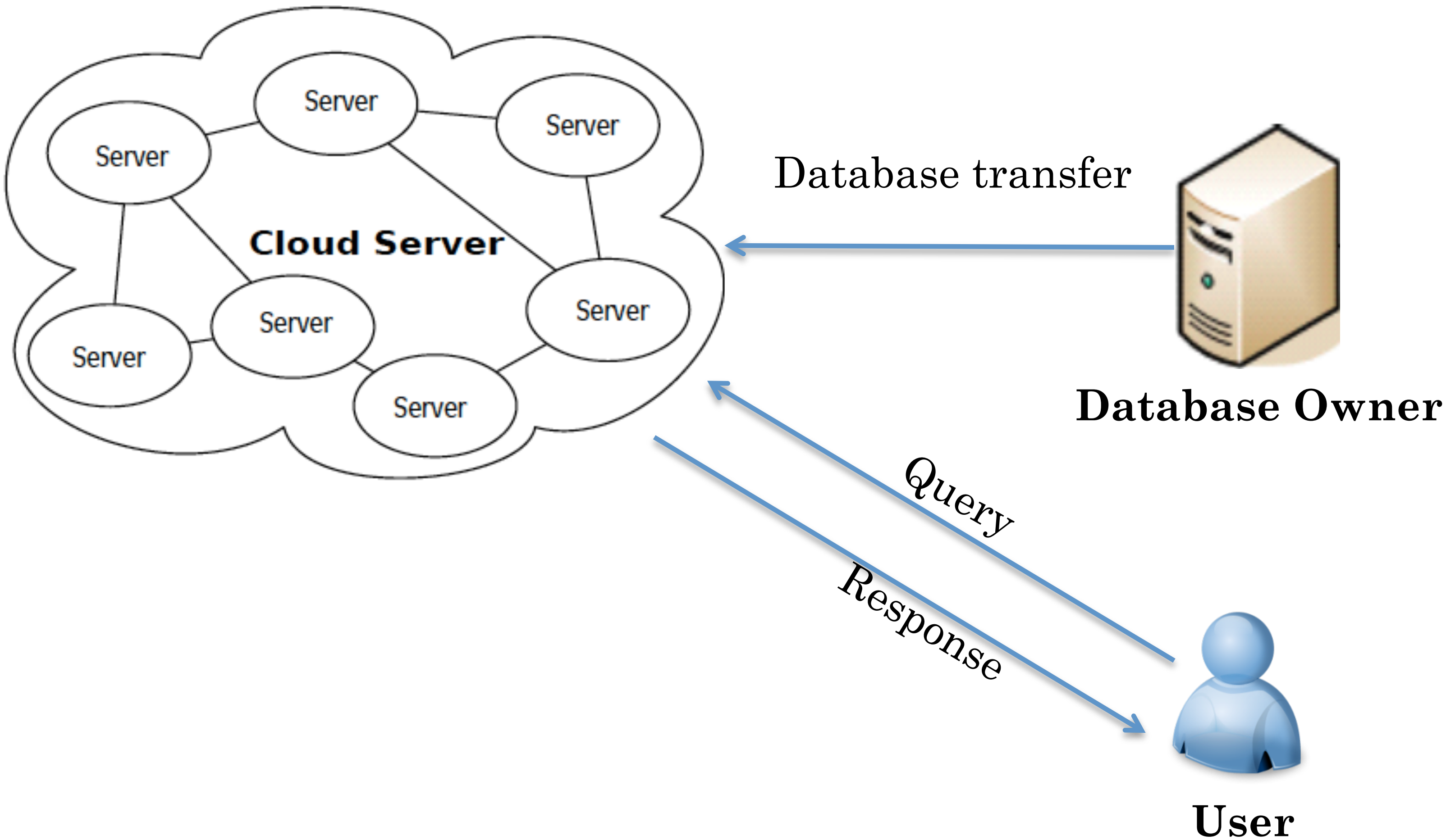
Privacy-Preserving Logarithmic-time Search on Encrypted Data in Cloud

Yanbin Lu
University of California, Irvine
(NDSS'11, Feb 6)

Cloud database environment

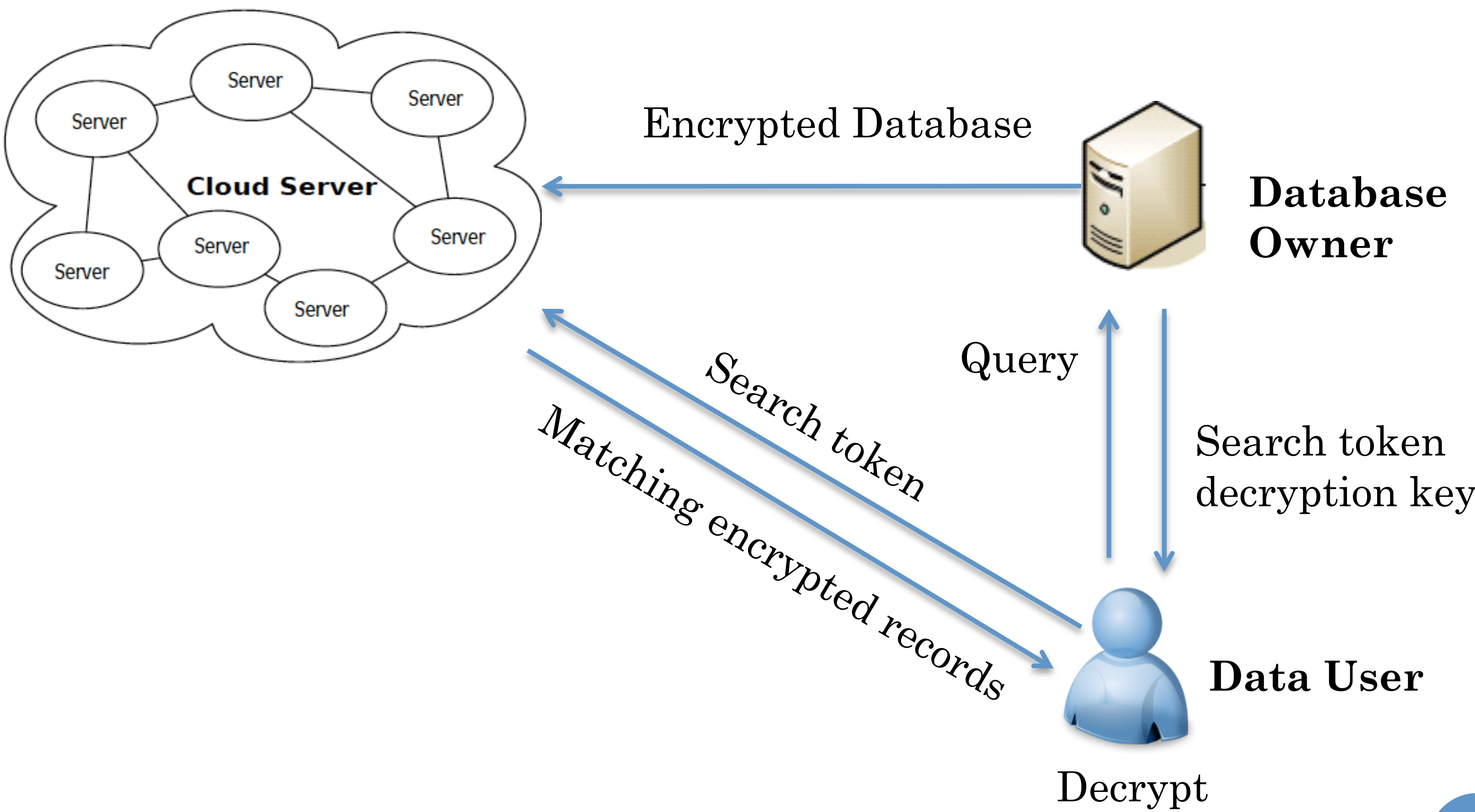


PRIVACY REQUIREMENTS

- Privacy requirements:
  - Cloud server learns no information about database
  - Cloud server learns no information about user query
  - Owner can exercise access control over user query

- Personal data vault example:
  - Owner: Patient
  - Database: Heart beat rate
  - Cloud server: Amazon RDS
  - User: Cardiologist

PRIVACY-PRESERVING SOLUTION



REQUIREMENTS

- Sublinear search
  - Linear search is untolerable in massive data
- Query result integrity
  - Prevent cloud server from cheating user
- Provable database update
  - Prevent cloud server from cheating database owner

Related work

- Order preserving encryption
  - Deterministic and not IND-CPA secure
  - Domain distribution is fixed
- Bellare et al. [crypto'07]
  - Deterministic and not IND-CPA secure
  - Only equality search is supported
- Predicate encryption
  - Useful in privacy-preserving cloud database
  - Linear complexity

PREDICATE ENCRYPTION

- Setup(1^k): output secret key SK.
- Encrypt(SK, I, m): encrypt message m under attributes I with key SK.
- Key-extraction(g): outputs key k_g
- Decrypt(k_g, C_I): decrypts iff $g(I) = 1$

Building blocks

- Range predicate encryption (RPE)
  - Ciphertext associated with point t
  - Decryption key associated with a range Q
  - Decryption works if $t \in Q$

- Inner-product predicate encryption (IPE)
  - Ciphertext associated with vector $\vec{x}$
  - Decryption key associated with vector $\vec{v}$
  - Decryption works if $\langle \vec{v}, \vec{x} \rangle = 0$

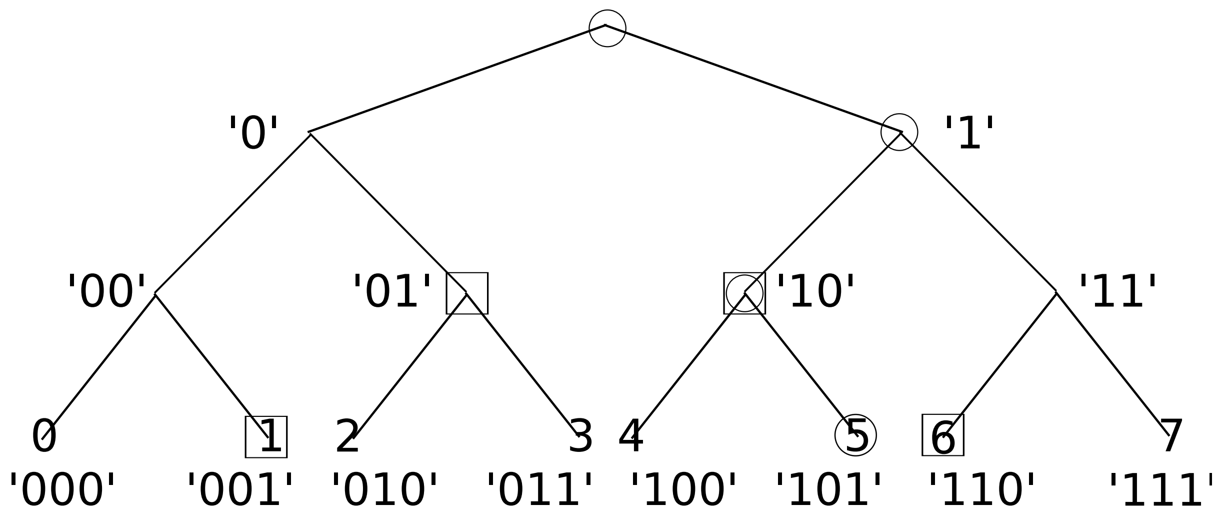
Strawman RPE building from IPE

- Encrypt(t): create $\vec{x} = (x_1, \ldots, x_i, \ldots, x_T)$ where
  $x_i = 1$ if $i = t$ and $x_i = 0$ otherwise.
  Run IPE encryption.

- Extract(Q): create $\vec{y} = (y_1, \ldots, y_i, \ldots, y_T)$ where
  $y_i = 0$ if $i \in Q$ and $y_1 = 1$ otherwise.
  Run IPE key extraction.

- Decrypt(e_t, k_Q): Run IPE decryption.

Efficient range representation



- Any range can be covered by $2 \cdot (\log T - 1)$ nodes.
- Point path intersects with range representation

EFFICIENT RANGE PREDICATE ENCRYPTION

- Encrypting point t:
  - $$P(X) = \prod_{v \in \mathcal{CP}(t)} (X - v) = \sum_{i=0}^{\log T} \alpha_i X^i$$
  - $\vec{A} = (\alpha_0, \ldots, \alpha_{\log T})$
- Key extraction for range Q:
  - $\vec{K}_x = (x^0, \ldots, x^{\log T}), \forall x \in \mathcal{MCS}(Q)$
- Observation:
  - $\vec{A} \cdot \vec{K}_x = \alpha_0 \cdot x^0 + \alpha_1 \cdot x^1 + \cdots + \alpha_{\log T} \cdot x^{\log T} = P(x)$

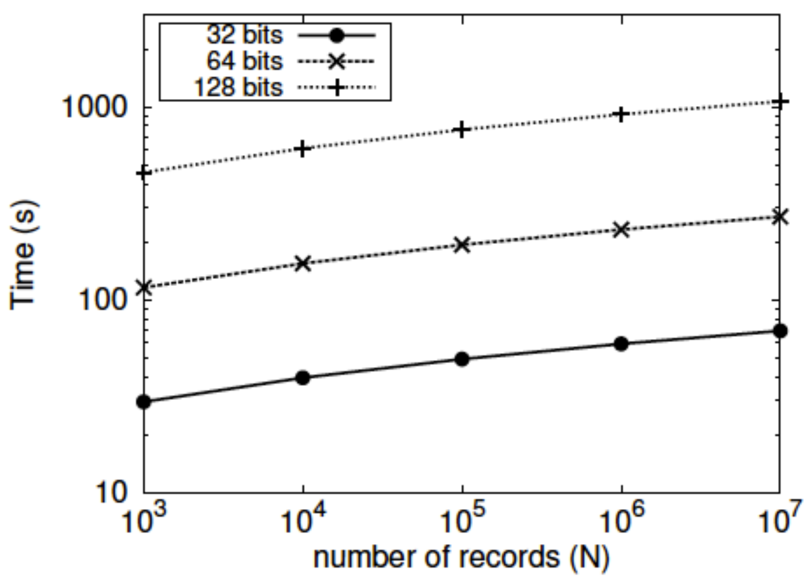
LOGARITHMIC-TIME SEARCH

- Encrypting each node of B-tree
  - One RPE for search token
  - One RPE for real message
- Search token extraction involves two rounds
  - One for left range
  - One for right range
  - Example:
    - Domain size [0-100]
    - Query range [10-20]
    - Left range [0-9], right range [21-100]

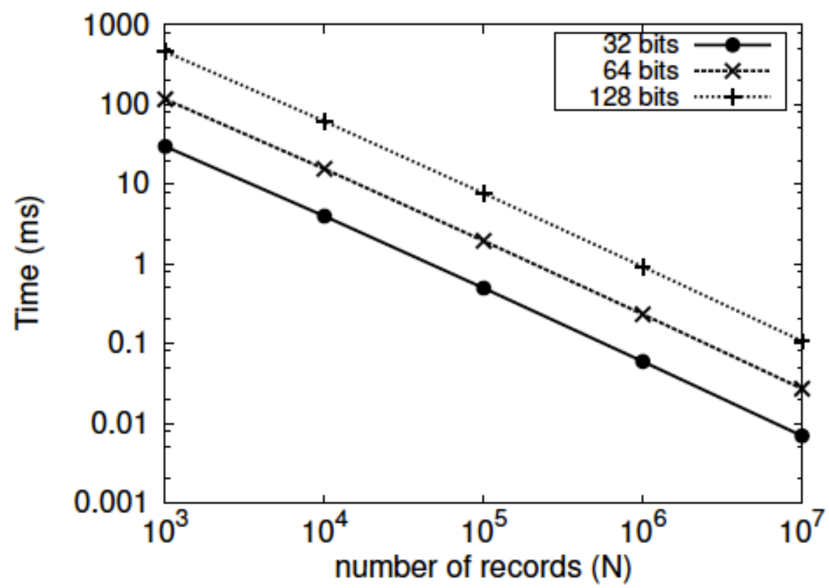
Query authentication

- Authenticated data structure
  - Encrypted B-tree
  - Authenticated root
- Query result verification
  - Left and right boundary to query range
  - Verification without leaking records out of range
- Provable data update
  - Owner first verifies change path
  - Reconstructs and authenticates root

PERFORMANCE



Total search time



Search time per record

Thank you!