Query-private DB Query Processing Technique

김명선

정보보호학과@수원대학교

1차워크샵@암호연구회, May 20, 2016

Contents

1. System model
2. Private query
3. Problem statement
4. Idea sketch
5. On-going works

Contents

1. System model
2. Private query
3. Problem statement
4. Idea sketch
5. On-going works

Contents

1. System model
2. Private query
3. Problem statement
4. Idea sketch
5. On-going works

Contents

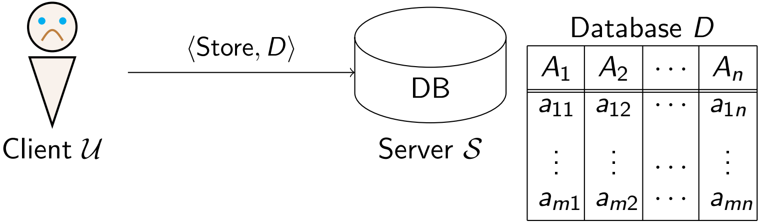
1. System model
2. Private query
3. Problem statement
4. Idea sketch
5. On-going works

Contents

1. System model
2. Private query
3. Problem statement
4. Idea sketch
5. On-going works

System model

- **Outsourced** Database: Naïve model
  - ∗ Client $\mathcal{U}$: DB D를 Server $\mathcal{S}$에 저장
  - ∗ Privacy issue: S learns all tuples in D

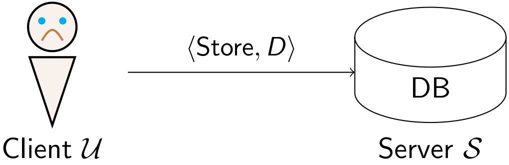


Database D

A_1	A_2	$\cdots$	A_n
a_{11}	a_{12}	$\cdots$	a_{1n}
$\vdots$	$\vdots$	$\cdots$	$\vdots$
a_{m1}	a_{m2}	$\cdots$	a_{mn}

System model

- **Outsourced** Database: Naïve model
  - $*$ Client $\mathcal{U}$: DB D를 Server $\mathcal{S}$에 저장
  - $*$ Privacy issue: S learns all tuples in D

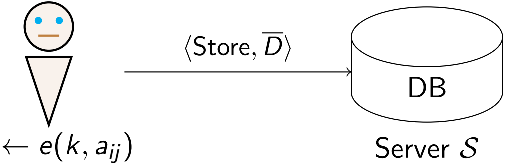


Database D

A_1	A_2	$\cdots$	A_n
a_{11}	a_{12}	$\cdots$	a_{1n}
$\vdots$	$\vdots$	$\cdots$	$\vdots$
a_{m1}	a_{m2}	$\cdots$	a_{mn}

System model

- **Outsourced** Database: Somewhat private···
  - ∗ Client $\mathcal{U}$: 암호화한 DB $\overline{D}$를 Server $\mathcal{S}$에 저장
  - ∗ Technical issue: How to get $\alpha = \sum_{i=1}^{m} a_{ij}$?



$\overline{a}_{ij} \leftarrow e(k, a_{ij})$

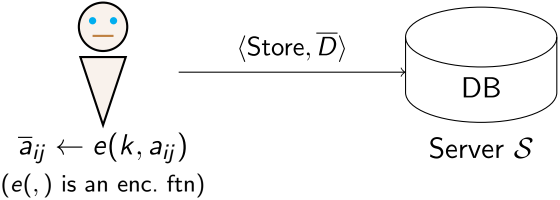
($e(,)$ is an enc. ftn)

Database $\overline{D}$

A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

System model

- **Outsourced** Database: Somewhat private$\cdots$
  - $*$ Client $\mathcal{U}$: 암호화한 DB $\overline{D}$를 Server $\mathcal{S}$에 저장
  - $*$ Technical issue: How to get $\alpha = \sum_{i=1}^{m} a_{ij}$?

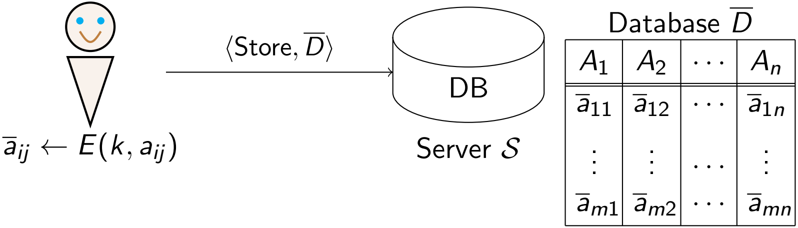

$\langle \text{Store}, \overline{D} \rangle$
DB
Server $\mathcal{S}$
$\overline{a}_{ij} \leftarrow e(k, a_{ij})$
$(e(,)$ is an enc. ftn$)$

Database $\overline{D}$

A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

System model

- **Outsourced** Database: Better private!
  - $*$ Client $\mathcal{U}$: Homo Enc $E()$로 암호화한 DB $\overline{D}$를 Server $\mathcal{S}$에 저장
  - $*$ $\mathcal{S}$ computes $\overline{\alpha} = \sum_{i=1}^{m} \overline{a}_{ij}$
  - $*$ $\mathcal{U}$ learns α by decrypting $\overline{\alpha}$

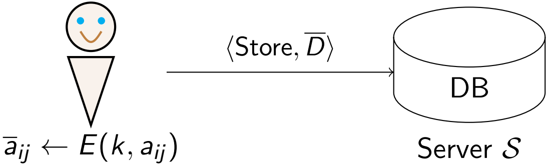

$\overline{a}_{ij} \leftarrow E(k, a_{ij})$
$\langle \text{Store}, \overline{D} \rangle$
DB
Server $\mathcal{S}$

Database $\overline{D}$

A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

System model

- **Outsourced** Database: Better private!
  - $*$ Client $\mathcal{U}$: Homo Enc $E()$로 암호화한 DB $\overline{D}$를 Server $\mathcal{S}$에 저장
  - $*$ $\mathcal{S}$ computes $\overline{\alpha} = \sum_{i=1}^{m} \overline{a}_{ij}$
  - $*$ $\mathcal{U}$ learns α by decrypting $\overline{\alpha}$

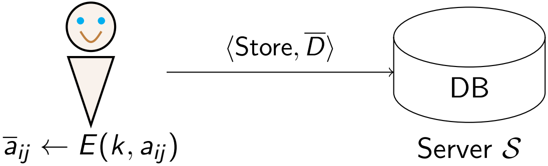

$\langle$Store, $\overline{D}\rangle$
DB
Server $\mathcal{S}$
$\overline{a}_{ij} \leftarrow E(k, a_{ij})$

Database $\overline{D}$

A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

System model

- **Outsourced** Database: Better private!
  - ∗ Client $\mathcal{U}$: Homo Enc $E()$로 암호화한 DB $\overline{D}$를 Server $\mathcal{S}$에 저장
  - ∗ $\mathcal{S}$ computes $\overline{\alpha} = \sum_{i=1}^{m} \overline{a}_{ij}$
  - ∗ $\mathcal{U}$ learns α by decrypting $\overline{\alpha}$

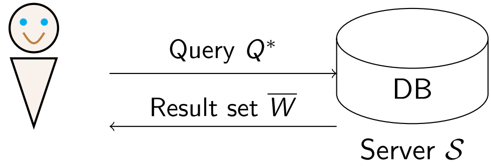

$\langle \text{Store}, \overline{D} \rangle$
DB
Server $\mathcal{S}$
$\overline{a}_{ij} \leftarrow E(k, a_{ij})$

Database $\overline{D}$

A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

Private Query

- How to query over $\overline{D}$?



Database $\overline{D}$

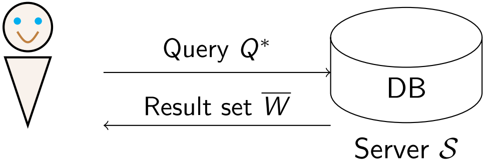
A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\ldots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

$Q^* \leftarrow$ Compiler$(k, D, \ldots)$

- How to work the **Compiler**?
  1. Q ::= SELECT attribute_list FROM Relation WHERE select_condition;
  2. Encrypt all constants in select_condition
  3. Q^* ::= SELECT attribute_list FROM Relation WHERE $\overline{\text{select\_condition}}$;
  4. Run Q^* over $\overline{D}$

Private Query

- How to query over $\overline{D}$?



Database $\overline{D}$

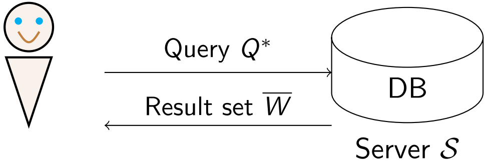
A_1	A_2	$\cdots$	A_n
$\bar{a}_{11}$	$\bar{a}_{12}$	$\cdots$	$\bar{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\bar{a}_{m1}$	$\bar{a}_{m2}$	$\cdots$	$\bar{a}_{mn}$

$Q^* \leftarrow \text{Compiler}(k, D, \ldots)$

- How to work the **Compiler**?
  1. Q ::= SELECT attribute_list FROM Relation WHERE select_condition;
  2. Encrypt all constants in select_condition
  3. Q^* ::= SELECT attribute_list FROM Relation WHERE $\overline{\text{select\_condition}}$;
  4. Run Q^* over $\overline{D}$

Private Query

- How to query over $\overline{D}$?



Database $\overline{D}$

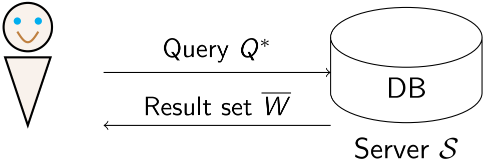
A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

$Q^* \leftarrow \text{Compiler}(k, D, \ldots)$

- How to work the **Compiler**?
  1. Q ::= `SELECT` attribute_list `FROM` Relation `WHERE` select_condition;
  2. Encrypt all constants in select_condition
  3. Q^* ::= `SELECT` attribute_list `FROM` Relation `WHERE` $\overline{\text{select\_condition}}$;
  4. Run Q^* over $\overline{D}$

Private Query

- How to query over $\overline{D}$?



$Q^* \leftarrow \text{Compiler}(k, D, \ldots)$

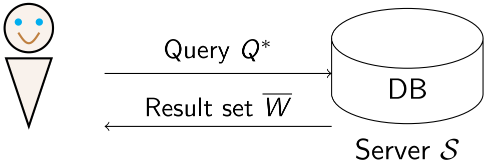
Database $\overline{D}$

A_1	A_2	$\cdots$	A_n
$\bar{a}_{11}$	$\bar{a}_{12}$	$\cdots$	$\bar{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\bar{a}_{m1}$	$\bar{a}_{m2}$	$\cdots$	$\bar{a}_{mn}$

- How to work the **Compiler**?
  1. Q ::= `SELECT` attribute_list `FROM` Relation `WHERE` select_condition;
  2. Encrypt all constants in select_condition
  3. Q^* ::= `SELECT` attribute_list `FROM` Relation `WHERE` $\overline{\text{select\_condition}}$;
  4. Run Q^* over $\overline{D}$

Private Query

- How to query over $\overline{D}$?



Database $\overline{D}$

A_1	A_2	$\cdots$	A_n
$\overline{a}_{11}$	$\overline{a}_{12}$	$\cdots$	$\overline{a}_{1n}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\overline{a}_{m1}$	$\overline{a}_{m2}$	$\cdots$	$\overline{a}_{mn}$

$Q^* \leftarrow \text{Compiler}(k, D, \ldots)$

- How to work the **Compiler**?
  1. Q ::= `SELECT` attribute_list `FROM` Relation `WHERE` select_condition;
  2. Encrypt all constants in select_condition
  3. Q^* ::= `SELECT` attribute_list `FROM` Relation `WHERE` $\overline{\text{select\_condition}}$;
  4. Run Q^* over $\overline{D}$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - An example
    - Q = SELECT Name FROM STUDENT WHERE Grd='A' AND Sex='M';
    - $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - Q^* = SELECT Name FROM STUDENT WHERE Grd=$\overline{a}_1$ AND Sex=$\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - Construct a circuit C_{Q^*} : $\forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - Run the circuit, $\overline{W} = \{\overline{w}_1, \dots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\overline{a}, \overline{b}) := (a = b)\ ?\ \overline{1} :\ \overline{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - Q = SELECT Name FROM STUDENT WHERE Grd='A' AND Sex='M';
    - $\bar{a}_1 \leftarrow E(k, \text{'A'})$ and $\bar{a}_2 \leftarrow E(k, \text{'M'})$
    - Q^* = SELECT Name FROM STUDENT WHERE Grd=$\bar{a}_1$ AND Sex=$\bar{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit C_{Q^*} : $\forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \bar{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \bar{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\bar{a}, \bar{b}) := (a = b) \ ? \ \bar{1} : \ \bar{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - $Q = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd='A' $\texttt{AND}$ Sex='M';
    - $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - $Q^* = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd=$\overline{a}_1$ $\texttt{AND}$ Sex=$\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit C_{Q^*} : $\forall i, \text{Name}[i] \cdot (\mathtt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \mathtt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\mathtt{EQ}(\overline{a}, \overline{b}) := (a = b)\ ?\ \overline{1} :\ \overline{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - $Q = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd='A' $\texttt{AND}$ Sex='M';
    - $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - $Q^* = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd$=\overline{a}_1$ $\texttt{AND}$ Sex$=\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit C_{Q^*} : $\forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\overline{a}, \overline{b}) := (a = b)\ ?\ \overline{1} :\ \overline{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - – Q = SELECT Name FROM STUDENT WHERE Grd='A' AND Sex='M';
    - – $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - – Q^* = SELECT Name FROM STUDENT WHERE Grd=$\overline{a}_1$ AND Sex=$\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit $C_{Q^*} : \forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\overline{a}, \overline{b}) := (a = b) \ ?\ \overline{1} :\ \overline{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - – $Q = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd='A' $\texttt{AND}$ Sex='M';
    - – $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - – $Q^* = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd=$\overline{a}_1$ $\texttt{AND}$ Sex=$\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit C_{Q^*} : $\forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\overline{a}, \overline{b}) := (a = b)\ ?\ \overline{1} :\ \overline{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - $Q = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd='A' $\texttt{AND}$ Sex='M';
    - $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - $Q^* = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd$=\overline{a}_1$ $\texttt{AND}$ Sex$=\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit $C_{Q^*} : \forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\overline{a}, \overline{b}) := (a = b)\ ?\ \overline{1} :\ \overline{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - – $Q = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd='A' $\texttt{AND}$ Sex='M';
    - – $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - – $Q^* = \texttt{SELECT}$ Name $\texttt{FROM}$ STUDENT $\texttt{WHERE}$ Grd$=\overline{a}_1$ $\texttt{AND}$ Sex$=\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit C_{Q^*} : $\forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\overline{a}, \overline{b}) := (a = b) \;?\; \overline{1} : \;\overline{0}$$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - – Q = SELECT Name FROM STUDENT WHERE Grd='A' AND Sex='M';
    - – $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - – Q^* = SELECT Name FROM STUDENT WHERE Grd=$\overline{a}_1$ AND Sex=$\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit $C_{Q^*} : \forall i, \text{Name}[i] \cdot (\mathtt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \mathtt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$\mathtt{EQ}(\overline{a}, \overline{b}) := (a = b)\ ?\ \overline{1} :\ \overline{0}$

Private Query

- Details of $\overline{\text{select\_condition}}$
  - ∗ An example
    - – Q = SELECT Name FROM STUDENT WHERE Grd='A' AND Sex='M';
    - – $\overline{a}_1 \leftarrow E(k, \text{'A'})$ and $\overline{a}_2 \leftarrow E(k, \text{'M'})$
    - – Q^* = SELECT Name FROM STUDENT WHERE Grd=$\overline{a}_1$ AND Sex=$\overline{a}_2$;

- Computation on $\mathcal{S}$'s side
  - ∗ Construct a circuit C_{Q^*} : $\forall i, \text{Name}[i] \cdot (\texttt{EQ}(\text{Grd}[i], \overline{a}_1) \cdot \texttt{EQ}(\text{Sex}[i], \overline{a}_2))$
  - ∗ Run the circuit, $\overline{W} = \{\overline{w}_1, \ldots, \overline{w}_m\} \leftarrow C_{Q^*}(\overline{D})$

- Is everybody happy?

  ∗ ∗ ∗ No!! ∗ ∗ ∗

$$\texttt{EQ}(\overline{a}, \overline{b}) := (a = b) \; ? \; \overline{1} : \; \overline{0}$$

Problem statement

Why "No"?

- $\mathcal{S}$ can learn the logical operators (e.g., `and`/`or`) from Q^*
- Our problem

How to hide the operators from the suspicious server?

An Example

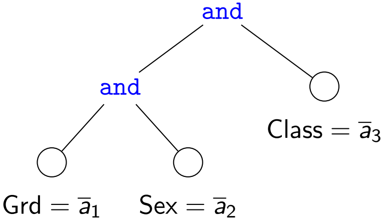


Figure 1: A Query Tree

```
SELECT  Name, Depart, Address
  FROM  STUDENT
 WHERE  Grd = ā1
   AND  Sex = ā2
   AND  Class = ā3;
```

Figure 2: An SQL Statement

Idea sketch

Our goals

1. Hide the select_condition clause
   ⇒ Protect private constants & logical operators
2. Harmonize security and performance
   ⇐ Reveal the `select`-statement & the `from`-statement
   ⇐ SIMD, Automorphism, Dynamic programming, Heuristics

Our assumptions

1. Underlying encryption: (leveled) Fully homomorphic encryption
2. Primitive: `EQ` circuit
   - $\mathrm{depth}(\mathtt{EQ}) = \lceil \log n \rceil$ for two n-bit inputs
3. Conjunctive, Disjunctive and Threshold Conjunctive queries

Our idea

- Express all of target queries as the **same structure** of circuits

Idea sketch

Our goals

1. Hide the select_condition clause
   $\Rightarrow$ Protect private constants & logical operators
2. Harmonize security and performance
   $\Leftarrow$ Reveal the `select`-statement & the `from`-statement
   $\Leftarrow$ SIMD, Automorphism, Dynamic programming, Heuristics

Our assumptions

1. Underlying encryption: (leveled) Fully homomorphic encryption
2. Primitive: `EQ` circuit
   - $\ast$ $\mathrm{depth}(\mathtt{EQ}) = \lceil \log n \rceil$ for two n-bit inputs
3. Conjunctive, Disjunctive and Threshold Conjunctive queries

Our idea

- Express all of target queries as the **same structure** of circuits

Idea sketch

Our goals

1. Hide the select_condition clause

   $\Rightarrow$ Protect private constants & logical operators

2. Harmonize security and performance

   $\Leftarrow$ Reveal the `select`-statement & the `from`-statement

   $\Leftarrow$ SIMD, Automorphism, Dynamic programming, Heuristics

Our assumptions

1. Underlying encryption: (leveled) Fully homomorphic encryption
2. Primitive: `EQ` circuit
   - $\ast$ $\mathrm{depth}(\mathtt{EQ}) = \lceil \log n \rceil$ for two n-bit inputs
3. Conjunctive, Disjunctive and Threshold Conjunctive queries

Our idea

- Express all of target queries as the **same structure** of circuits

Idea sketch

Our goals

1. Hide the select_condition clause
   $\Rightarrow$ Protect private constants & logical operators
2. Harmonize security and performance
   $\Leftarrow$ Reveal the `select`-statement & the `from`-statement
   $\Leftarrow$ SIMD, Automorphism, Dynamic programming, Heuristics

Our assumptions

1. Underlying encryption: (leveled) Fully homomorphic encryption
2. Primitive: `EQ` circuit
   - $\ast$ $\mathrm{depth}(\texttt{EQ}) = \lceil \log n \rceil$ for two n-bit inputs
3. Conjunctive, Disjunctive and Threshold Conjunctive queries

Our idea

- Express all of target queries as the **same structure** of circuits

Idea sketch

Our goals

1. Hide the select_condition clause
   $\Rightarrow$ Protect private constants & logical operators
2. Harmonize security and performance
   $\Leftarrow$ Reveal the `select`-statement & the `from`-statement
   $\Leftarrow$ SIMD, Automorphism, Dynamic programming, Heuristics

Our assumptions

1. Underlying encryption: (leveled) Fully homomorphic encryption
2. Primitive: `EQ` circuit
   - $*$ $\mathrm{depth}(\mathtt{EQ}) = \lceil \log n \rceil$ for two n-bit inputs
3. Conjunctive, Disjunctive and Threshold Conjunctive queries

Our idea

- Express all of target queries as the **same structure** of circuits

Idea sketch

Our goals

1. Hide the select_condition clause
   ⇒ Protect private constants & logical operators
2. Harmonize security and performance
   ⇐ Reveal the `select`-statement & the `from`-statement
   ⇐ SIMD, Automorphism, Dynamic programming, Heuristics

Our assumptions

1. Underlying encryption: (leveled) Fully homomorphic encryption
2. Primitive: `EQ` circuit
   - $\ast$ $\mathrm{depth}(\mathtt{EQ}) = \lceil \log n \rceil$ for two n-bit inputs
3. Conjunctive, Disjunctive and Threshold Conjunctive queries

Our idea

- Express all of target queries as the **same structure** of circuits

Idea sketch

Our goals

1. Hide the select_condition clause
   $\Rightarrow$ Protect private constants & logical operators
2. Harmonize security and performance
   $\Leftarrow$ Reveal the `select`-statement & the `from`-statement
   $\Leftarrow$ SIMD, Automorphism, Dynamic programming, Heuristics

Our assumptions

1. Underlying encryption: (leveled) Fully homomorphic encryption
2. Primitive: `EQ` circuit
   - $*$ $\mathrm{depth}(\texttt{EQ}) = \lceil \log n \rceil$ for two n-bit inputs
3. Conjunctive, Disjunctive and Threshold Conjunctive queries

Our idea

- Express all of target queries as the **same structure** of circuits

Idea sketch

Main observations

1. Using EQ, target queries can be expressed into the same circuit
   - $\mathbb{F}_{2^\ell}$: the plaintext domain
   - $a_i, b, c, d \in \mathbb{F}_{2^\ell}$ and $A_i \in \mathbb{F}_{2^\ell}$
   - Circuit $C^\star$ defined by

$$C^\star = \overline{d} + \prod_{i=1}^{n} \left(\overline{b} + \mathtt{EQ}\left(\overline{A}_i, \overline{a}_i\right) \cdot \overline{c}\right)$$

2. Evaluation table

Query type	b	c	d	Result of $C^\star$
Conjunction	0	1	0	0/1
Disjunction	1	1	1	0/1
Threshold Conjunction	1	$1+t$	0	t^κ

$t \in \mathbb{F}^*_{2^\ell}$ and κ: # of threshold conditions

Idea sketch

Main observations

1. Using EQ, target queries can be expressed into the same circuit
   - $\mathbb{F}_{2^\ell}$: the plaintext domain
   - $a_i, b, c, d \in \mathbb{F}_{2^\ell}$ and $A_i \in \mathbb{F}_{2^\ell}$
   - Circuit $C^\star$ defined by

$$C^\star = \overline{d} + \prod_{i=1}^{n} \left(\overline{b} + \mathtt{EQ}\left(\overline{A}_i, \overline{a}_i\right) \cdot \overline{c}\right)$$

2. Evaluation table

Query type	b	c	d	Result of $C^\star$
Conjunction	0	1	0	0/1
Disjunction	1	1	1	0/1
Threshold Conjunction	1	$1+t$	0	t^κ

$t \in \mathbb{F}^*_{2^\ell}$ and κ: # of threshold conditions

Idea sketch

Main observations–continued

- A little bit problem: the results for threshold queries are still $\overline{t^i}$
  $\Rightarrow$ Modify the results into $\overline{0}$ or $\overline{1}$
- Our technique
  - $\mathcal{S}$ is required to evaluate an encrypted polynomial $\overline{g}(X)$ regardless of query types

$$g(X) = \begin{cases} h(X) & \text{if threshold queries} \\ X & \text{otherwise} \end{cases}$$

where $h(t^\kappa) = 1$ if $\kappa > T$ for a threshold $T \in \mathbb{N}$; $h(t^\kappa) = 0$ otherwise

Idea sketch

Main observations–continued

- A little bit problem: the results for threshold queries are still $\overline{t^i}$
  $\Rightarrow$ Modify the results into $\overline{0}$ or $\overline{1}$
- Our technique
  - $\mathcal{S}$ is required to evaluate an encrypted polynomial $\overline{g}(X)$ regardless of query types

$$g(X) = \begin{cases} h(X) & \text{if threshold queries} \\ X & \text{otherwise} \end{cases}$$

where $h(t^\kappa) = 1$ if $\kappa > T$ for a threshold $T \in \mathbb{N}$; $h(t^\kappa) = 0$ otherwise

On-going works

- Complete our protocol
- Security modeling
  - Security definition
  - Security proof
- Proof-of-concept implementation
  - HElib [HElib] library for the BGV encryption scheme [BGV12], NTL library [NTL]
  - A sample DB generated by a script

On-going works

- Complete our protocol
- Security modeling
  - ∗ Security definition
  - ∗ Security proof
- Proof-of-concept implementation
  - ∗ HElib [HElib] library for the BGV encryption scheme [BGV12], NTL library [NTL]
  - ∗ A sample DB generated by a script

On-going works

- Complete our protocol
- Security modeling
  - ∗ Security definition
  - ∗ Security proof
- Proof-of-concept implementation
  - ∗ HElib [HElib] library for the BGV encryption scheme [BGV12], NTL library [NTL]
  - ∗ A sample DB generated by a script

Wrap-up

Summary

- Project review
- High-level description of our techniques
- Preview of on-going works

Wrap-up

Summary

- Project review
- High-level description of our techniques
- Preview of on-going works

********** Thanks & Question? **********

Wrap-up

Summary

- Project review
- High-level description of our techniques
- Preview of on-going works

********** Thanks & Question? **********

References

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan: *(Leveled) fully homomorphic encryption without bootstrapping*. ITCS2012:309-325.

[HElib] Shai Halevi and Victor Shoup: *Algorithms in HElib*, Eurocrypt 2014.

[NTL] NTL: A Library for doing Number Theory, `http://http://www.shoup.net/ntl/`.