When Query Authentication Meets Fine-Grained Access Control: A Zero-Knowledge Approach

Cheng Xu[1], Jianliang Xu[1], Haibo Hu[2], and Man Ho Au[2]

[1]Hong Kong Baptist University {chengxu, xujl}@comp.hkbu.edu.hk

[2]Hong Kong Polytechnic University {haibo.hu@, csallen@comp.}polyu.edu.hk

Problem Statement

- **Outsourced Query with Fine-Grained Access Control**
  - Data owner outsources her database to a third-party service provider.
  - Data are cryptographically enforced with **fine-grained** access control.
  - Access policy is presented as **monotone boolean function** over user access roles.
- **Threat Model**
  - Service Provider may be untrusted.
    - ✓ Service Provider returns query results with *verification object* (VO).
    - ✓ User verifies the **soundness** and **completeness** of the results.
  - Users may be curious on inaccessible data.
    - ✓ **Data Content Confidentiality** The content of inaccessible data is protected.
    - ✓ **Access Policy Confidentiality** The access policy of inaccessible data is protected.
    - ✓ **Zero-Knowledge Confidentiality** Any information regarding inaccessible data, including its existence, is protected.
  - Service Provider and users are independent and share no common interest.

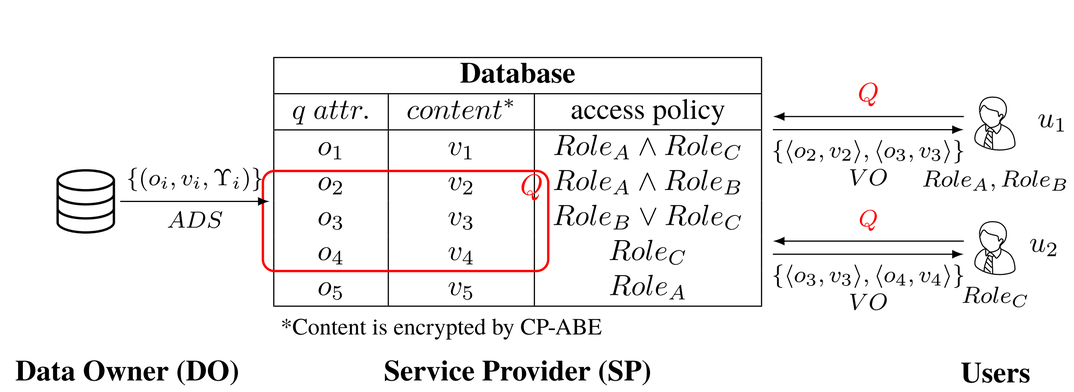


Fig. 1: Query Authentication with Access Control

Preliminaries

- **Ciphertext-Policy Attribute-Based Encryption (CP-ABE)**
  It enables fine-grained access control by embedding the access policy into the ciphertext.
  - CP-ABE.Setup$(1^\lambda) \to (mk, pp)$ Generate master private key and public key.
  - CP-ABE.KeyGen$(mk, \mathcal{A}) \to sk_\mathcal{A}$ Generate decryption key w.r.t. attribute set $\mathcal{A}$.
  - CP-ABE.Encrypt$(pp, x, \Upsilon) \to c_\Upsilon$ Encrypt plaintext x w.r.t. the access policy Υ.
  - CP-ABE.Decrypt$(sk_\mathcal{A}, c_\Upsilon) \to x$ Decrypt ciphertext c_Υ if and only if $\Upsilon(\mathcal{A}) = 1$.
- **Attribute-Based Signature (ABS) with Predicate Relaxation**
  It signs a message with a monotone boolean function predicate that is satisfied by the attributes obtained from the authority.
  - ABS.Setup$(1^\lambda) \to (msk, mvk)$ Generate master signing key and verification key.
  - ABS.KeyGen$(msk, \mathcal{A}) \to sk_\mathcal{A}$ Generate signing key w.r.t. attribute set $\mathcal{A}$.
  - ABS.Sign$(sk_\mathcal{A}, m, \Upsilon) \to \sigma_{m,\Upsilon}$ Sign message m w.r.t. predicate Υ such that $\Upsilon(\mathcal{A}) = 1$.
  - ABS.Verify$(mvk, m, \sigma_{m,\Upsilon}) \to \{0, 1\}$ Verify the signature.
  - ABS.Relax$(\sigma_{m,\Upsilon}, \mathcal{A}') \to \sigma_{m,\Upsilon'}$ Output a new signature w.r.t. predicate $\Upsilon' = \vee_{a\in\mathcal{A}'} a$, if and only if $\Upsilon(\mathbb{A}\backslash\mathcal{A}') = 0$, where $\mathbb{A}$ is the attribute universe.

Equality Query Authentication

- **Handle Non-existent Data**
  - Introduce a global pseudo access role $Role_\emptyset$, 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**

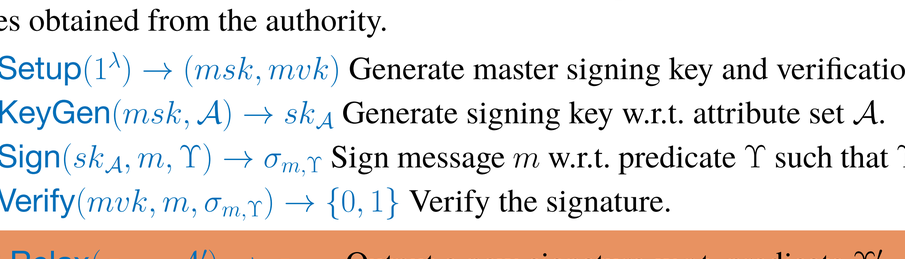


Fig. 2: Equality Query Authentication

- **APP Signature** proves the authenticity of the accessible data record which has query attribute o_i, data content v_i, and access policy Υ_i.

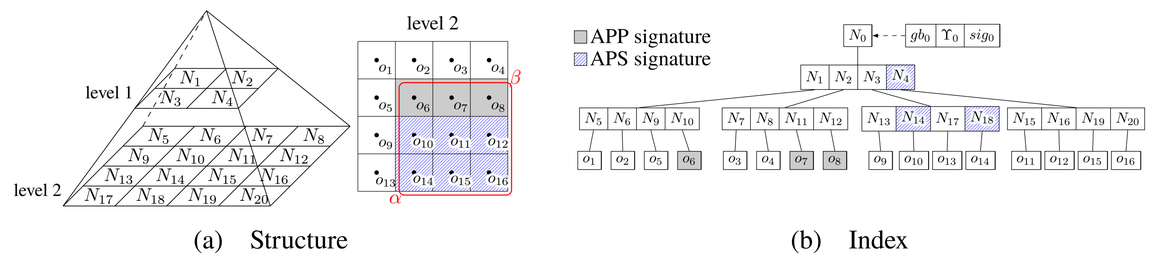
$$\sigma_i = \mathsf{ABS.Sign}(sk_{\mathrm{DO}}, hash(o_i)|hash(v_i), \Upsilon_i)$$

  Generated by data owner.
- **APS Signature** proves the authenticity of the inaccessible data record whose query attribute is o_i, to the user whose role set is $\mathcal{A}$.

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

  where $\mathbb{A}$ is the global access role set.
  Generated by service provider from APP signatures **without knowing the signing key**.
- Query results and VO are encrypted with CP-ABE before sending to the users to prevent impersonation attacks.

Range Query Authentication



(a) Structure (b) Index

Fig. 3: 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.

Join Query Authentication

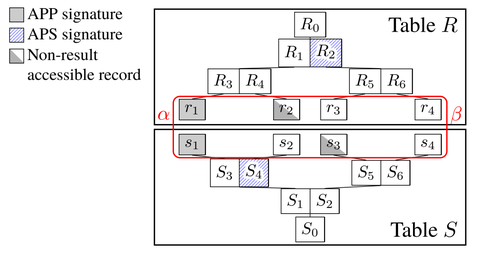
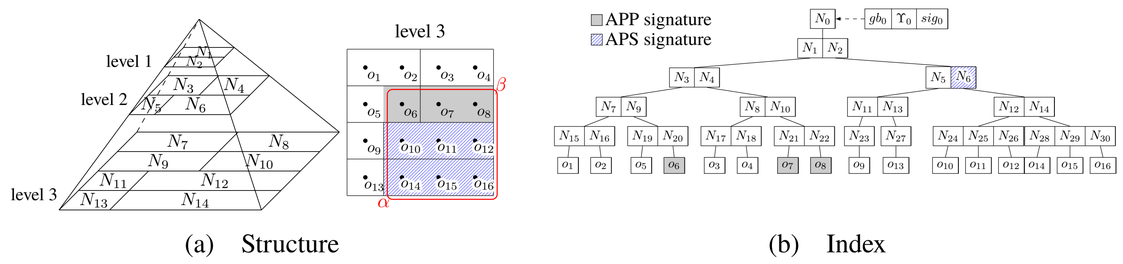


Fig. 4: Equality Join Query Authentication

- Use APP signature to prove soundness.
- Use APS signature to prove completeness.

Optimizations

- **General Optimizations**: (i) Hierarchical Role Assignment. (ii) Parallelism.
- **Relaxing Zero-Knowledge Requirement**:



(a) Structure (b) Index

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

Performance Evaluation