vABS: Towards Verifiable Attribute-Based Search over Shared Cloud Data

Yang Ji[†1], Cheng Xu[†2], Jianliang Xu[†3], Haibo Hu[‡4]

[†]*Department of Computer Science, Hong Kong Baptist University, Hong Kong*
[‡]*Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong*
{[1]yangji, [2]chengxu, [3]xujl}@comp.hkbu.edu.hk, [4]haibo.hu@polyu.edu.hk

Problem Statement

- **Verifiable Attribute-Based Search over Shared Cloud Data**
  - Cloud data engines provide information search services on behalf of data owner.
  - The correctness of search results cannot be guaranteed if the SP tampers with data records deliberately.
  - Users might be curious about inaccessible data, which motivates to protect data access against unauthorized users.
- **Threat Model**
  - Users need to ensure the integrity of query results from the following two perspectives:
    - **Soundness:** No records in results are tampered with and are truly the results with respect to their own roles.
    - **Completeness:** All records not in results are either non-results or inaccessible to users.
  - Data are cryptographically enforced with fine-grained access control.
  - **Data content** and **access policy** are protected in an **zero-knowledge** manner.

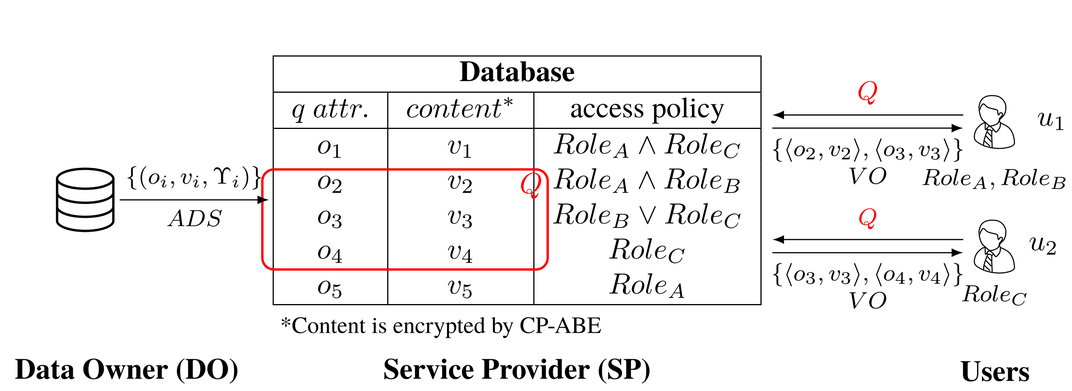


Fig. 1: Verifiable Attribute-based Search

vABS Architecture

- Verifiable attribute-based search services over shared cloud data.
- Client side: **attribute-based search** and **result verification**.
- Server side: **query processing** and **VO construction**.

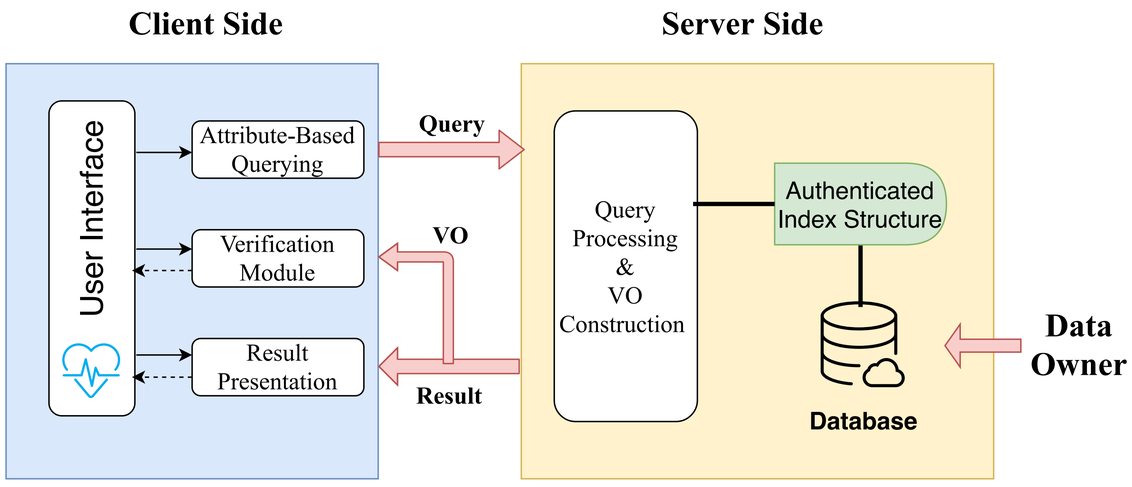


Fig. 2: System Architecture of vABS

Verifiable Equality Search

- **Handle Non-existent Data**
  - Introduce a global pseudo access role $Role_{\varnothing}$, which is not possessed by any user.
  - Treat non-existent data records as the data records that cannot be accessed by any user.
  - Therefore, a data record is either accessible or inaccessible to the query user.
- **ADS Generation and Query Processing**

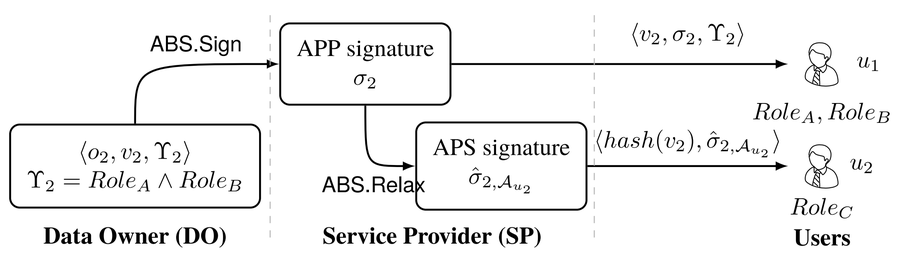


Fig. 3: Verifiable Equality Search

– **APP Signature** proves the authenticity of the accessible data record. It captures query attribute o_i, data content v_i, access policy Υ_i and is signed by the data owner for ADS generation.

$$\sigma_i = \mathsf{ABS.Sign}(sk_{\mathrm{DO}}, hash(o_i)|hash(v_i), \Upsilon_i)$$

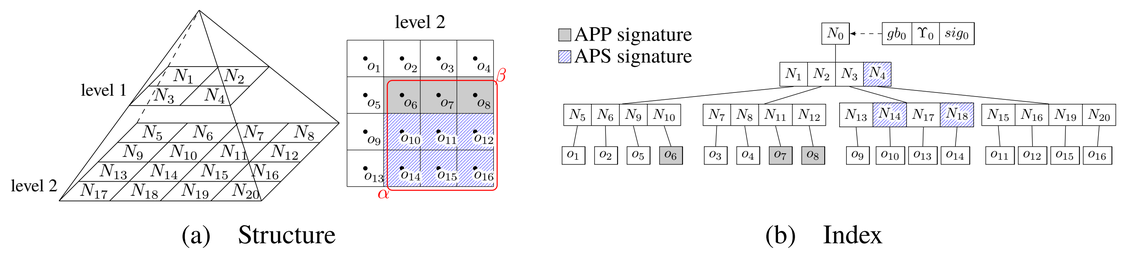
– **APS Signature** proves the authenticity of the inaccessible data record whose query attribute is o_i, to the user whose role set is $\mathcal{A}$. It is derived by the SP from APP signature **without knowing the signing key**.

$$\hat{\sigma}_{i,\mathcal{A}} = \mathsf{ABS.Sign}(sk_{\mathrm{DO}}, hash(o_i)|hash(v_i), \hat{\Upsilon}_{\mathcal{A}})$$
$$\hat{\Upsilon}_{\mathcal{A}} = a_1 \vee a_2 \vee \cdots \vee a_n, \quad a_i \in \mathbb{A}\backslash\mathcal{A}$$

– Query results and VO are encrypted with CP-ABE before sending to the users to prevent impersonation attacks.

Verifiable Range Search

- **Access-Policy-Preserving Grid-Tree**

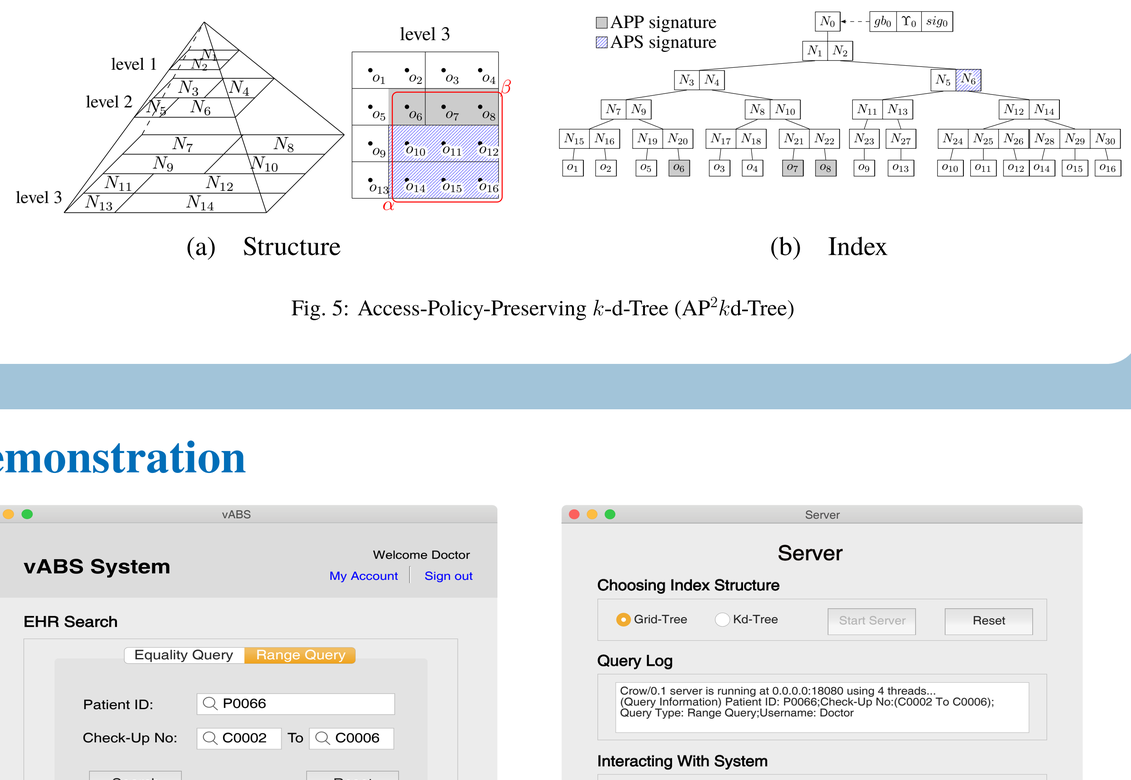


(a) Structure (b) Index

Fig. 4: Access-Policy-Preserving Grid-Tree (AP^{2}G-Tree)

– **Non-Leaf Node**:
  - *Access policy* $p_i = p_{c_1} \vee p_{c_2} \vee \cdots \vee p_{c_C}$
  - *APP signature* $sig_i = \mathsf{ABS.Sign}(sk_{\mathrm{DO}}, gb_i, p_i)$
– **Leaf Node**: *Access policy* and *APP signature* are identical to those of underlying data.

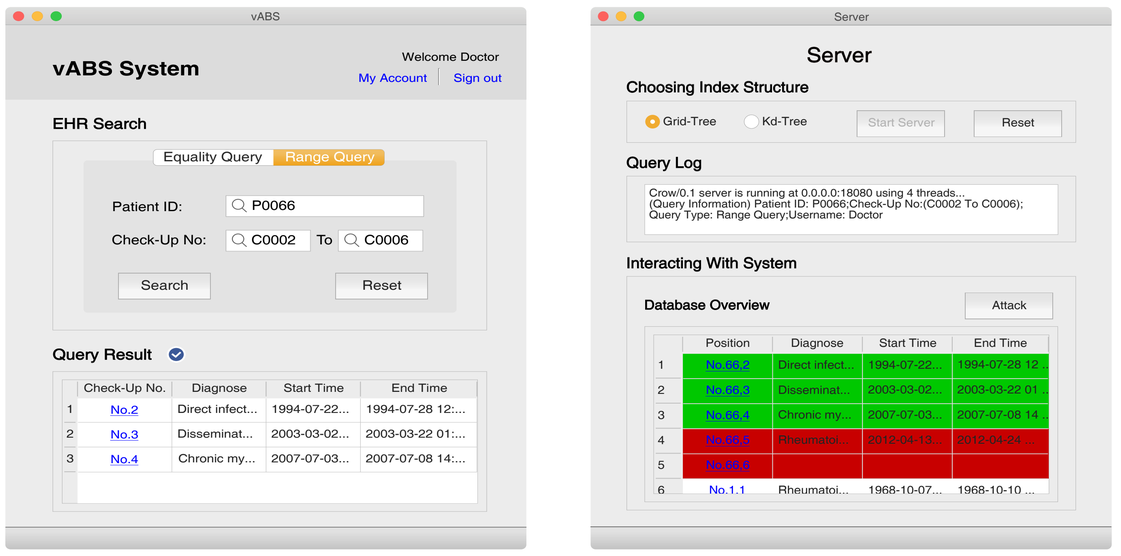
- **Relaxing Zero-Knowledge Requirement**



(a) Structure (b) Index

Fig. 5: Access-Policy-Preserving k-d-Tree (AP^2kd-Tree)

Demonstration



(a) Client Interface (b) Server Interface

Fig. 6: Demonstration System Interface

Reference

[1] C. Xu, J. Xu, H. Hu, and M. H. Au, "When query authentication meets fine-grained access control: A zero-knowledge approach," in *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data*, Houston, TX, USA, Jun. 2018, pp. 147–162.