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

June 2018

Background

- *Data-as-a-Service* (DaaS) and cloud computing are gaining popularity for big data analytics

Background

- *Data-as-a-Service* (DaaS) and cloud computing are gaining popularity for big data analytics

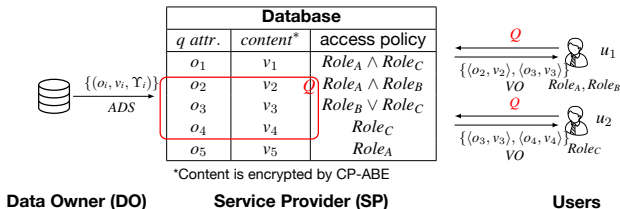**Data Owner (DO)** **Services Provider (SP)** **Users**

- Fine-Grained Access Control: enable big data sharing

Background

- *Data-as-a-Service* (DaaS) and cloud computing are gaining popularity for big data analytics

Data Owner (DO) → Data & ADS → **Services Provider (SP)** ← Queries / Results & VO → **Users**
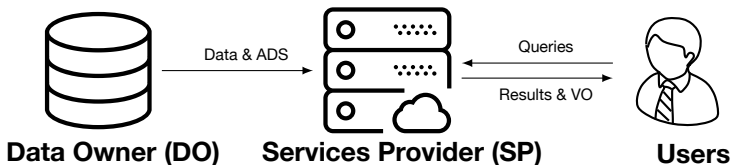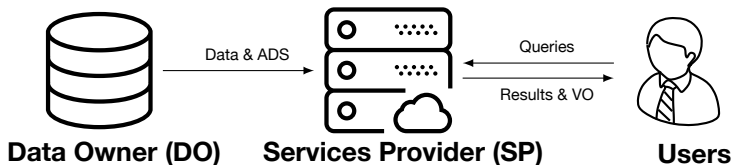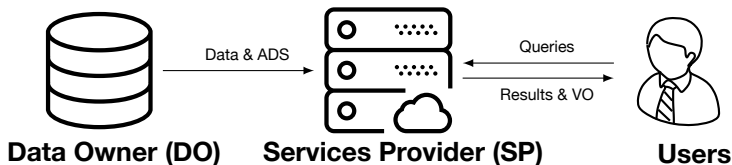
- Fine-Grained Access Control: enable big data sharing
- Security Threats:
    - Query result integrity not guaranteed
   - Leaking information beyond query result may breach privacy

Problem Model

Database		
q attr.	$content^*$	access policy
o_1	v_1	$Role_A \wedge Role_C$
o_2	v_2	$Role_A \wedge Role_B$
o_3	v_3	$Role_B \vee Role_C$
o_4	v_4	$Role_C$
o_5	v_5	$Role_A$

*Content is encrypted by CP-ABE

Data Owner (DO) **Service Provider (SP)** **Users**

$\{(o_i, v_i, \Upsilon_i)\}$
ADS

Q
$\{\langle o_2, v_2 \rangle, \langle o_3, v_3 \rangle\}$
VO $Role_A, Role_B$ u_1

Q
$\{\langle o_3, v_3 \rangle, \langle o_4, v_4 \rangle\}$
VO $Role_C$ u_2

- Fine-grained access policy as monotone boolean function

Problem Model

*Content is encrypted by CP-ABE

Data Owner (DO)   **Service Provider (SP)**   **Users**
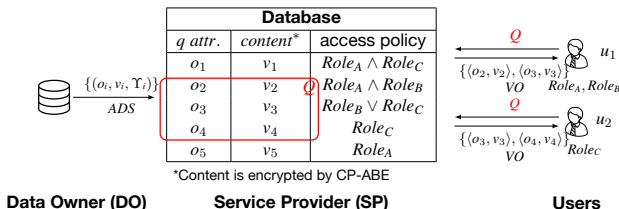
- Fine-grained access policy as monotone boolean function
- Our solution:
  - Integrity: SP returns a *verification object* (VO) to prove
    - Soundness
    - Completeness
  - Zero-Knowledge Confidentiality:
    VO leaks no information beyond query results

Our Contributions

- Develop a novel ABS-based APP signature
  - Authenticate accessible records
  - Prove inaccessibility in zero-knowledge
- Supported query types:
  - Equality queries
  - Range queries
  - Join queries
- Optimization techniques to reduce verification cost
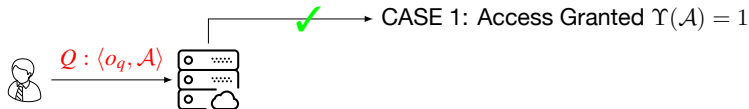
Our Contributions

- Develop a novel ABS-based APP signature
  - Authenticate accessible records
  - Prove inaccessibility in zero-knowledge
- Supported query types:
  - Equality queries
  - Range queries
  - Join queries
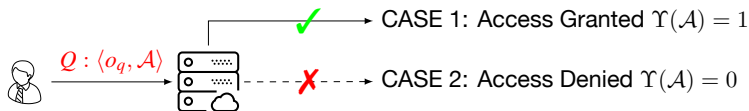- Optimization techniques to reduce verification cost

Equality Query

$Q : \langle o_q, \mathcal{A} \rangle$
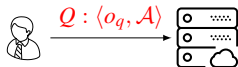
User   **Service Provider**

- User submits a query key o_q and a role set $\mathcal{A}$

Equality Query

CASE 1: Access Granted $\Upsilon(\mathcal{A}) = 1$

$Q : \langle o_q, \mathcal{A} \rangle$

User   **Service Provider**   **Outcomes**

- User submits a query key o_q and a role set $\mathcal{A}$

Equality Query

$Q : \langle o_q, \mathcal{A} \rangle$

✓ → CASE 1: Access Granted $\Upsilon(\mathcal{A}) = 1$

✗ → CASE 2: Access Denied $\Upsilon(\mathcal{A}) = 0$

User   **Service Provider**   **Outcomes**

- User submits a query key o_q and a role set $\mathcal{A}$

Equality Query

CASE 1: Access Granted $\Upsilon(\mathcal{A}) = 1$

CASE 2: Access Denied $\Upsilon(\mathcal{A}) = 0$

CASE 3: No Record ⚠️ Leak information!

User   Service Provider   Outcomes

- User submits a query key o_q and a role set $\mathcal{A}$
- Non-existent record will leak information

Equality Query

CASE 1: Access Granted $\Upsilon(\mathcal{A}) = 1$

$Q : \langle o_q, \mathcal{A} \rangle$

CASE 2: Access Denied $\Upsilon(\mathcal{A}) = 0$

CASE 3: ~~No Record~~ ⚠ — Leak information!

Access Denied $\Upsilon'(\mathcal{A}) = 0$
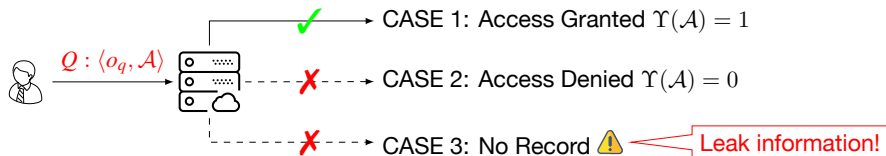
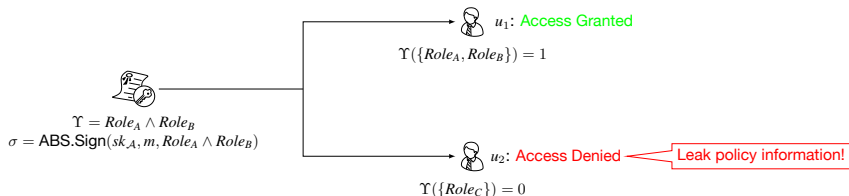**User**   **Service Provider**   **Outcomes**
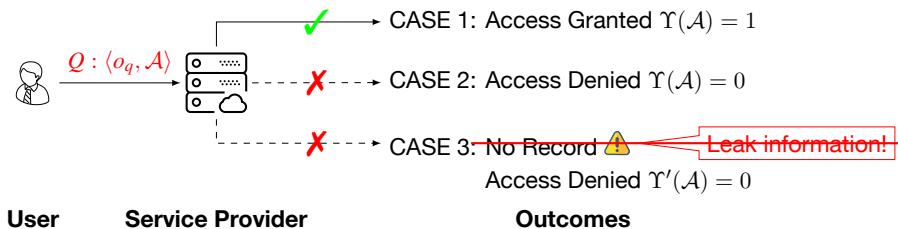
- User submits a query key o_q and a role set $\mathcal{A}$
- ~~Non-existent record will leak information~~
- Treat non-existent records as inaccessible by anyone
  i.e. $\Upsilon' = Role_\emptyset$

ABS with Predicate Relaxation
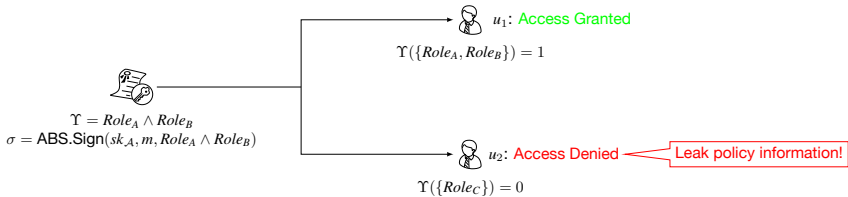
- Attribute Based Signature (ABS)

  It signs a message with a monotone boolean function predicate that is satisfied by the attributes obtained from the authority
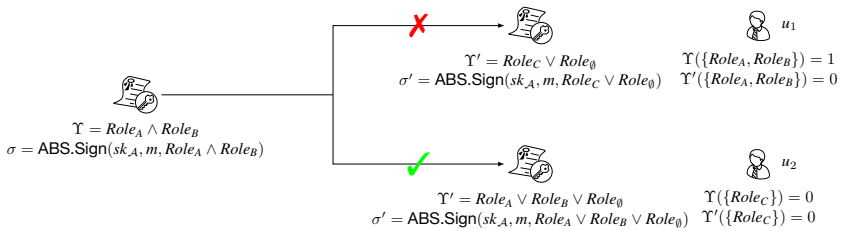
ABS with Predicate Relaxation

- **Attribute Based Signature (ABS)**
  It signs a message with a monotone boolean function predicate that is satisfied by the attributes obtained from the authority

$\Upsilon = Role_A \wedge Role_B$
$\sigma = \text{ABS.Sign}(sk_{\mathcal{A}}, m, Role_A \wedge Role_B)$

u_1: Access Granted
$\Upsilon(\{Role_A, Role_B\}) = 1$

u_2: Access Denied ← Leak policy information!
$\Upsilon(\{Role_C\}) = 0$

- **Predicate Relaxation**
  Derive a weaker ABS signature without knowing secret key

$\Upsilon = Role_A \wedge Role_B$
$\sigma = \text{ABS.Sign}(sk_{\mathcal{A}}, m, Role_A \wedge Role_B)$

✗
$\Upsilon' = Role_C \vee Role_{\emptyset}$
$\sigma' = \text{ABS.Sign}(sk_{\mathcal{A}}, m, Role_C \vee Role_{\emptyset})$

u_1
$\Upsilon(\{Role_A, Role_B\}) = 1$
$\Upsilon'(\{Role_A, Role_B\}) = 0$

✓
$\Upsilon' = Role_A \vee Role_B \vee Role_{\emptyset}$
$\sigma' = \text{ABS.Sign}(sk_{\mathcal{A}}, m, Role_A \vee Role_B \vee Role_{\emptyset})$

u_2
$\Upsilon(\{Role_C\}) = 0$
$\Upsilon'(\{Role_C\}) = 0$

Authenticated Data Structures (ADS)

- Access-Policy-Preserving (APP) signature
  - Signed by DO and used as ADS
  - It captures three parts of information:
   query attribute o_i, data content v_i, and access policy Υ_i

Example 1

$\text{Record}_2 \leftarrow \langle o_2, v_2, \Upsilon_2 = Role_A \wedge Role_B \rangle$

$\sigma_2 \leftarrow \text{ABS.Sign}(sk_{DO}, hash(o_2)|hash(v_2), Role_A \wedge Role_B)$

Authenticated Data Structures (ADS)

- Access-Policy-Preserving (APP) signature
  - Signed by DO and used as ADS
  - It captures three parts of information:
    query attribute o_i, data content v_i, and access policy Υ_i

> **Example 1**
>
> $$\mathsf{Record}_2 \leftarrow \langle o_2, v_2, \Upsilon_2 = Role_A \wedge Role_B \rangle$$
> $$\sigma_2 \leftarrow \mathsf{ABS.Sign}(sk_{DO}, hash(o_2)|hash(v_2), Role_A \wedge Role_B)$$
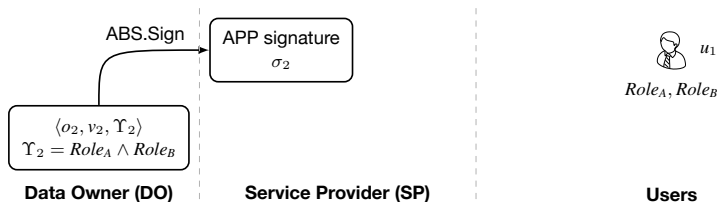
- Access-Policy-Stripped (APS) signature
  - Replace Υ_i to $\hat{\Upsilon}_{\mathcal{A}} = a_1 \vee a_2 \vee \cdots \vee a_n, a_i \in \mathbb{A} \backslash \mathcal{A}$
  - Be used to prove inaccessibility in zero-knowledge

> **Example 2**
>
> $$\mathbb{A} = \{Role_A, Role_B, Role_C, Role_\emptyset\}, \hat{\Upsilon}_{\{Role_C\}} = Role_A \vee Role_B \vee Role_\emptyset$$
> $$\hat{\sigma}_2 \leftarrow \mathsf{ABS.Sign}(sk_{DO}, hash(o_2)|hash(v_2), Role_A \vee Role_B \vee Role_\emptyset)$$
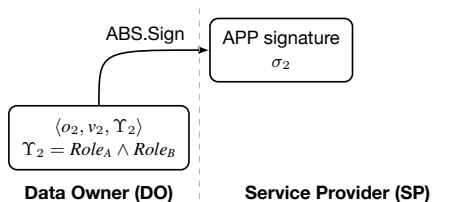
Query Processing

- DO generates ADS and sends to the SP

Query Processing

- DO generates ADS and sends to the SP
- u_1 can access the data,

Query Processing

- DO generates ADS and sends to the SP
- u_1 can access the data, APP signature is the VO

Query Processing

ABS.Sign → APP signature σ_2 → $\langle v_2, \sigma_2, \Upsilon_2 \rangle$ → u_1

$Role_A, Role_B$

$\langle o_2, v_2, \Upsilon_2 \rangle$
$\Upsilon_2 = Role_A \wedge Role_B$

u_2

$Role_C$

Data Owner (DO) · **Service Provider (SP)** · **Users**

- DO generates ADS and sends to the SP
- u_1 can access the data, APP signature is the VO
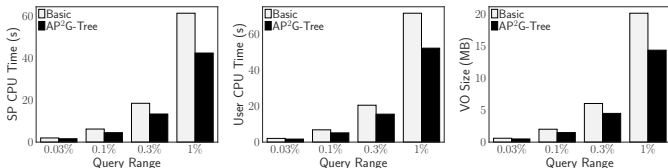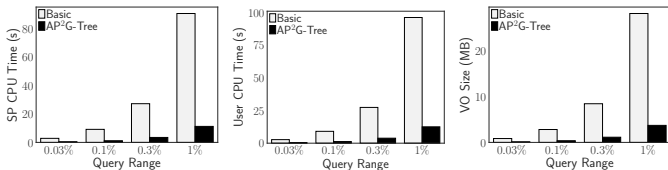- u_2 cannot access the data,

Query Processing
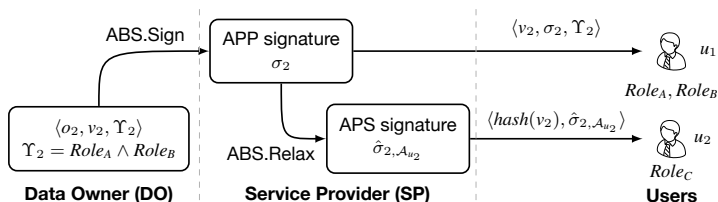
- DO generates ADS and sends to the SP
- u_1 can access the data, APP signature is the VO
- u_2 cannot access the data, SP generates an APS signature as VO

Performance Evaluation

- TPC-H dataset ($1\,800\,000$ records)
- 10 distinct policies (10 global roles, max policy length is 6)

Range Query Performance vs. Range

Join Query Performance vs. Range

Thanks

Q&A