Secure Data Outsourcing

Tutorial @ COMAD 2006, New Delhi, India

Radu Sion
Stony Brook NSAC Lab
sion@cs.stonybrook.edu
"I have much experience only in teaching graduate students [...] and as a result [...] I know that I don't know how to teach."
Overview

- Crypto Crash Course
- Data Outsourcing
- Query Correctness
- Data Confidentiality
- Access Privacy
- Searching on Encrypted Data
- Trusted Hardware
• Randomness
• Crypto Hashes
• Encryption
• Public key encryption
• Signatures
• Ciphers
• Semantic Security
• Forward Secrecy
• Performance
• Merkle/Hash trees
Crypto: Meet the cast

Stony Brook Network Security and Applied Cryptography Lab

Alice (innocent)

Trent (trusted guy)

Eve (eavesdrops, passive malicious)

Bob (mostly innocent, sometimes malicious)

Mallory ("malicious", bad guy)

M ("malicious", bad guy)

just listens

does stuff too

\[ E_k(M) \]

k

Radu Sion

Secure Data Outsourcing (COMAD, Dec 2006)
**Crypto: Randomness**

**Cryptographically random numbers**: a sequence of numbers $X_1, X_2, \ldots$ such that for any integer $k > 0$, it is **impossible** for an observer to predict $X_k$ even if all of $X_1, \ldots, X_{k-1}$ are known.

**Problem**: True RNGs cannot be deterministically algorithmic in a closed system. “Anyone who considers arithmetic methods … is in a state of sin” (von Neuman)

**Being creative**: simulate a sequence of cryptographically random numbers but generate them by an algorithm.

**Pseudo-random numbers**: a sequence of numbers $X_1, X_2, \ldots$ such that for any integer $k > 0$, it is **hard** for an observer to predict $X_k$ even if all of $X_1, \ldots, X_{k-1}$ are known.
• A hash is a **one-way, non-invertible** function of that produces **unique** (with **high likelihood**), **fixed-size** outputs for different inputs.

• The probability of any bit “flipping” in the output bit-string should be always $\frac{1}{2}$ for any change (even one bit) in the input (“randomness”).
Crypto: PKI

private_A  public_A  

Trent

private_B  public_B

1. $S_T$(time, expiration, “Bob”, public_B)

2. $E_{public_B}(M)$

3. $M = D_{private_B}(E_{public_B}(M))$

“certificate authority”

no problema

“public key certificate”

Alice

Bob

private_B

Trent

public_B

“certificate authority”

Mallory

Eve

Radu Sion

Secure Data Outsourcing (COMAD, Dec 2006)
\[ M = D_{\text{private}_A}(E_{\text{public}_A}(M)) = D_{\text{public}_A}(E_{\text{private}_A}(M)) \]

Alice \quad public_A \quad private_A

Mallory

Eve

no problemo

\[ S_A(M) = E_{\text{private}_A}(M) \]

Bob

M = D_{\text{public}_A}(S_A(M)) ?
Crypto: RSA in a nutshell

Stony Brook Network Security and Applied Cryptography Lab

n = pq

e = 17

message \( m < n \)

c = m^e \mod n

Extended Euclidean

d = e^{-1} \mod (p-1)(q-1)

RSA Encryption

RSA Decryption

m = c^d \mod n

Alice

Bob

Radu Sion

Secure Data Outsourcing (COMAD, Dec 2006)
Crypto: Condensed RSA

messages \{m_i\}

\begin{align*}
\{s_i = m_i^d \mod n\}
\end{align*}

"Condensed RSA Signature"

s = \prod s_i

unforgeable against adaptive chosen message attacks

verification: check that \( s = (\prod m_i)^e \)
The compromise of individual blocks should not lead to the compromise of past communication!
\textbf{E()} is \textbf{indistinguishable under a chosen plaintext attack} (IND-CPA, “semantically secure”) if no probabilistic polynomial time-bounded Mallory can succeed in finding \(x'\), significantly better than guessing.
Crypto: Semantic Security

Stony Brook Network Security and Applied Cryptography Lab

Secure Data Outsourcing (COMAD, Dec 2006)
• Deterministic + stateless = insecure !
• Semantic security implies *bit security* !
• RSA : non-semantically secure ! Why ?!
• RSA + padding (e.g., RSA-OAEP): ok
Future compromise (e.g., of PK secrets) should not propagate backwards in time.
Illustrative baseline.
Pentium 4. 3.6GHz.
1GB RAM. 11000 MIPS.
OpenSSL 0.9.7f

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES/CBC</td>
<td>70MB/sec</td>
</tr>
<tr>
<td>RC4</td>
<td>138MB/sec</td>
</tr>
<tr>
<td>MD5</td>
<td>18-615MB/sec</td>
</tr>
<tr>
<td>SHA1</td>
<td>18-340MB/sec</td>
</tr>
<tr>
<td>Modular MUL 1024</td>
<td>273000/sec</td>
</tr>
<tr>
<td>RSA1024 Sign</td>
<td>261/sec</td>
</tr>
<tr>
<td>RSA1024 Verify</td>
<td>5324/sec</td>
</tr>
<tr>
<td>3DES</td>
<td>26MB/sec</td>
</tr>
</tbody>
</table>
trust this (store or authenticate)

compare

Idea: no need to be binary

Idea: sign stuff (when ?)
Crypto: Hash chains

Can we break the chain?

Trust these (store or authenticate)

\[ h_1 = H(x_1 | h_0) \]
\[ h_2 = H(x_2 | h_1) \]
\[ h_i = H(x_i | h_{i-1}) \]

Radu Sion

Secure Data Outsourcing (COMAD, Dec 2006)
Overview

- **Crypto Crash Course**
- **Data Outsourcing**
- **Query Correctness**
- **Data Confidentiality**
- **Access Privacy**
- **Searching on Encrypted Data**
- **Trusted Hardware**
Data Outsourcing

"outsourcing" ∈ \{query correctness, data confidentiality, access privacy\}

Data Pre-Processor

Online Query Interface

Query Processor

data client(s!)

Mallory  Eve

data server (un-trusted)

assurances ⊆ \{query correctness, data confidentiality, access privacy\}

encrypted

plaintext

"owner"

Data

"outsourcing"
Outsourcing Challenges

Un-trusted server:
- lazy: incentives to perform less
- curious: incentives to acquire information
- malicious:
  - denial of service
  - incorrect results
  - possibly compromised

Why is this hard?
- how?
  - arbitrary expressivity
  - overheads
    - network
    - computational costs

What do we do?
- query assurances
- full privacy
  - of queries (even encrypted)
  - of access patterns
- data confidentiality
Hacigumus (2002)

Stony Brook Network Security and Applied Cryptography Lab


Stored Data Confidentiality

```sql
SELECT decrypt(discount, key)
FROM lineitem
WHERE custid = 300
```
Overview

- Crypto Crash Course
- Data Outsourcing
- Query Correctness
- Data Confidentiality
- Access Privacy
- Searching on Encrypted Data
- Trusted Hardware
Client requires quantifiable assurances that query results are correct, for arbitrary query types in the presence of a server that could be …

… lazy

… and/or fully malicious (!)
The owner provides database updates and summary signatures to the un-trusted publisher. When users make inquiries with the publisher, they get responses which can be verified using a returned verification-object. Only $sk_o$ is secret, $pk_o$ is authenticated.
A Merkle tree, with a continuous sub-range $q$, with a least common ancestor $\text{LCA}(q)$, and upper and lower bounds. Note the verifiable hash path “l” from $\text{LCA}(q)$ to the root, and the proximity sub-trees (thick lines) for the “near miss” tuples for $\text{LUB}(q)$ and $\text{GLB}(q)$ which show that $q$ is complete.
Supported claimed operations:

- selections
- projections
  - (1) maintaining VOs before duplicate elimination
  - (2) pre-computing VOs for common projections
- equiJOIN
  - (1) keep materialized cartesian product \( S \times R \)
    - construct VO on sorted version of product
    - (according to difference \( (S.A-R.A) \)) – this yields
      3 types of leaf nodes (“0”, “<”, “>”) in Merkle tree
  - (2) all kinds of other tricks
- set operations
  - union (client does it and verifies VOs for input sets)
  - intersection (?)
  - multi-dimensional range queries (generalizing hash tree to “multi-dimensional range tree”)
Covering canonical roots (CCR): roots of the canonical sub-trees precisely covering the leaves with values in the interval.
SELECT $S.A4$ FROM $S,R$
WHERE $S.A1=R.A1$ AND $A2<10$ AND $A3>17$
Issues:
- query expressiveness
- query flexibility
  - works only on data with VOs
- “universe split” phenomenon
  - use timestamps, expiration times
- expensive operations (!)
Discusses the use of batch verification of signatures and similar techniques (condensed RSA) to authenticate results.

<table>
<thead>
<tr>
<th>Sign</th>
<th>Condensed-RSA</th>
<th>Batch-DSA</th>
<th>BGLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 signature</td>
<td>6.82</td>
<td>3.82</td>
<td>3.54</td>
</tr>
<tr>
<td>Verify</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t = 1000, k = 1</td>
<td>0.16</td>
<td>8.52</td>
<td>62</td>
</tr>
<tr>
<td>t = 100, k = 10</td>
<td>44.12</td>
<td>1623.59</td>
<td>184.88</td>
</tr>
<tr>
<td>t = 1000, k = 10</td>
<td>441.1</td>
<td>16203.5</td>
<td>1570.8</td>
</tr>
</tbody>
</table>

Cost comparison (in msecs): verification and signing. Notation: t – # signatures, k – # signers
“edge computing”
Claimed problems with [Devanbu 2000]

- A hash tree is needed for every sort-order
- VOs need to contain links all the way to the root,
  - VOs grow linearly to query result and logarithmic to base table size
- Projections may have to be performed by clients
- No provision for dynamic updates on the database

Aim 1: VO size just linear in query result
Aim 2: do not push projections to client
Idea: use different hash function

- \( h(x) = g^x \mod q \)
- \( h \) is commutative, \( h(x+y) = h(y+x) \)
  - Digests can be combined arbitrarily
  - Projection can be performed at the edge servers
  - Facilitates insertion of new tuples with minimal effect on other digests

- but: significantly (1000-10000 times) slower
- trade-off: computation vs. communication
Verifying Selection
(no need to go up to the root
as everything is also signed)
Similar expressiveness. But …

Asks: what about access control rules? (Devanbu seems to reveal too much: boundary tuples)

Also claims: lower overheads for queries and updates.

Introduces “precision” (only data matching the query should be returned)
Idea: use signature chains – thus no need to reveal boundary elements.

$$\text{sig}(r_i) = s(h(g(r_{i-1}) \mid g(r_i) \mid g(r_{i+1})))$$

Server: $\cdots \ r_{i-1} \quad r_i \quad r_{i+1} \quad r_{i+2} \quad \cdots$

User: $\cdots \ g(r_{i-1}) \quad g(r_i) \quad g(r_{i+1}) \quad g(r_{i+2}) \quad \cdots$

Result $Q$:

$$s^{-1}(\text{sig}(r_i))? \quad s^{-1}(\text{sig}(r_{i+1}))?$$
But what is $g$: $g(r) = h^{U-r-1}(r)$

**Distributor:**

$$h^\alpha r_{a-1}^{-1}$$

$$r_{a-1} \quad r_a \quad r_{a+1} \quad \cdots \quad r_n \quad g(r_{n+1})$$

**User:**

Hash $U - \alpha$ times

$$g(r_{a-1}) \quad g(r_a) \quad g(r_{a+1}) \quad \cdots \quad g(r_n)$$

$$s^{-1}(\text{sig}(r_a)) \quad s^{-1}(\text{sig}(r_{a+1}))$$

**Result Q**

Query: $\alpha \leq r$
Relational Query: $\alpha \leq K \leq \beta$

Result: $\{ | r_a, r_{a+1}, \ldots, r_b | \}$

Record $r_{a-1}$: $[ K, A_1, A_2, \ldots, A_R ]$
Asks: What about arbitrary queries?

P. Golle and I. Mironov, "Uncheatable Distributed Computations", RSA 2001 (Cryptographer's track)
A challenge token (computed by client) is sent together with the batch of queries. Upon return, batch execution is proved if \( x = x' \).
The behavior of $P'(w, r, f)$ (fake tokens) plotted against $P_c(w, r)$ (client-side result checking mechanism) showing that the query execution proof mechanism (with fake tokens) significantly decreases the ability to "get away" with less work.

Only handles lazy server!
Overview

- Crypto Crash Course
- Data Outsourcing
- Query Correctness
- Data Confidentiality
- Access Privacy
- Searching on Encrypted Data
- Trusted Hardware
Hacigumus (SIGMOD 2002)  

Stony Brook Network Security and Applied Cryptography Lab

---

Client Site

- Query Executor
- Temporary Results

Actual Results
- Meta Data
- Query Translator
- Original Query

Web Browser (USER)

Server Site

- Encrypted Results
- Query over Encrypted Data

Service Provider
- Encrypted Client Database
Main Steps:
1. Partition sensitive domains
   • Order preserving: supports comparison
   • Random: query rewriting becomes hard
2. Rewrite queries to target partitions
3. Execute queries and return results
4. Prune/post-process results on client
Hacigumus (SIGMOD 2002)

```
SELECT emp.name FROM emp
WHERE emp.salary >
  (SELECT AVG(salary) FROM emp WHERE did=1)
```

(a) Original query tree.

(b) Replacing encrypted relations.
Client pruning could be expensive

(c) Doing selection at server.

(d) Multiple interactions between Client and Server.
Confidentiality-Overhead Trade-off

Larger segments ==
  increased privacy ==
  increased overheads
Goal: For a uniform distribution of queries - minimize any leaks to any adversaries (even) knowing segmentation parameters.

Idea 1: Maximize variance of distribution of values in segment
Idea 2: Increase segment entropy

Issue: What about performance?
Solution: “Controlled Diffusion”

Idea:
1. design for efficiency, then …
2. … diffuse (re-distribute) elements inside the segments to increase per-segment entropy and variance
Asks: Similarly, how to structure query trees to optimally balance the security-efficiency trade-off in [Hacigumus 2002].

Idea: client generates optimal partitioned query execution plans given statistics and metadata input from the server.
Overview

- Crypto Crash Course
- Data Outsourcing
- Query Correctness
- Data Confidentiality
- Access Privacy
- Searching on Encrypted Data
- Trusted Hardware
QR PIR

**Question d[i]=?**

Perform same protocol per column and look at returned product of interest

\[ \Pi(v[i]\cdot d[i]) = \text{QNR} ? \]

- Yes
  - \( d[i] = 1 \)

\[ \Pi(v[i]\cdot d[i]) \]

\[ v[i] \]

\[ \text{bit string } d[n] \]

**escape O(n) costs**

Perform same protocol per column and look at returned product of interest

\[ \Pi_1, \Pi_2, \ldots, \Pi_n \]
The $n$ bits of the database are organized logically at the server as a bi-dimensional matrix $M$ of size $\sqrt{n} \times \sqrt{n}$. To retrieve bit $M(x, y)$ with computational privacy, the client:

- randomly chooses two prime numbers $p$ and $q$ of similar bit length, computes their product, $N = pq$ and sends it to the server.

- generates $\sqrt{n}$ numbers $s_1, s_2, \ldots, s_{\sqrt{n}}$, such that $s_x$ is a quadratic non-residue (QNR) and the rest are quadratic residues (QR) in $\mathbb{Z}_N^*$.

- sends $s_1, s_2, \ldots, s_{\sqrt{n}}$ to the server.

For each “column” $j \in (1, \sqrt{n})$ in the $\sqrt{n} \times \sqrt{n}$ matrix, the server:

- computes the product $r_j = \prod_{0 < i < \sqrt{n}} q_{ij}$ where $q_{ij} = s_i^2$ if $M(i, j) = 1$ and $q_{ij} = s_i$ otherwise $^2$.

- sends $r_1, \ldots, r_{\sqrt{n}}$ to the client.

The client then simply checks if $r_y$ is a QR in $\mathbb{Z}_N^*$ which implies $M(x, y) = 1$, else $M(x, y) = 0$. 

Radu Sion Secure Data Outsourcing (COMAD, Dec 2006) 55
PIR is (still) impractical

Comparison between the time required to perform PIR and the time taken to transfer the database, between 1995 and 2005. (logarithmic)
Overview

- Crypto Crash Course
- Data Outsourcing
- Query Correctness
- Data Confidentiality
- Access Privacy
- Searching on Encrypted Data
- Trusted Hardware
Search on Encrypted Data

Stony Brook Network Security and Applied Cryptography Lab

- Sequential Scan
- Index-based
Encryption:

\[ W_i \]

\[ L_i \quad R_i \]

\[ L_i \leftarrow G_i \text{ (seed)} , \quad R_i \leftarrow F_K(L_i) \]
Decryption:

\[ L_i \leftarrow G_i \text{(seed)}, \quad R_i \leftarrow F_K( L_i) \]
Search:

Check: $R_i' = F_K(L_i')$?
Yes $\Rightarrow$ match,
( false positive rate $= 1 / 2^{m-n}$ )
“Hidden” Search:

\[
L_i \leftarrow G_i \text{(seed)}, \\
R_i \leftarrow F_{K_i}(L_i)
\]

where \(K_i = F'_K(E_1(W_i))\)

---

\[
L_i \leftarrow G_i \text{(seed)}, \\
R_i \leftarrow F_{K_i}(L_i)
\]

where \(K_i = F'_K(E_1(W_i))\)
Chang (2004)

Index

Files

$|D|$
Chang (2004)
random seed $s_2$

Pseudo Random Function

Send the pseudo random seed to Server

Pseudo Random Function

Masked Index
Server stores capabilities for conjunctive queries (linear in the total number of documents). These can be transferred offline.

The client is required to know before-hand future conjunctive queries.

Query part is sent online at the time of search. It is of constant size (number of keyword fields per documents).
Asks: What about correctness + privacy?

Idea: Deploy modified version of computational PIR targeted at a server-side index. Augment with "multiplicative checksums".
Overview

- Crypto Crash Course
- Data Outsourcing
- Query Correctness
- Data Confidentiality
- Access Privacy
- Searching on Encrypted Data
- Trusted Hardware

Stony Brook Network Security and Applied Cryptography Lab
Trusted Hardware

IBM 47xx
IBM 4764 Architecture

Stony Brook Network Security and Applied Cryptography Lab

Radu Sion

Secure Data Outsourcing (COMAD, Dec 2006)
Trust Propagation

Software:
- Applications
- Environment/OS
- Kernel
- Loaders

Firmware:
- Post
- Miniboot

Hardware:
- Processor
- Flash, RAM, ROM
- Locks
- Tamper-responding unit
- Crypto functions
SCPU Performance

RSA1024 Sign: 848/sec
RSA1024 Verify: 1157/sec
3DES: 1-8MB/sec
DES: 1-8MB/sec
SHA1: 1-21MB/sec

IBM 4764-001: 266MHz PowerPC. 64KB battery-backed SRAM storage. Crypto hardware engines: AES256, DES, TDES, DSS, SHA-1, MD5, RSA. FIPS 140-2 Level 4 certified.
Possible Benefits

A secure co-processor on the data management side may allow for significant leaps in expressivity for queries where privacy and completeness assurance are important.
For conjunctive keyword searches on document (email, files) servers, oblivious search index structures could be queried in secure memory achieving a novel zero-leak query model.
Hash-JOIN could be naturally accommodated.
For Merge-JOIN, order-preserving encryption primitives could be deployed to minimize the amount of data parsing required in the sorting phase.
Sample DON’T

Secure Data Outsourcing (COMAD, Dec 2006)

Stony Brook Network Security and Applied Cryptography Lab

Good idea?  
not so sure!

crypto work

client/server interaction

data client

database server

queries

Outsourced Data

Server Storage

Host CPU

crypto work

crypto work

“client proxy”

“client-server” interaction

queries

data client

database server

Outsourced Data

Server Storage

Host CPU
Other DON’Ts

• Process entire queries on SCPU (!)

• Dedicate (one) SCPU per query or equivalent
  • e.g., limit TPS by SCPU TPS

• Synchronize CPU with SCPU
  • e.g., block main CPU until SCPU completes

• Transfer >= O(n) on SCPU-CPU bus (!)

• Anything else un-smart 😊
Bouganim (VLDB 2002)

Stony Brook Network Security and Applied Cryptography Lab

Intruder

Insider

Client

Secured communications

DBMS

Encryption

Decryption

Encrypted Database

Database Server

Client

Encryption

Decryption

Secured communications

DBMS

Encrypted Database

Database Server
Chip-Secured Data Access:

Smartcard: 32 bit RISC processor (≈ 40Mips), limited communication bandwidth (10 to 100 Kbps), tiny RAM, writes in EEPROM very costly.
Equi-predicate-only Queries:

Select * from Customers where City = 'Hong Kong'

<table>
<thead>
<tr>
<th>Id</th>
<th>name</th>
<th>City</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Jim</td>
<td>Hong</td>
<td>good</td>
</tr>
<tr>
<td>19</td>
<td>Joe</td>
<td>Hong</td>
<td>bad</td>
</tr>
</tbody>
</table>
General queries:

Select sum(amount) from orders where CustId = 22

Select ygefh from iuzgs where lpaszj = "euys"

C-SDA

Access Rights

Compute

Decrypt

DBMS

Encrypted Database

Sum

1200

Bouganim (VLDB 2002)  Stony Brook Network Security and Applied Cryptography Lab
Tsudik (2005)
Practical maturity: in infancy, barely crawling. Very hard problems remain to be tackled:

- operators with integrated assurances
  - confidentiality
  - privacy of access
  - correctness
- scalable protocols for secure hardware
  - massive data
  - good utilization of host CPUs
- areas
  - relational data
  - file systems
  - streaming data
/bin/yes > /dev/lunchtime

THANK YOU!


refs: correctness


Tran Khanh Dang, Privacy-Preserving Search and Updates for Outsourced Tree-Structured Data on Untrusted Servers, Springer LNCS 3477/2005, pages 338-354

Hakan Hacigumus, Bala Iyer, Sharad Mehrotra, Query Optimization in Encrypted Database Systems, DASFAA 2005

Hakan Hacigumus, Bala Iyer, Sharad Mehrotra, Efficient Execution of Aggregation Queries over Encrypted Relational Databases, DASFAA 2004

Hakan Hacigumus, Sharad Mehrotra, Williams Jonker, Milan Petkovic, Efficient key updates in encrypted database systems, SDM 2005

Hakan Hacigümüs, Bala Iyer, Sharad Mehrotra, Ensuring the Integrity of Encrypted Databases in the Database-as-a-Service Model. DBSEC 2003

Hakan Hacigümüs, Bala Iyer, Sharad Mehrotra, Encrypted Database Integrity in Database Service Provider Model. Certification and Security in E-Services 2002
refs: service providers


Bouganim L., Pucher P., "Chip-Secured Data Access: Confidential Data on Untrusted Servers", Int. Conf. on Very Large Data Bases VLDB 2002.


“Practical server privacy with secure coprocessors”, IBM Systems Journal 2001, S. W. Smith, D. Safford


A. Iliev, S.W. Smith, "Prototyping an Armored Data Vault: Rights Management on Big Brother's Computer.", Privacy-Enhancing Technology 2002


Maheshwari, Vingralek, and Shapiro, How to build a trusted database system on untrusted storage, OSDI 2000
E. Kushilevitz and R. Ostrovsky, “Replication is not needed: single database, computationally-private information retrieval”, FOCS 1997

Radu Sion and Bogdan Carbunar, On the Practicality of Private Information Retrieval, NDSS 2006


Josep Domingo-Ferrer, Ricardo X. Sanchez del Castillo, An Implementable Scheme for Secure Delegation of Computing and Data, 1st International Information and Communications Security Conference ICICS 1997