Privacy-Preserving Query Processing over Encrypted Data in Cloud

CS573 Data Privacy and Security

Yousef M. Elmehdwi
Department of Mathematics and Computer Science
Emory University

October 31, 2016
Cloud Computing

Definition

Type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications.

Outsourcing

- Data owner outsources its data as well as processing functionalities to a cloud.
- Reduced management cost, less overhead of data storage, and improved quality of service.

Key Challenge

- Cloud cannot be fully trusted.
- Protect data confidentiality, query privacy, and data access patterns.
How to Ensure Data Confidentiality

- Data owners encrypt their data before outsourced to a cloud
- **Key challenge**: query processing over encrypted data without the cloud ever decrypting the data
Computing on Encrypted Data

Basic idea

- Party $P_1$ sends encrypted data to party $P_2$
- Party $P_2$ performs some computation and returns the encrypted result to party $P_1$
- Party $P_1$ decrypts to find out the answer

Ways to perform computations on encrypted data

- Fully homomorphic encryption (impractical)
- Additive/Multiplicativematic homomorphic encryption schemes (Additive adopted in this work)
The Goal of this Work

- Develop distributed protocols to allow the cloud to perform queries directly over encrypted data.
- During query processing, the cloud cannot infer any information about the outsourced data, the user queries, or data access patterns.
- Such a protocol is termed as privacy-preserving query processing (PPQP).
Desired Output and Security Guarantee

Basic formulation

\[ \text{PPQP}(\langle C : T' \rangle, \langle Bob : q \rangle) \rightarrow \langle Bob : q_{out} \rangle \]

- **Input** - \( T' \) denotes the encrypted database and \( q \) the user query
- **Output** - \( q_{out} \) denotes set of records that satisfies \( q \)

Security requirements

1. Data confidentiality and query privacy
2. Privacy/Hide data access patterns
3. Output security
4. Information that can be inferred from input/output is not a security violation

Other desirable requirements

1. End-user efficiency and correctness
Example: Insurance Company
Two-Cloud Environment

Basic Assumptions

- Existence of two cloud service providers denoted by $C_1$ and $C_2$ (e.g., Google and Amazon)
- Alice owns a database $T$ of $n$ records $t_1, \ldots, t_n$ and $m$ attributes
- Alice generates two keys $(pk, sk)$ based on the AH-ENC system
- Alice encrypts $T$ attribute-wise, and sends the encrypted database $T'$ to $C_1$ and $sk$ to $C_2$
- Bob wants to execute his input query $q = \langle q_1, \ldots, q_m \rangle$ on $T'$ in the cloud in a privacy-preserving manner
- $C_1$ is the data host, who stores all uploaded (encrypted) data $T'$
- $C_2$ is called the key holder since it stores Alice’s private key $sk$
Architecture of Two-Cloud Setting Based Solution

- **Overview**
  - Security model
  - Privacy-preserving primitives
  - Proving Security of SM

- **Motivation**
  - Two-Cloud Setting
  - PP$k$NN Classification

- **Conclusion and Future Research**

---

**Architecture Diagram**

- **Alice**
  - $T$
  - Key

- **Cloud 1 (Data host)**
  - $C_1$
  - $T'$

- **Cloud 2 (Key holder)**
  - $C_2$
  - $sk$

- **Bob**
  - $q'$

- **ANN Classification**

- **Steps**
  1. $T'$
  2. $q'$
  3. Class label ($c_q$)
Adopted Security Model

More realistic model: **Secure multiparty computation (SMC)**

- Parties collaboratively compute the functionality in a secure fashion without using a trusted third party
- In SMC, security means guaranteeing the *correctness* of the output as well as the *privacy* of the parties’ inputs
Adopted Adversarial Model

Adversarial model

Generally specifies what an adversary or attacker is allowed to do during an execution of a secure protocol

<table>
<thead>
<tr>
<th>Common adversary models under SMC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semi-honest</strong>: follow the protocol faithfully, but can try to infer the secret information of the other parties from the data they see during the protocol execution</td>
</tr>
<tr>
<td><strong>Malicious</strong>: may do anything to infer secret information (e.g., input modification, sending the wrong values)</td>
</tr>
<tr>
<td>In our work, we adopt the semi-honest adversary model</td>
</tr>
</tbody>
</table>
Two reasons for adopting Semi-Honest Model

- Developing protocols under the semi-honest setting is an important first step towards constructing protocols with stronger security guarantees.

- Both $C_1$ and $C_2$ were assumed to be two cloud service providers. Today, cloud service providers in the market are legitimate, well-known companies (e.g., Amazon, Google, and Microsoft). These companies maintain reputations that are invaluable assets that need to be protected at all costs. Thus, a collusion between them is highly unlikely as it will damage their reputation, which, in turn, affects their revenues.
Additive Homomorphic Probabilistic Encryption

Let $E_{pk}$ and $D_{sk}$ be the encryption and decryption functions. Given $m_1, m_2 \in \mathbb{Z}_N$, the AH-ENC system exhibits the following properties.

**Homomorphic Addition**

- $D_{sk}(E_{pk}(m_1 + m_2)) = D_{sk}(E_{pk}(m_1) \cdot E_{pk}(m_2))$

**Homomorphic Multiplication**

- Given a constant $c$ and a ciphertext $E_{pk}(m_1)$
- $D_{sk}(E_{pk}(c \cdot m_1)) = D_{sk}(E_{pk}(m_1)^c)$

**Probabilistic**

- Let $c_1 = E_{pk}(m_1)$ and $c_2 = E_{pk}(m_2)$
- Probability for $c_1 \neq c_2$ is very high even if $m_1 = m_2$

**Semantic Security**

- Given $E_{pk}(m_1)$, an adversary cannot derive any information about $m_1$
Secure Multiplication (SM)

\[
SM(\langle C_1 : E_{pk}(a), E_{pk}(b) \rangle, \langle C_2 : sk \rangle) \rightarrow (\langle C_1 : E_{pk}(a \times b) \rangle, \langle C_2 : \emptyset \rangle)
\]

- **Input**: \(E_{pk}(a), E_{pk}(b),\) and private key \(sk\)
- **Output**: encryption of \(a \times b\)

Secure Squared Euclidean Distance (SSED)

\[
SSED(\langle C_1 : E_{pk}(X), E_{pk}(Y) \rangle, \langle C_2 : sk \rangle) \rightarrow (\langle C_1 : E_{pk}(|X - Y|^2) \rangle, \langle C_2 : \emptyset \rangle)
\]

- **Input**: \(X\) and \(Y\) are \(m\)-dimensional vectors, and private key \(sk\), where \(E_{pk}(X) = \langle E_{pk}(x_1), \ldots , E_{pk}(x_m) \rangle, E_{pk}(Y) = \langle E_{pk}(y_1), \ldots , E_{pk}(y_m) \rangle\)
- **Output**: encryption of squared Euclidean distance between \(X\) and \(Y\)
Sub-Protocols 2

Secure Bit-OR (SBOR)

\[
SBOR(\langle C_1 : E_{pk}(o_1), E_{pk}(o_2) \rangle, \langle C_2 : sk \rangle) \rightarrow (\langle C_1 : E_{pk}(o_1 \lor o_2) \rangle, \langle C_2 : \emptyset \rangle)
\]

- **Input**: \(o_1\) and \(o_2\) are two bits, and \(sk\) is private key \(sk\)
- **Output**: encryption of the boolean OR operation between \(o_1\) and \(o_2\)

Secure Bit-Decomposition (SBD)

\[
SBD(\langle C_1 : E_{pk}(z) \rangle, \langle C_2 : sk \rangle) \rightarrow (\langle C_1 : [z] \rangle, \langle C_2 : \emptyset \rangle)
\]

- **Input**: \(E_{pk}(z)\) such that \(0 \leq z < 2^l\) and \(sk\) is private key
- **Output**: \([z] = \langle E_{pk}(z_1), \ldots, E_{pk}(z_l) \rangle\)

SBD: Example

Let \(z = 5\), \(l = 3\). \(SBD(E_{pk}(5), sk) \rightarrow [z] = \langle E_{pk}(1), E_{pk}(0), E_{pk}(1) \rangle\)
New Secure Minimum Sub-Protocol

Secure Minimum (SMIN)

\[ \text{SMIN}(\langle C_1 : (u', v') \rangle, \langle C_2 : sk \rangle) \rightarrow (\langle C_1 : \text{[min}(u, v)], E_{pk}(s_{\text{min}(u,v)}) \rangle, \langle C_2 : \emptyset \rangle) \]

- **Input:** \( u' = ([u], E_{pk}(s_u)), v' = ([v], E_{pk}(s_v)) \), and \( sk \) is private key
  - \([u]\) (resp., \([v]\)) denotes the encryption of individual bits of binary representation of \( u \) (resp., \( v \))
  - \( s_u \) (resp., \( s_u \)) denotes the secret corresponding to \( u \) (resp., \( v \))

- **Output:**
  - \([\text{min}(u, v)]\): encryptions of individual bits of minimum between \( u \) and \( v \)
  - \( E_{pk}(s_{\text{min}(u,v)})\): encryptions of secret corresponds to minimum of \( u \) and \( v \)

SMIN: Example

Let \( u' = ([5], E_{pk}(s_5)) \) and \( v' = ([3], E_{pk}(s_3)) \), where \( [5] = \langle E_{pk}(1), E_{pk}(0), E_{pk}(1) \rangle \), \( [3] = \langle E_{pk}(0), E_{pk}(1), E_{pk}(1) \rangle \) \( \Rightarrow \text{SMIN}(u', v') = ([3], E_{pk}(s_3)) \)
New Secure Minimum out of $n$ numbers Sub-Protocol

Secure Minimum out of $n$ numbers ($\text{SMIN}_n$)

$\text{SMIN}_n(\langle C_1 : (\theta_1, \ldots, \theta_n) \rangle, \langle C_2 : sk \rangle) \rightarrow (\langle C_1 : [d_{\text{min}}], E_{pk}(s_{d_{\text{min}}}) \rangle, \langle C_2 : \emptyset \rangle)$

- **Input:**
  $\forall_{i=1}^{n} \theta_i = ([d_i], E_{pk}(s_{d_i}))$ and $sk$ is private key
  - $[d_i]$: encryption of individual bits of binary representation of $d_i$ for $1 \leq i \leq n$
  - $s_{d_i}$: secret corresponding to $d_i$ for $1 \leq i \leq n$

- **Output:**
  - $[\min(d_1, \ldots, d_n)] = [d_{\text{min}}]$: encryptions of individual bits of global minimum
  - $E_{pk}(s_{\min(d_1, \ldots, d_n)}) = E_{pk}(s_{d_{\text{min}}})$: encryptions of secret corresponds to global minimum

$\text{SMAX}$ and $\text{SMAX}_n$

Similarly, one can design $\text{SMAX}$ and $\text{SMAX}_n$ to compute the global maximum
New Secure Frequency Sub-Protocol

Secure Frequency (SF)

\[ SF(\langle C_1 : (\Lambda, \Lambda') \rangle, \langle C_2 : sk \rangle) \rightarrow (\langle C_1 : E_{pk}(f(c_1)), \ldots, E_{pk}(f(c_w)) \rangle, \langle C_2 : \emptyset \rangle) \]

- **Input:** \( \Lambda = \langle E_{pk}(c_1), \ldots, E_{pk}(c_w) \rangle \), \( \Lambda' = \langle E_{pk}(c'_1), \ldots, E_{pk}(c'_k) \rangle \), and \( sk \) is private key
- **Output:**
  - \( E_{pk}(f(c_j)) \): encryption of the frequency of \( c_j \) in the list \( \langle c'_1, \ldots, c'_k \rangle \), for \( 1 \leq j \leq w \)
  - \( c_j \)'s are unique and \( c'_i \in \{c_1, \ldots, c_w\} \) for \( 1 \leq i \leq k \)

SF: Example

(i.e., \( w = 3 \) and \( k = 6 \)), \( \Lambda = \langle E_{pk}(2), E_{pk}(3), E_{pk}(5) \rangle \) and
\( \Lambda' = \langle E_{pk}(3), E_{pk}(2), E_{pk}(3), E_{pk}(2), E_{pk}(5), E_{pk}(2) \rangle \)
\( \Rightarrow SF(\Lambda, \Lambda') \rightarrow \langle E_{pk}(f(2)) = E_{pk}(3), E_{pk}(f(3)) = E_{pk}(2), E_{pk}(f(5)) = E_{pk}(1) \rangle \)
Key ideas

- Information deduced from the messages in the real execution image of a protocol should be computationally indistinguishable from the information deduced based on the corresponding messages in the simulated view.

- For this, all the intermediate messages seen by an adversary during the execution of a protocol should be either random or pseudo-random.

- To prove a protocol is secure under semi-honest model, it required to show that the execution image of a protocol does not leak any information regarding the private inputs of participating parties. We need to show that the simulated execution image of that protocol is computationally indistinguishable from its actual execution image.

- An execution image generally includes the messages exchanged and the information computed from these messages.