Dynamic Searchable Encryption in Very Large Databases: Data Structures and Implementation

David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, **Marcel Roşu** and Michael Steiner

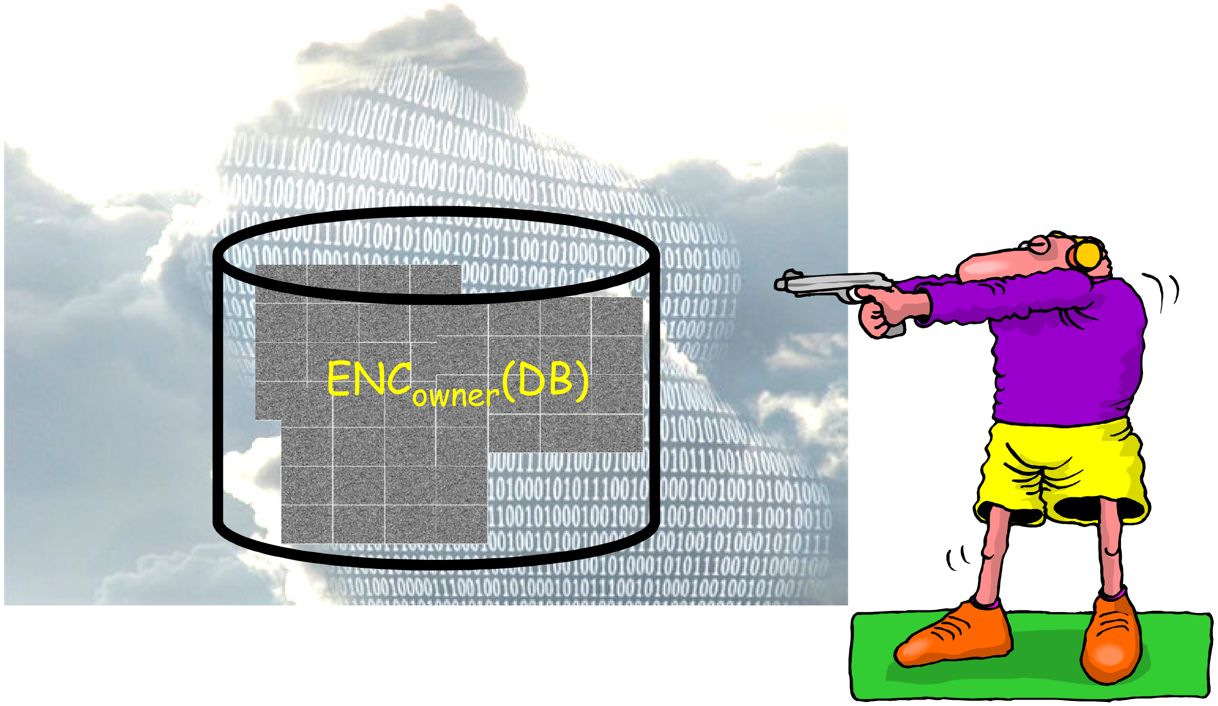
Rutgers University, U.C. Irvine, IBM Research

Searchable Symmetric Encryption (SSE)

- Definition
  - Client encrypts its own data (with its own keys): DB → EDB
  - Outsources EDB to a cloud server, keeps a single cryptographic key K
  - Later, using K only, performs *keyword-based* searches by sending the cloud *encrypted queries,* and receiving back the *encrypted matching records*
- Security goal: *Cloud does not learn plaintext data or queries*
  - Some forms of statistical leakage allowed: data access patterns (e.g. repeated retrieval, size info), query patterns (e.g., repeated queries), etc.
    - Plaintext data/queries *never directly exposed,* but statistical inference possible
    - Security argued on the basis of formal leakage profiles and well defined adv's
- Application: outsourced private data repositories (email, file system, backup, database, …)

With SSE…

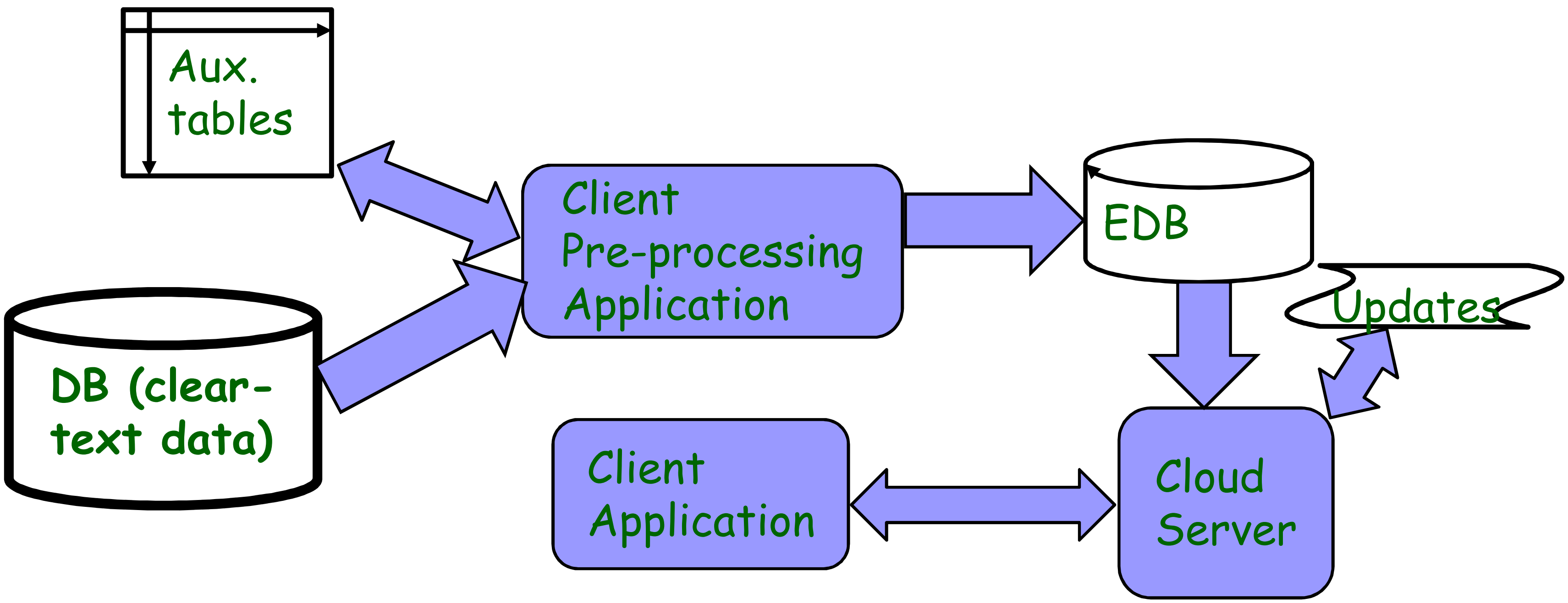
The cloud cannot disclose your data… ***not even at gun point!***



Practical Goals and Trade-offs

- Tens of Billions of distinct (keyword, recId) pairs
  - DB: Relational tables or document collections
  - EDB Workload dominated by searches
    - Encrypted Search Performance should be comparable to Clear-Text Search
  - DB->EDB (pre-processing) done periodically
- Moderate Hardware requirements (set by the funding entity)
  - 4-12 CPU cores, ~100GB RAM, ~10TB additional storage
- Tradeoff: **extensive pre-processing to speed-up encrypted searches**
  - Updates separated from the encrypted database
  - Pre-process to integrate updates or to limit leakage

Prototype



- Carefully designed to scale beyond RAM
  - Big challenge: Security implies maximal randomization yet efficiency calls for maximal "sequentialization" in disk and DB access!!
- Code: 65+k lines of C, lex/yacc & Perl

Quantifying Leakage

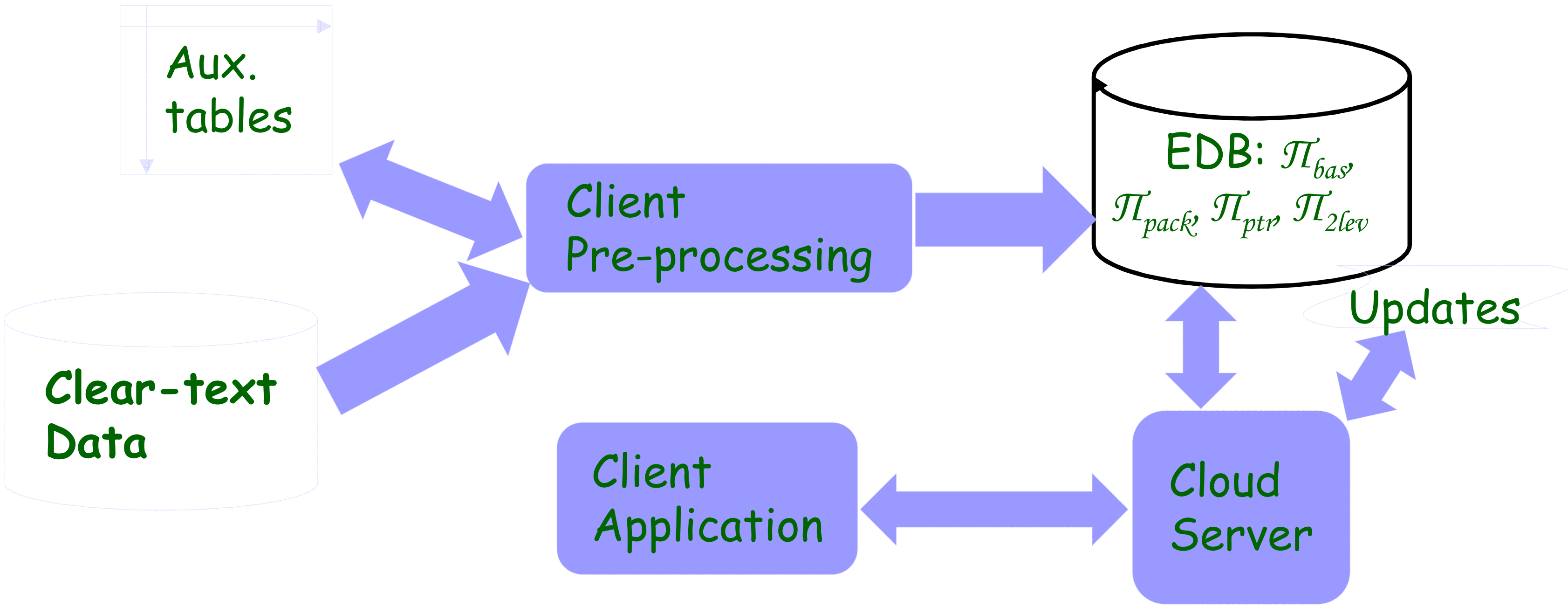
- Static SSE Scheme for Single Keyword Search (SKS)
  - Setup(DB): Encrypting clear-text data (pre-processing)
  - Search(w): Querying encrypted data
- Static SSE Schemes: Π_{bas}, Π_{pack}, Π_{ptr}, Π_{2lev}
  - Leakage functions $\mathcal{L}_{bas}$, $\mathcal{L}_{pack}$, $\mathcal{L}_{ptr}$, $\mathcal{L}_{2lev}$
- Each scheme Π is proven $\mathcal{L}$-secure against adaptive attacks!!

Quantifying Leakage contd.

- Dynamic SSE Scheme for Single Keyword Search (SKS)
  - Setup(DB): Encrypting clear-text data (pre-processing)
  - Search(w): Querying encrypted data
  - Update: Inserting, Deleting, Modifying records
- Dynamic schemes Π^{+}, Π^{dyn} and leakage functions $\mathcal{L}^{+}$, $\mathcal{L}^{dyn}$
- Each scheme Π is proven $\mathcal{L}$-secure against adaptive attacks!!

- Integrated formal protocol and system design!
  - Π_{pack} and Π_{2lev} implemented and evaluated!

EDB Data Structures



Big challenge: **Security implies maximal randomization yet efficiency calls for maximal "sequentialization" in disk and DB access!**

Π_{pack} [Crypto 2013] 100x larger datasets than previous work

Π_{2lev} [NDSS 2014] another 100x over Π_{pack}

EDB as SKS dictionary: $(\mathrm{Enc}_{K_1}(w), \mathrm{EDB}(w))\ \forall\, w$, where $\mathrm{EDB}(w)=\{\mathrm{Enc}_{K_2}(\mathrm{Id})\mid w \in \mathrm{record}_{\mathrm{Id}}\}$

Complex Functional Settings

- Multi-Client SKS SSE: data owner shares its cloud data with friends
  - EDB_{MC}: $(Enc_{K_1}(w), EDB_{MC}(w))$, $\forall w$ where

    $EDB_{MC}(w) = \{Enc_{K_2}(Id, RDK_{Id}) \mid w \in record_{Id}\}$ [CCS 2013]
- Multi-Client, Conjunctive Search (OXT) in SSE setting
  - EDB_{OXT}: $(Enc_{K_1}(w), EDB_{OXT}(w))$, $\forall w$ where

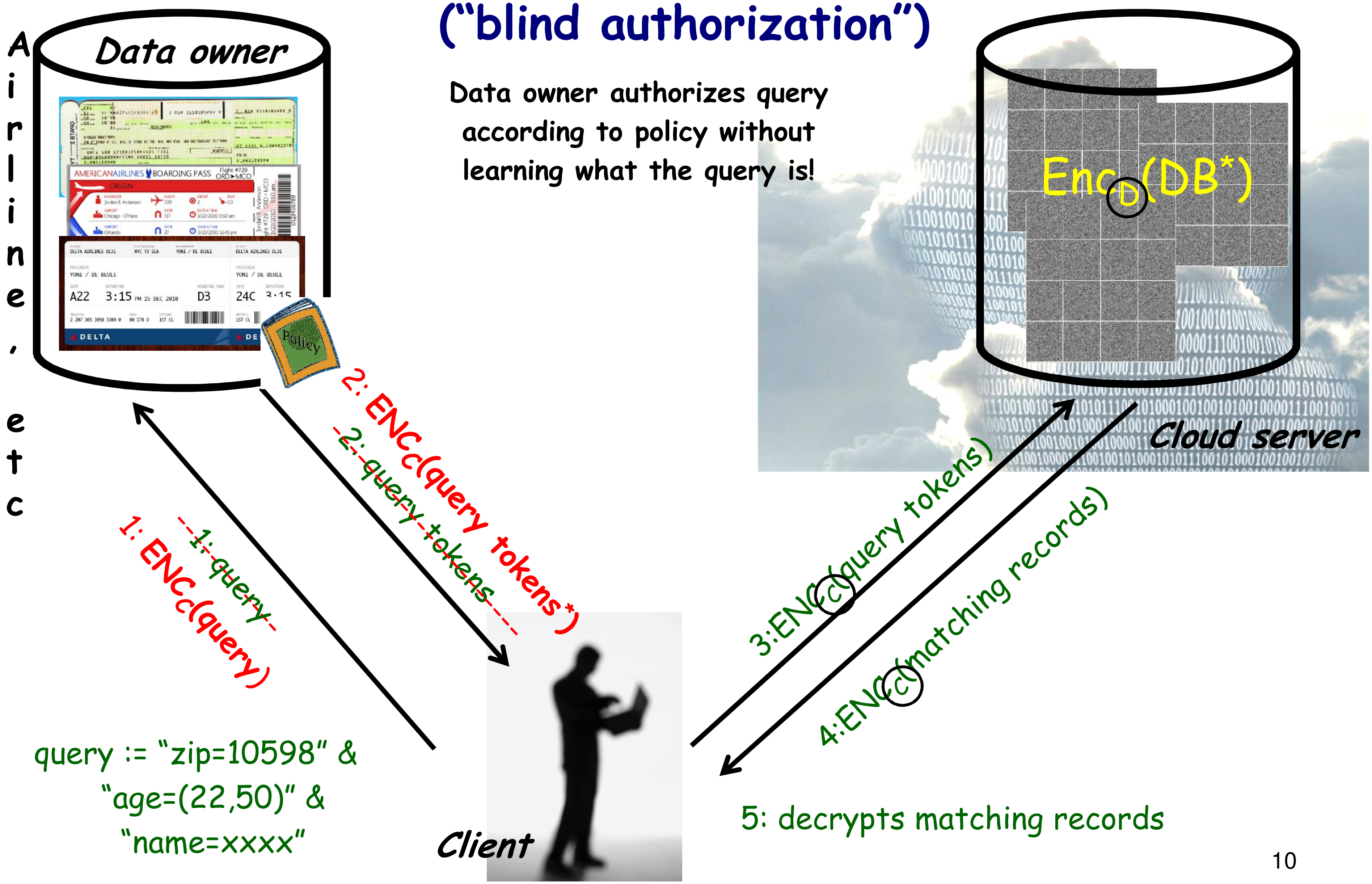
    $EDB_{OXT}(w) = \{Enc_{K_2}(Id, RDK_{Id}, \text{'xind'}, \text{'y'}) \mid w \in record_{Id}\}$,

    'xind' and 'y' are required for conjunctive queries [Crypto 2013]
- Outsourced Symmetric PIR: data owner authorizes clients to perform queries (policy)… without learning the search terms she authorizes
  - Data owner is malicious but she does not collude with Cloud server
  - 'Data owner' – 'Cloud server' separation crucial to avoid PIR cost.

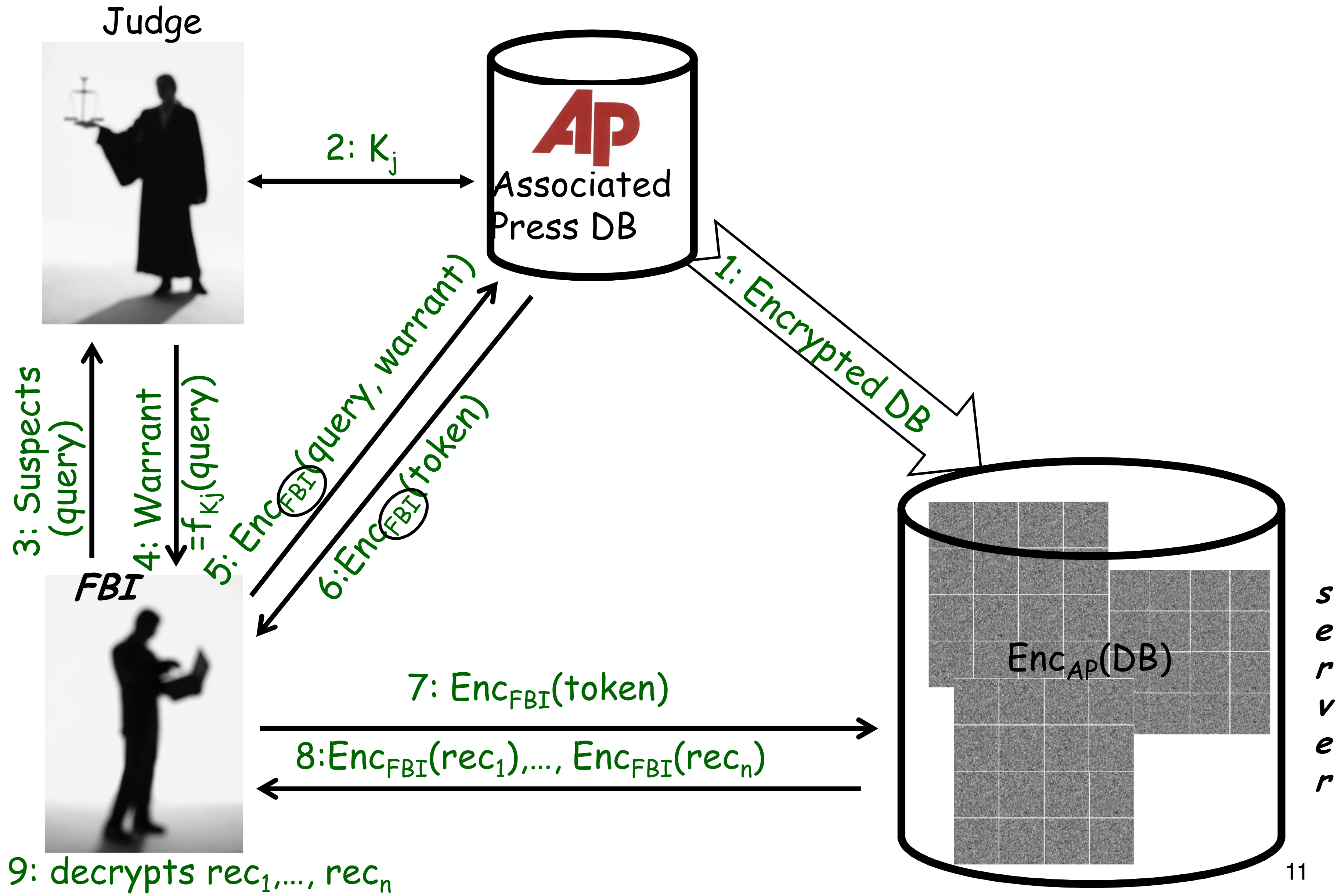
Outsourced Symmetric PIR Setting

("blind authorization")

Data owner authorizes query according to policy without learning what the query is!

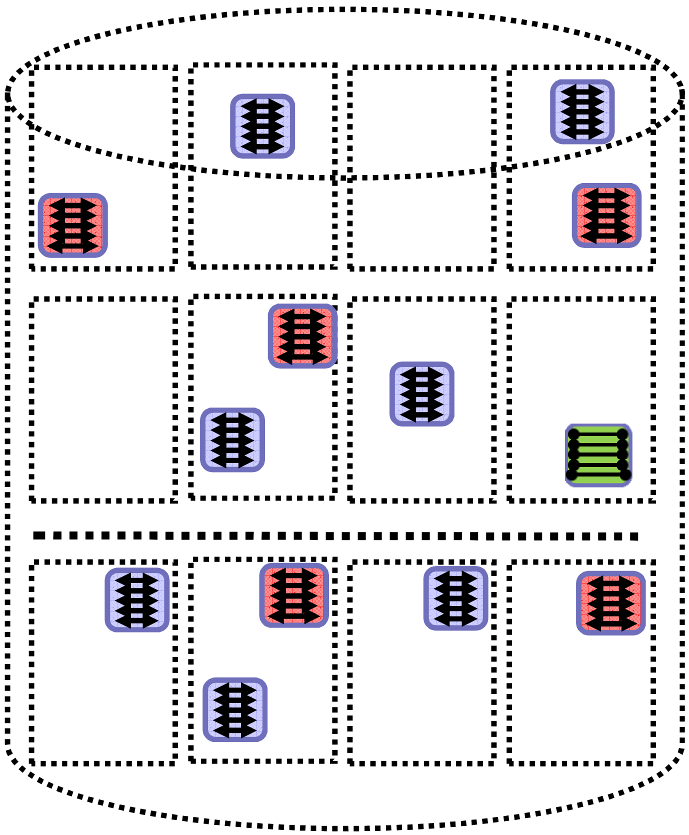


OSPIR w/ Warrant-based Authorization



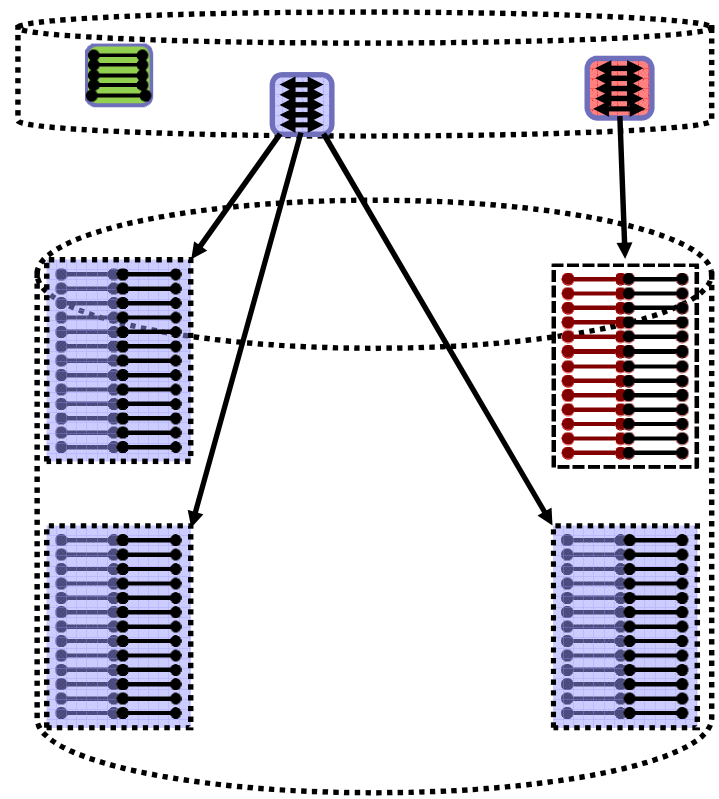
Faster Pre-Processing and Better Goodput

Π_{pack}: Bucket (Paged) Hash (PH)



- Low storage utilization (~60%)
- Cuckoo Hash fix (~90% util): sensitive to insertion history
- Low goodput

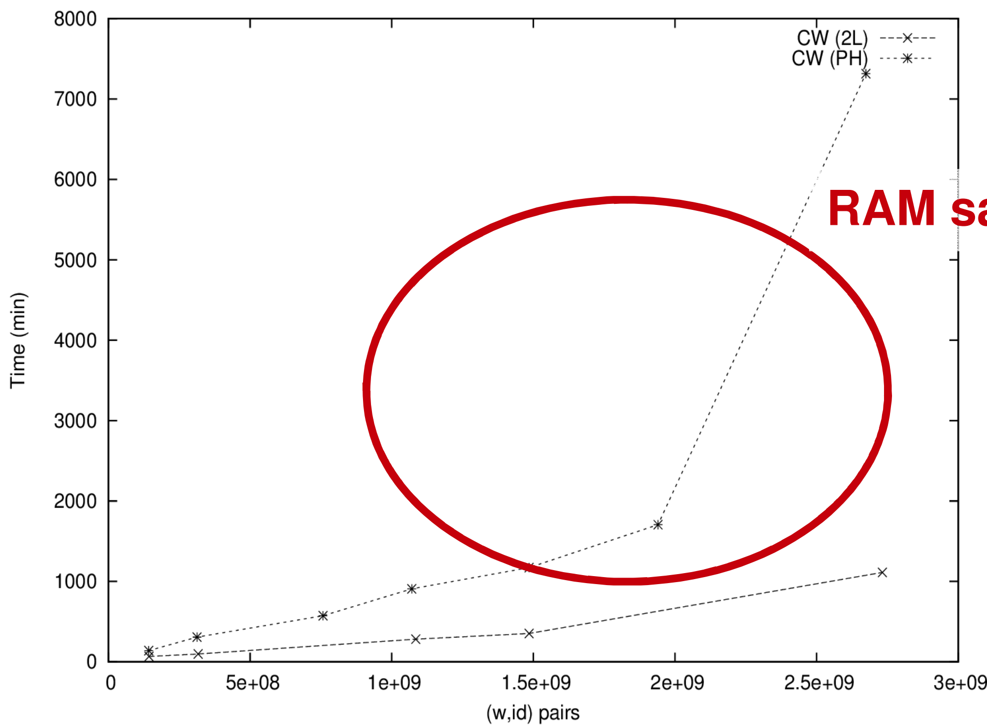
Π_{2lev}: Two Levels (2L)



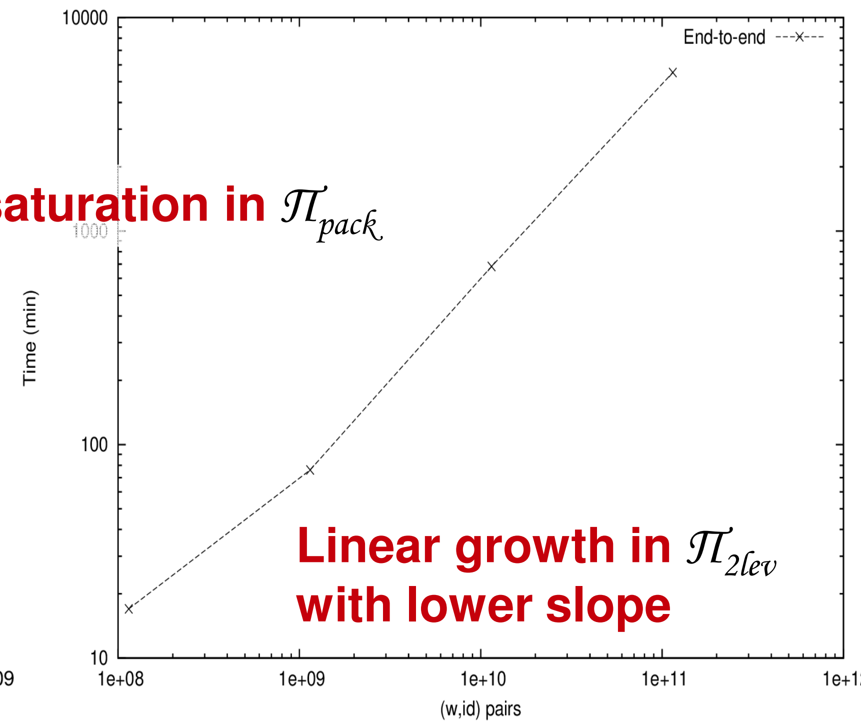
- Multi-modal keyword distribution
- Good storage utilization (92%)
- High goodput.

Pre-processing Scalability

π_{2lev} vs. π_{pack} on ClueWeb (OXT)



π_{2lev} on Census Data (SKS)

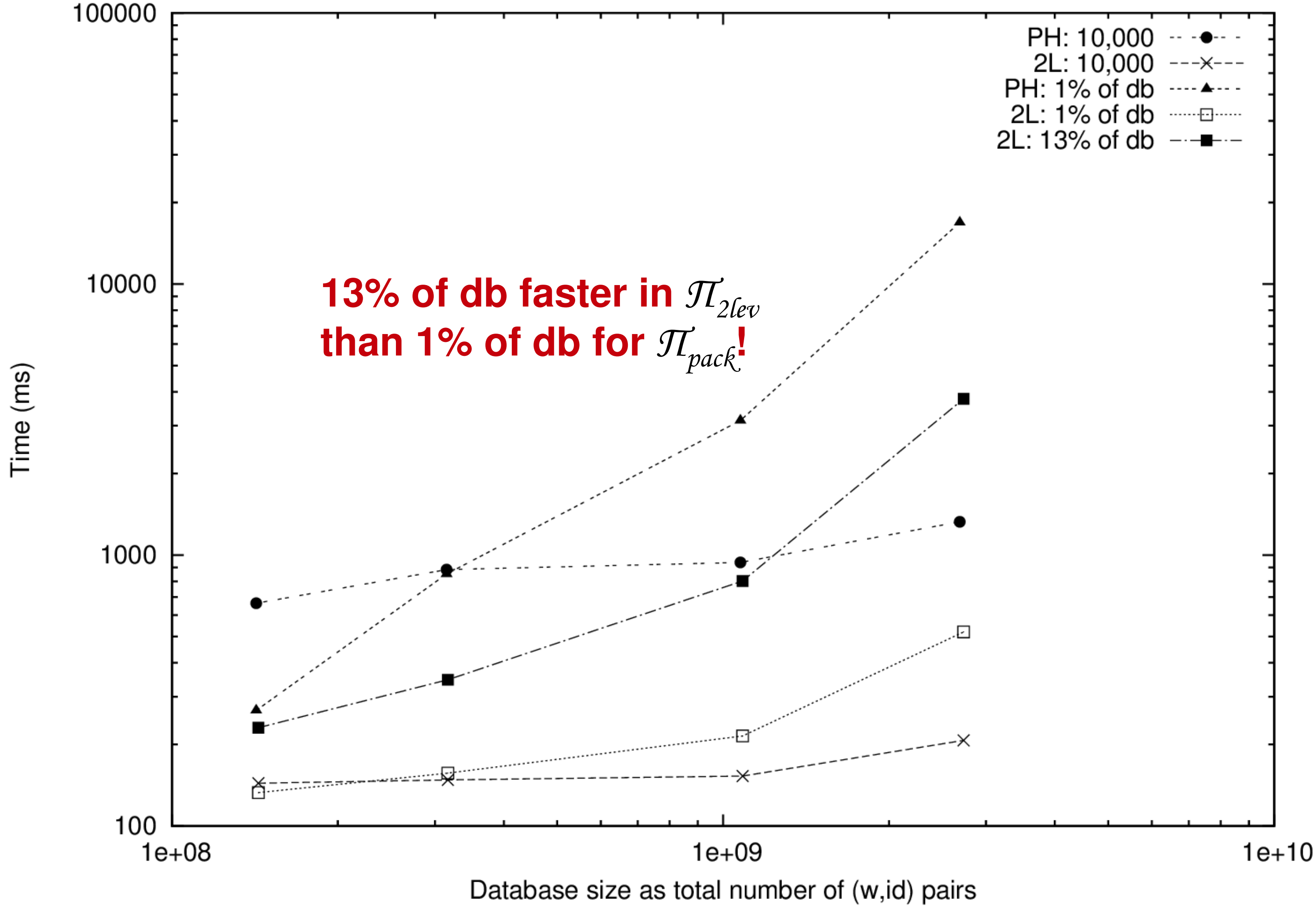


RAM saturation in π_{pack}

Linear growth in π_{2lev} with lower slope

ClueWeb: Subsets of ClueWeb09 data set, crawled web-pages including wikipedia.
Census Data: Lincoln Lab's database.

SKS Query Scalability: $\mathcal{T}_{2lev}$ vs. $\mathcal{T}_{pack}$



13% of db faster in $\mathcal{T}_{2lev}$ than 1% of db for $\mathcal{T}_{pack}$!

Acknowledgements and Future Work

- Supported by the Intelligence Advanced Research Projects Activity (IARPA) under the Security and Privacy Assurance Research (SPAR) program

- In the News: **President Obama** announcing plans for moving Telephone Data away from NSA (speech Jan 17th 2014)

The ***review group recommended that our current approach be replaced by one in which the providers or a third party retain the bulk records, with government accessing information as needed. Both of these options pose difficult problems. […]*** During the review process, some suggested that we may also be ***able to preserve the capabilities we need through a combination of existing authorities, better information sharing, and recent technological advances. But more work needs to be done to determine exactly how this system might work.***

Thanks!