Security Aspects of Database Outsourcing

Vahid Khodabakhshi
Hadi Halvachi

Dec, 2012
Security Aspects of Database Outsourcing
Outline

• Introduction to Database Outsourcing
• Query Correctness
• Secure Database Server
• Security Issues in Querying Encrypted Data
• Encryption Policies for Regulating Access to Outsourced Data
• Proxy Re-Encryption
• Search on Encrypted Data
Database Outsourcing

- Database Outsourcing - *Database As a Service*:
  - The external service provider provides mechanisms for clients to access the outsourced databases.
Database Outsourcing

• Advantages:
  – the high costs of in-house versus outsourced hosting. Outsourcing provides significant cost savings
  – higher availability
  – Fault Tolerance
  – more effective disaster protection

• Challenges:
  – Security:
    • Confidentiality
    • Integrity (=Correctness)
    • Privacy Preserving
DBO Model

- Informal:
  - Data owner
  - Users
  - Service provider (server)
DBO Model

• Formally:
  – Represent client and server as interactive polynomial time Turing Machine.
  – A client can interact with the server and issue a sequence of \((Q_1, \ldots, Q_i)\) that we call \textit{trace} \(T\).
  – After executing a query \(Q\), the client Turing Machine either outputs \(\top\) or \(\bot\), indicating whether the client accepts or rejects the server’s response (denoted as \(D_{T,Q}\)).
  – We write \(CLI(T, Q, D_{T,Q}) \in \{\top, \bot\}\) to denote output of the client as a result of the server’s execution of trace \(T\) and query \(Q\) yielding the result \(D_{T,Q}\).
Query Correctness -1

• A server’s response D is consistent($T, Q$) if an honest server, after starting with an empty database and executing trace $T$ honestly, would reply with $D$ to the query $Q$.

• $T, T'$ is similar with respect to $Q$, ($T \approx_Q T'$) if the query $Q$ yields the same answer when queried after a trace $T$ or $T'$ ($D_{T,Q} = D_{T',Q}$).
Query Correctness - 2

• A query mechanism is correct if the server is bound to the sequence of update requests performed by the client. Either the server responds correctly to a query or its malicious behavior is immediately detected by the client.

• Definition - *Query Correctness*
  
  – A query protocol is correct, if for all traces \( T \) and \( T' \) with \( T \not\equiv_Q T' \), any query \( Q \) and server response \( D_{T',Q} \) we have \( CLI(T, Q, D_{T',Q}) = \bot \).
Secure Database Server

- [Kantarc et al. 2005] shows how results from cryptography prove the impossibility of developing a server that meets cryptographic-style definitions of security and is still efficient enough to be practical.

- So, [Kantarc et al. 2005] propose a definition of a secure database server that provides probabilistic security guarantees.
• **Definition - Indistinguishability of Encryptions**
  
  An encryption scheme with efficient key generation algorithm $G$, efficient encryption algorithm $E$ and decryption algorithm $D$ where $D_k(E_k(x)) = x$ for secret key $k$ has indistinguishable private key encryption if for every polynomial-size circuit family $\{C_n\}$, every polynomial $p$, all sufficiently large $n$ and every $a; b \in 0; 1^{\text{poly}(n)}$ with equal length,

$$|Pr\{C_n(E_{G(1^n)}(a)) = 1\} - Pr\{C_n(E_{G(1^n)}(b)) = 1\}| < \text{negl}(n)$$
Secure Database Server - 2

• Any two pairs of ciphertexts and plaintexts of the same length, it must be infeasible to figure out which ciphertext goes with which plaintext.

• This means that any two database tables with the same schema and the same number of tuples must have indistinguishable encryptions. To be more precise, we now give a database-specific adaptation of the definitions as below:
Secure Database Server - 3

• Definition - Secure Database Server
  – An encryption scheme \((G, E, D)\) for database tables; consisting of key generation scheme \(G\), encryption function \(E\), and decryption function \(D\); has indistinguishable encryptions if for every polynomial-size circuit family \(\{C_n\}\), every polynomial \(p\), and all sufficiently large \(n\), every database \(R_1\) and \(R_2 \in \{0, 1\}^{\text{poly}(n)}\) with the same schema and the same number of tuples (i.e., \(|R_1| = |R_2|\)):

\[
\left| Pr\{C_n(E_{G(1^n)}(R_1)) = 1\} - Pr\{C_n(E_{G(1^n)}(R_2)) = 1\}\right| < \text{negl}(n)
\]
• Definition - *Correctness*
  – Assume database D is stored securely on a server w.r.t previous Definition.
  – Let E(D) be the securely encrypted database and let Q be a query issued on the database.
  – A query execution is said to be correct if given (Q, E(D)), an honest server provides a result enabling the query issuer to learn Q(D).
• Definition - **Privacy**

  – For every query pair Qi, Qj that run on the same set of tables over D and have the same size results, the messages \( m_{Qi} \), \( m_{Qj} \) sent for executing the queries are computationally indistinguishable if for every polynomial-size circuit family \( \{ C_n \} \), every polynomial \( p \), all sufficiently large \( n \), \( m_{Qi} \) and \( m_{Qj} \in \{0, 1\}^{\text{poly}(n)} \),

\[
\left| \Pr[C_n(E_G(1^n)(m_{Qi})) = 1] - \Pr[C_n(E_G(1^n)(m_{Qj})) = 1] \right| < \text{negl}(n)
\]
Security Aspects of Database Outsourcing
Policies

• Authorization Policies → Encryption Policies

• Encryption Policies:
  – which data are encrypted with which key
  – which keys are released to which users
  – Assign one key to each user
  – Encrypting each resource at most once.

• we propose a formal model for representing an authorization policy.

• We also introduce the definition of minimum encryption policy.
Authorization Policies - 1

• Authorization Policy:
  – Let $U$ and $R$ be the set of users and resources in the system. An authorization policy over $U$ and $R$, denoted $A$, is a triple $\langle U, R, P \rangle$, where $P$ is a set of permissions of the form $\langle u, r \rangle$, with $u \in U$ and $r \in R$, stating the accesses to be allowed.
  – The set of permissions ($P$) can be represented through an access matrix $M_A$.
  – $acl(r)$ denotes the access control list of $r$ (i.e., the set of users that can access $r$).
• Authorization Policy Graph:
  
  – Informally: We model an authorization policy as a directed and bipartite graph $G_A$ having a vertex for each user $u \in U$ and for each resource $r \in R$, and an edge from $u$ to $r$ for each permission $\langle u, r \rangle \in P$ to be enforced.

  – Formally: let $A = \langle U, R, P \rangle$ be an authorization policy. The authorization policy graph over $A$, denoted $G_A$ is a graph $\langle V_A, E_A \rangle$, where $V_A = U \cup R$ and $E_A = \{(u, r) | \langle u, r \rangle \in P\}$. 
## Authorization Policies - 3

### Security Aspects of Database Outsourcing

<table>
<thead>
<tr>
<th></th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
<th>r5</th>
<th>r6</th>
<th>r7</th>
<th>r8</th>
<th>r9</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Diagram showing relationships between entities A to F and rows r1 to r9.
Encryption Policies - 1

• Key and Token Graph:
  – \( k_i, k_j \in K \), a token \( t_{i,j} = k_j \oplus h(k_i, l_j) \).
  – Informally: We model the relationships between keys through tokens allowing derivation of one key from another, via a graph. The graph has a vertex for each pair \( \langle k, l \rangle \). There is an edge from a vertex \( \langle k_i, l_i \rangle \) to a vertex \( \langle k_j, l_j \rangle \) if there exists a token \( t_{i,j} \) allowing the derivation of \( k_j \) from \( k_i \).
  – Formally: Let \( K \) be a set of keys, \( L \) be a set of publicly available labels, and \( T \) be a set of tokens, a key and token graph over \( K, L, \) and \( T \), denoted \( G_{K,T} \), is a graph \( \langle V_{K,T}, E_{K,T} \rangle \), where \( V_{K,T} = \{ \langle k_i, l_i \rangle | k_i \in K, l_i \in L \} \) and \( E_{K,T} = \{ \langle k_i, l_i \rangle, \langle k_j, l_j \rangle | t_{i,j} \in T \} \).
Encryption Policies - 2

![Graph Diagram]

Security Aspects of Database Outsourcing
• Key assignment and encryption schema:

  – Let $U, R, K, L$ be the set of users, resources, keys, and labels in the system, respectively. A key assignment and encryption schema over $U, R, K, L$ is a function $\varphi : U \cup R \rightarrow L$ that returns for each user $u \in U$ the label $l \in L$ associated with the (single) key $k$ in $K$ released to the user and for each resource $r \in R$ the label $l \in L$ associated with the (single) key $k$ in $K$ with which the resource is encrypted.
• Encryption policy:
  – Let $U$ and $R$ be the set of users and resources in the system, respectively. An encryption policy over $U$ and $R$, denoted $E$, is a 6-tuple $\langle U, R, K, L, \varphi, T \rangle$, where $K$ is the set of keys defined in the system, $L$ is the set of corresponding labels, $\varphi$ is a key assignment and encryption schema, and $T$ is a set of tokens defined on $K$ and $L$. 
• Encryption policy graph
  
  – Let $E = \langle U, R, K, L, \varphi, T \rangle$ be an encryption policy. The encryption policy graph over $E$, denoted $G_E$, is the graph $\langle V_E, E_E \rangle$ where:
    
    • $V_E = V_{K,T} \cup U \cup R$;
    
    • $E_E = E_{K,T} \cup \{(u, \langle k, l \rangle)|u \in U \land l = \varphi(u)\}$
      
      $\cup \{((k, l), r)|r \in R \land l = \varphi(r)\}$.
Encryption Policies - 6

Security Aspects of Database Outsourcing
Policies

• Our goal is then to translate an *authorization policy* $A$ into an *equivalent encryption policy* $E$, meaning that $A$ and $E$ allow exactly the same accesses.

• Policy equivalence:

  – Let $A = \langle U, R, P \rangle$ be an *authorization policy* and $E = \langle U, R, K, L, \varphi, T \rangle$ be an *encryption policy*. $A$ and $E$ are *Equivalent*, denoted $A \equiv E$, iff

  1. $\forall u \in U, r \in R: u \rightarrow r \Rightarrow u \rightarrow r$

  2. $\forall u \in U, r \in R: u \rightarrow r \Rightarrow u \rightarrow r$
Proxy Re-Encryption

• We have not Time!
Search on Encrypted Data - 1

- Song et al. (2000)
- search for the word ‘w’ on encrypted data returns all the positions where ‘w’ occurs
- Encryption:

\[ E(W_i) = S'_i = F_{K_i}(S_i) \]

**Plain text**

Alg-0: \( K_i \) is constant.
Alg-1:
\( K_i \) is Trapdoor of \( W_i \):
\[ K_i = f_{k'}(E(wi)) \]
\( k' \) is constant key

**Cipher text**
Search on Encrypted Data - 2

- **Search**
  - send $E(w)$ and trapdoor to server
  - XOR cipher text and $E(w)$ and generate $<S_i,S'_i>$
  - if $S'_i = F_{K_i}(S_i)$ then return $i$.

- **Decrypt:**
  - $W = D(E(w))$. 

*Security Aspects of Database Outsourcing*
• Def: R-break and R-secure
  – an attack R-breaks a cryptographic primitive if the
  attack algorithm succeeds in breaking the primitive
  with resources specified by R, and we say that a
  crypto primitive is R-secure if there is no algorithm
  that can R-break it.

• Def: Distinguishing Probability or Advantage
  – Let A: \(\{0,1\}^n \rightarrow \{0,1\}\) be an arbitrary alg.
  – Let X, Y is Random var. on\(\{0,1\}^n\)
  – Advantage of A:
    • \(\text{Adv}(A) = |\Pr[A(X) = 1] - \Pr[A(Y) = 1]|\)
A pseudorandom generator $G$:
- ie, stream cipher
- We say that $G: K_g \rightarrow S$ is a $(t,e)$-secure pg if every algorithm $A$ with running time at most $t$ has advantage $< e$ ($\text{Adv}(A) < e$).

A pseudorandom function $F$:
- ie, HMAC
- We say that $F: K_f \ast X \rightarrow Y$ is a $(t,q,e)$-secure pf if every algorithm $A$ making at most $q$ queries and with running time at most $t$ has advantage $< e$ ($\text{Adv}(A) < e$).
• A pseudorandom permutation $E$
  – i.e., a block cipher
  – We say that $E: K_E \times X \rightarrow Y$ is a $(t,q,e)$-secure pp if every algorithm $A$ making at most $q$ queries and with running time at most $t$ has advantage $< e$ $(\text{Adv}(a) < e)$.

• In general, the $(t,q,e)$-security represents resistance to attacks that use at most $t$ offline work and at most $q$ adaptive chosen-text queries.
Theorems - 1

1. If $F$ is a $(t, l, e_F)$-secure pf and $G$ is a $(t, e_G)$-secure pg, then the algorithm-0 for generating the $<S_i, S’i>$ ($S’i = F_{K_i}(si))$ is a $(t-\varepsilon, e)$-secure pg, where

$$e = l \cdot e_f + e_G + \frac{l \cdot (l-1)}{2 \cdot |X|} \text{ and } \varepsilon \text{ is negl}(t).$$

2. Suppose $F$ is a $(t, l, e_F)$-secure pf, $f$ is a $(t, l, e_f)$-secure pf, and $G$ is a $(t, e_G)$-secure pg, then the algorithm-1 for generating the $<S_i, S’i>$ will be a $(t-\varepsilon, e_H)$-secure pg, where

$$e_H = l \cdot e_F + e_f + e_G + \frac{l \cdot (l - 1)}{2 \cdot |X|}.$$
3. Suppose $E$ is a $(t,l,e_E)$-secure pp, $F$ is a $(t,l,e_F)$-secure pf, $f$ is a $(t,l,e_f)$-secure pf, $G$ is a $(t,e_G)$-secure pg. Then the algorithm-2 for generating the $<S_i, S'_i>$ will be a $(t-\varepsilon, e_H)$-secure pseudorandom generator, where

$$e_H = l \times e_F + e_f + e_G + \frac{l \times (l-1)}{2 \times |X|}$$
References


Proof of Theorem-1

• Define $H': K_F \ast K_G \rightarrow (X \ast Y)^l$ by

\[ H'(k, k_j) = \langle s_1, F_k(s_1), \ldots, s_l, F_k(s_l) \rangle \]

• Lemma-1. if $F$ is a $(t, l, e_F)$-secure pf and $G$ is a $(t, e_G)$-secure pg, then $H'$ is a $(t - \varepsilon, e_{H'})$-secure pg, where

\[ e_{H'} = e_F + e_G + \frac{l \ast (l-1)}{2 \ast |X|} \text{ and } \varepsilon \text{ is } \text{negl}(t). \]
Proof of Theorem-1

- Define $H: (K_F)^l \times K_G \rightarrow (X \times Y)^l$ by
  
  $H(k, k_j) = \langle s_1, F_k(s_1), ..., s_l, F_k(s_l) \rangle$

  that $k = \langle k_1, ..., k_l \rangle$

- Lemma-2. if $F$ is a $(t, l, e_F)$-secure pf and $G$ is a $(t, e_G)$-secure pg, then $H$ is a $(t - \epsilon, e_H)$-secure pg, where

  $e_H = l \times e_F + e_G$ and $\epsilon$ is $\text{negl}(t)$. 

Security Aspects of Database Outsourcing
Proof of Theorem-1

• proof of lemma-1:
  – Define \( \psi(k) = \langle u_1, F_k(u_1), \ldots, u_l, F_k(u_l) \rangle \)
    • \( u_i \) are independent random var. from uniform distribution \( X \).
  – Let \( U \) be a random var. with uniform distribution on \((X \times Y)^l\).
  – The goal is to show that \( H' \) and \( U \) are indistinguishable to any computationally-bounded adversary. first show that \( H' \) and \( \psi \) are indistinguishable, and second show that \( \psi \) and \( U \) are indistinguishable.
  – First, we show that no algorithm with running time \( t - \varepsilon \) can distinguish between \( H' \) and \( \psi \) with advantage better than \( e_G \).
Proof of Theorem-1

– Suppose not, i.e., there exists an algorithm $A$ with running time at most $t-\varepsilon$ and

\[\text{Adv}(A) = |\Pr[A(H') = 1] - \Pr[A(\psi) = 1]| \geq e_G.\]

– Then we exhibit an algorithm $B$ with running time at most $t$ which distinguishes the output of $G$ from a truly random bit string with advantage at least $e_G$.

– The algorithm $B$:

  • Input: $s = \langle s_1, \ldots, s_l \rangle \in X^l$
  • Run $A$ on input $I = \langle s_1, F_k(s_1), \ldots, s_l, F_k(s_l) \rangle$
  • Output: $A(I)$

\[\text{Adv}(B) = |\Pr[B(G(k_G)) = 1] - \Pr[B(U') = 1]| \]
\[= |\Pr[A(H') = 1] - \Pr[A(\psi) = 1]| \]
\[= \text{Adv}(A) \geq e_G \]

$U'$ is a uniformly-distributed random variable on $X^l$. 