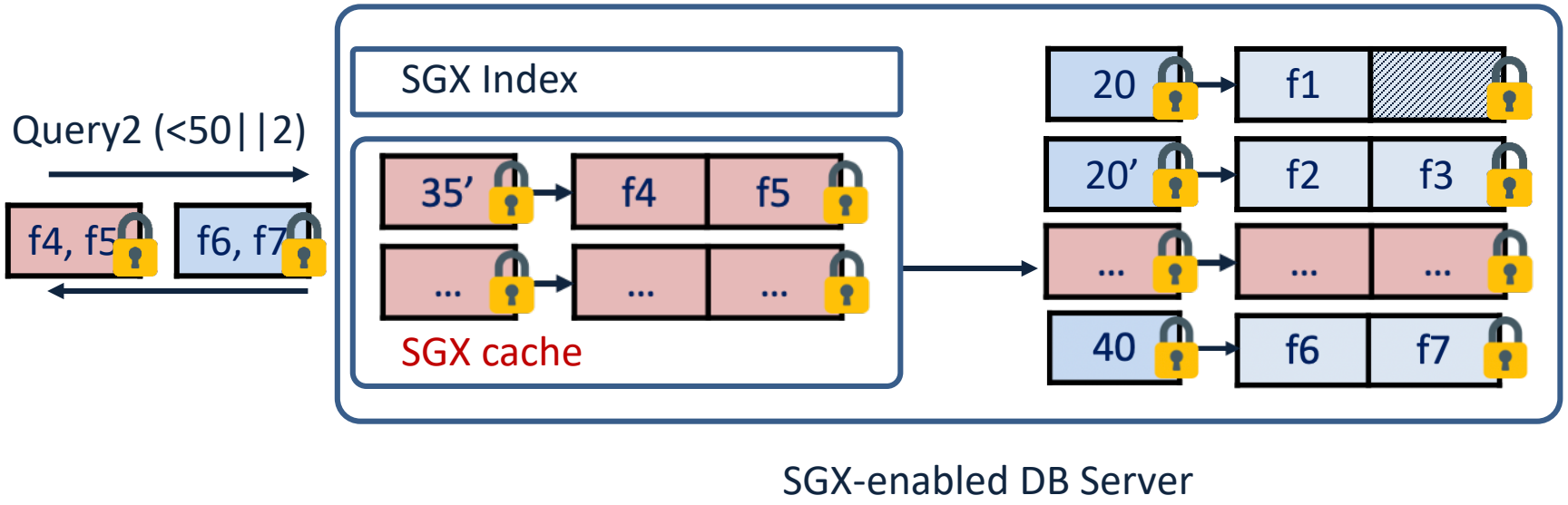
HybrIDX: caching and shuffling

- Query process from cache and external structure
  - Identify the external items to be returned
  - Randomly choose enclave cached items for eviction
  - Upon shuffling and re-encryption, swap them with external items



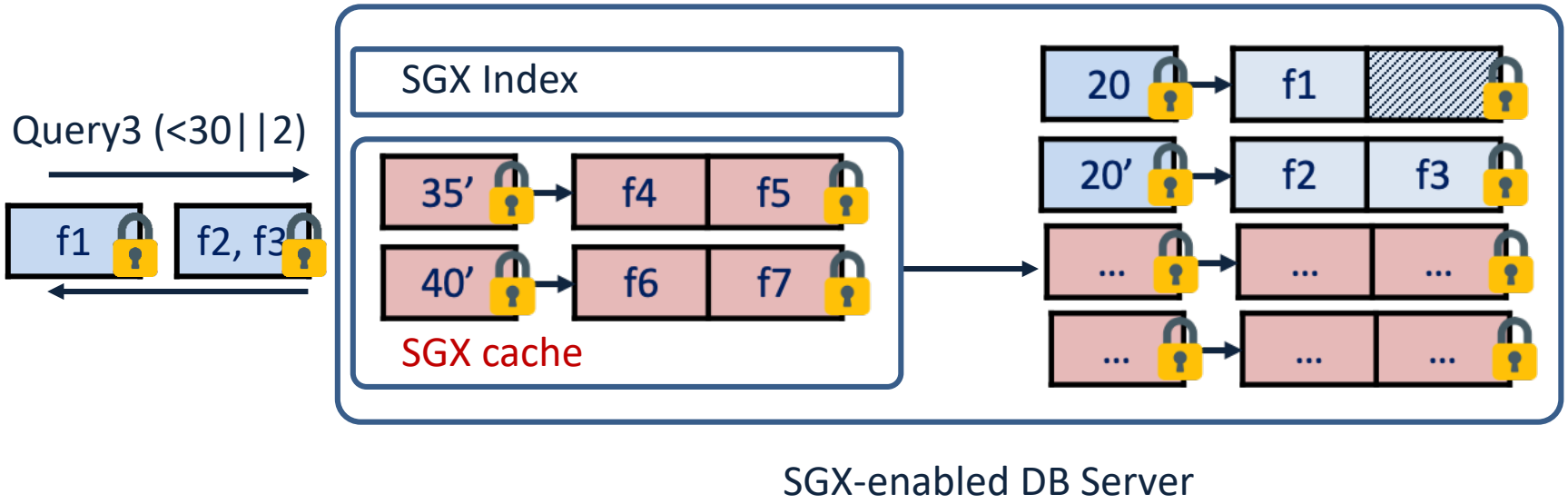
HybrIDX: caching and shuffling

- Query process from cache and external structure
  - Identify the external items to be returned
  - Randomly choose enclave cached items for eviction
  - Upon shuffling and re-encryption, swap them with external items



HybrIDX: caching and shuffling

- Query process from cache and external structure
  - Identify the external items to be returned
  - Randomly choose enclave cached items for eviction
  - Upon shuffling and re-encryption, swap them with external items



Security strength

- Adversarial server only views the following leakage profiles:
  - Partial access set A_q = set of (L, v) $\in$ SGX_{out} returned for q
  - Eviction set E_q = set of (L, v) $\in$ SGX_{in} evicted from enclave for q
  - Eviction history set EHP_q= {{q': (L, v) $\in$ A_q and (L, v) $\in$ $E_{q'}$ in Q} : q $\in$ Q}

$$L_{query}(q)=(A_q, E_q, EHP_q)$$

  - Remark： L--> prf label, v--> encrypted value, Q --> query list.

- The larger ratio of cache-size over query result size (volume), the better uncertainty of item tracking across queries.

Towards larger cache/response ratio

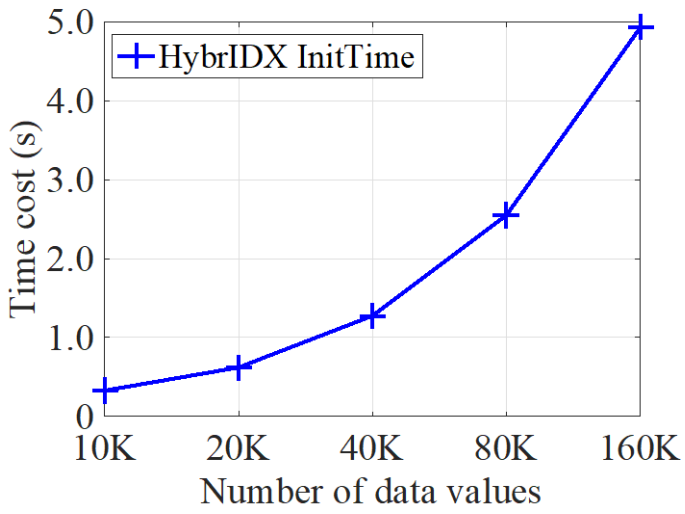
- Applications do not need to display all results at once

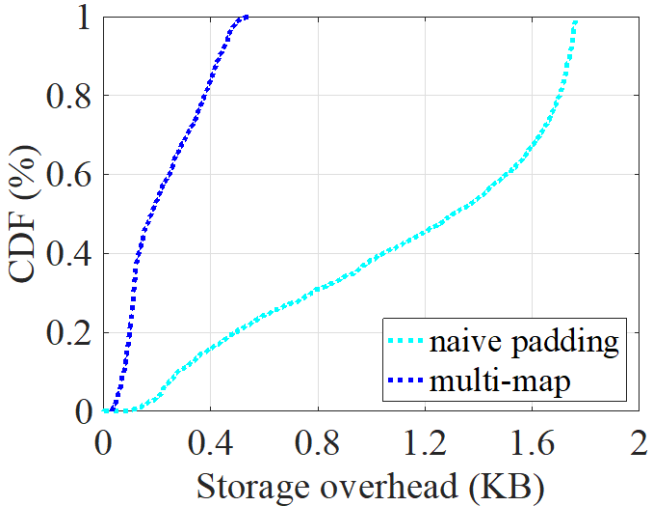
- Display a subset of results per round
  - Show more when needed
  - Easily supported with enclave in deployment
    - Inspired by similar practice from Oblix [SP'18] (for a different purpose)

Experiments

- Data sets: 160K data records and randomly assign them to 1K index values
- SGX-enabled server with an Intel(R) Core(TM) i7-7700 processor (3.6 GHz) and 16GB RAM
- Intel SGX SSL and OpenSSL (v1.1.0g)
- Symmetric encryption via AES-128 and the pseudo-random function via HMAC-256

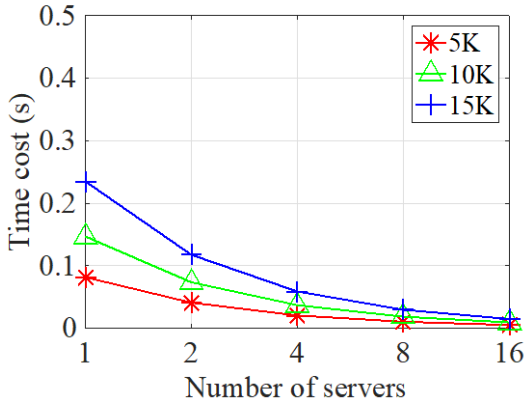
Setup cost

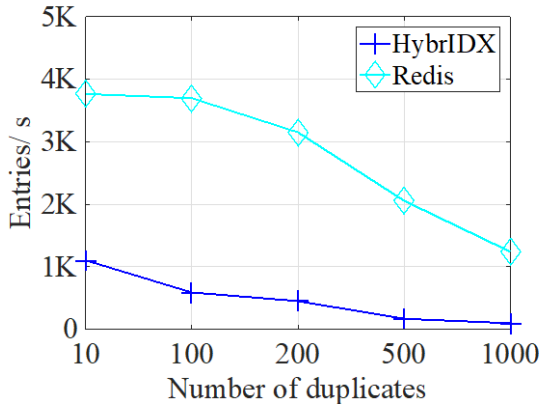


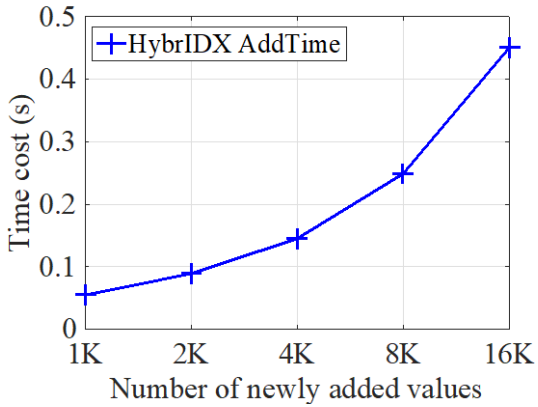


- For 160K records, the client takes less than 5s
- Padding overhead for over 80% load-factor indexes are less than 0.4 KB

Query performance







- For 10K values, the query latency is around 0.14s
  - 18× faster compared to the ORE-based scheme

Conclusion and future work

- Encrypted range query with much reduced leakage
  - hiding the volume of query results
  - obfuscating the results co-occurrence across queries
- Hybrid design: volume-hiding structure + TEE (SGX)
- To-do: build real-world applications on top
- Thank you