Secure and Verifiable Outsourcing of Computation and Storage

Yihua Zhang
University of Notre Dame
Outline

- Biometric Computation Outsourcing (*Tissec '13*)
- Storage Outsourcing (*Asiaccs' 13*)
- A General Framework for Private Distributed Computation (*NDSS13, CCS'13*)
Secure Outsourcing of Biometric Computation

› Biometric Data
  – Iris, Fingerprint, Face

› Privacy
  – biometric data are sensitive in nature
  – used for authentication purposes (cannot be revoked)

› Computation
  – test new algorithms for extracting biometric features.
  – force computation outsourcing to a distributed environment
  – preserve the integrity of outsourced computation

Secure and verifiable outsourcing of biometric computation!
Abstractions for Biometric Research

Four abstractions can aid biometric research

- **Select(R):** select a set of images based on requirements
  - gender, age, color
  - produce a image set S
- **Transform(S):** convert each image in S to extract features.
- **All-Pairs(S, F):** compare each pair of items in S using a function F.
  - produce a matrix M, where $M[x][y] = F(S[x], S[y])$
- **Quality(M, D):** Reduce M into a metric D that represents quality.
  - gather a distance distribution in matrix M
Contribution

- Provide a secure verification mechanism for biometric computation
  - All-Pairs
  - Quality

- Apply the technique for three different distance functions $F$
  - Hamming Distance (for iris)
  - Euclidean Distance (for face)
  - Set Intersection Cardinality (for fingerprint)

- Provides rigorous security analysis and performance evaluation results
Problem Formulation

- All-Pairs Computation:
  - given two biometric sets $S_1$ and $S_2$, compute distance between every pair of items $x \in S_1$ and $y \in S_2$ and outputs a distance matrix $M$.

- Quality Computation
  - post-process the distances in $M$ to compute distribution information for all possible distances.

- For example,
  - $S_1 = \{A_1, A_2, A_3\}$
  - $S_2 = \{B_1, B_2, B_3\}$
  - $F = \text{Hamming Distance}$

$$D_{ij} = \text{Distance Function}(A_i, B_j)$$

<table>
<thead>
<tr>
<th>Distance</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
</tr>
</tbody>
</table>

Distance Matrix $M$ Statistic Table $T$
How do we achieve privacy?

- **Step 1**: client sends each biometric item to the server in a secret form
  - secret shared
- **Step 2**: client sends each distance in the statistics table to server.
- **Step 3**: server performs All-Pairs computation securely
- **Step 4**: server obliviously updates the statistics table.
- **Step 5**: server sends the statistics results to client in a secret format
- **Step 6**: client reconstructs the results, and verifies its correctness.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_0$</td>
<td>$c_0$</td>
</tr>
<tr>
<td>$d_1$</td>
<td>$c_1$</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>$d_{L-2}$</td>
<td>$c_{L-2}$</td>
</tr>
<tr>
<td>$d_{L-1}$</td>
<td>$c_{L-1}$</td>
</tr>
</tbody>
</table>

\[
[d] := \text{dist}([x], [y]) \\
\text{for } i = 0, \cdots, (L-1) \\
[b] := ([d])^{2}_i = [d_i] \\
[c_i] := [c_i] + [b]
\]
How to verify the distribution information?

Informal answer

- insert fake biometric items into each data set
- separate the distance between real and real items from the ones between real and fake items.
Minimize the server's success rate

Strategy

- randomize the distances in the statistics table
- check a large number of ranges for real-fake distances

<table>
<thead>
<tr>
<th>Distance</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>103</td>
<td>5</td>
</tr>
<tr>
<td>201</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>202</td>
<td>1</td>
</tr>
</tbody>
</table>
Storage Outsourcing

Outsource.
Read
Write

SafeFile Inc.
Seattle, USA

Complete?
Previous Works

- **POR (Proofs of Retrievability)** and **PDP (Proofs of Data Possession)**
  - partition a collection of data into data blocks
  - store the blocks together with metadata
  - issue integrity verification queries periodically
  - use the metadata to ensure data integrity

- **DPDP (our focus)**
  - allow to issue dynamic operations
Previous Works

› Authenticated Data Structures
  
  – Merkle Hash Tree [Oprea et al. 07, Wang et al. 09, Popa et al. 11]
  – Skip List [Goodrich et al. 08, Heitzmann et al. 08, Erway et al. 09]

\[
\begin{align*}
N_1 &= H(H(C1) || H(C2)) \\
N_2 &= H(H(C3) || H(C4)) \\
R &= H(N1 || N2)
\end{align*}
\]
Contribution

Pros:
- efficiency in communication and computation
- support for revision control and range operations
- size of data structure independent of outsourced file

Cons:
- client needs to maintain the data structure by himself
Balanced Update Tree

- A node represents a dynamic operation
  - Insert [50, 60]
  - except for the retrieval operation
- Each node contains the following attributes
  - [L, U]: lower and upper bound of block range
  - Op: operation type (e.g., insert, delete, modify)
  - GID: identification of different insert or delete ops
  - Version: identification of different versions of same block
Balanced Update Tree

- How to organize the nodes within the structure?
  - the nodes are ordered based on their ranges $[L, U]$
  - all the nodes within a left subtree of a node have ranges fall into $[0, L-1]$
  - all the nodes within a right subtree of a node have ranges fall into $[U+1, +\infty]$.

- Balance the tree if necessary

- For example,
  - $[L, U]$
  - $\{\text{Op, V, ID, R}\}$
How to verify different operations?

- **Update (no verification)**
  - modify the update tree and obtain the correct attributes
  - compute a tag $t_i = \text{Mac}(d_i || Op || GID || index || V)$ for each updated block $d_i$.
  - outsource tags together with data blocks

- **Retrieve (do verification)**
  - the server returns $(d_i, t_i)$ for each retrieved block
  - the client obtains the values of $Op'$, $ID'$, $Index'$, and $V'$ for each block from the update tree.
  - the client verifies $t_i = \text{Mac}(d_i || Op' || GID' || index' || V')$
## Performance

### Computation

<table>
<thead>
<tr>
<th>Volume</th>
<th>Max Offset</th>
<th>Operations (x10^6)</th>
<th>MHT (sec)</th>
<th>SL (sec)</th>
<th>Utree (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project -0</td>
<td>170GB</td>
<td>4.2</td>
<td>6400</td>
<td>21000</td>
<td>8.4</td>
</tr>
<tr>
<td>Project -1</td>
<td>880GB</td>
<td>24</td>
<td>29000</td>
<td>95000</td>
<td>1200</td>
</tr>
<tr>
<td>Project -2</td>
<td>880GB</td>
<td>29</td>
<td>48000</td>
<td>120000</td>
<td>2200</td>
</tr>
<tr>
<td>Project -3</td>
<td>240GB</td>
<td>2.2</td>
<td>670</td>
<td>2300</td>
<td>3</td>
</tr>
<tr>
<td>Project-4</td>
<td>240GB</td>
<td>6.5</td>
<td>3400</td>
<td>12000</td>
<td>150</td>
</tr>
</tbody>
</table>
Private Distributed Computation

- **Goal:**
  - build a computational framework that privacy preserving execution on any functionality or program.

- **Features:**
  - support as wide range of functionalities as possible
    - data Structures
    - algorithms
  - Support for all major types of computation
  - achieves both theoretical security and **efficiency**
    - reduce the number of interactive ops or rounds
    - explore the use of parallelism
Framework

- How it works?
  - the compiler converts a C extension program into its secure implementation
    - based on linear secret sharing
  - the resulting program will be compiled by the native compiler and run by a number of distributed computational nodes
  - the compiler includes several aid programs
    - pre-process private inputs
    - recover the final outputs
Framework

How it works?

Compile:

Execution:
Sample Program

Source Program

private int t;

if (t>0)
  for (i=0; i<n; i+=5)
    a[i]=a[i]+1;

Compiled Program

mpz_t t;

mpz_t cond1;

mpz_t tmp1;

smc_gt(t,0,cond1);

for (i=0; i<n; i+=5) {
  tmp1=a[i];
  a[i]=a[i]+1;
  a[i]=cond1*a[i]+(1-cond1)*tmp1;
}
## Performance

<table>
<thead>
<tr>
<th>Experiment (ms)</th>
<th>PICCO (LAN)</th>
<th>PICCO (WAN)</th>
<th>Sharemind (LAN)</th>
<th>Sharemind (WAN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Multiplication (8 x 8)</td>
<td>0.45</td>
<td>32.1</td>
<td>168</td>
<td>376</td>
</tr>
<tr>
<td>Merge-sort (64 elements)</td>
<td>649.6</td>
<td>12080</td>
<td>15145</td>
<td>47636</td>
</tr>
<tr>
<td>AES (128 bit)</td>
<td>35.1</td>
<td>3179</td>
<td>652</td>
<td>N/A</td>
</tr>
<tr>
<td>FingerPrint (20 minutiae)</td>
<td>830</td>
<td>74704</td>
<td>24273</td>
<td>75820</td>
</tr>
</tbody>
</table>
Thank you !