TWO PARTY OBLIVIOUS DFA EVALUATION PROTOCOL

G. KHACHATRIAN (AUA)
M. HOVSEPYAN (RAU)
A. JIVANYAN (AUA)
TWO-PARTY SECURE FUNCTION EVALUATION

The client has a function $F: X \rightarrow Y$ and the server has an input argument $x \in X$ and the objective of these two parties to calculate $F(x)$ allowing one or both parties to learn the result, but none of them should learn any additional information about the other’s input.
REAL-LIFE APPLICATIONS

• The FBI wants to allow biogenetical researchers (clients) to find the number of people among the criminals who have a specific (featured by the researchers) pattern in their DNAs without providing the entire list of DNAs of such people, and without learning which patterns have been searched.

• A person wants to take a credit from the bank. The bank need to check whether he fits to their requirements, particularly they should check his credit history storied in Credit Report Agencies (CRA). But full credit report of a person may contain lots of private information as long as the criteria of a bank for giving a credit for the person also may be private. So the bank may want to check his criteria with the CRA without learning the full credit history and without providing the criteria to the CRA.
OTHER POSSIBLE REAL-LIFE APPLICATIONS

• Privacy-Preserving Genomic Computation
• Remote Diagnostics
• Graph Algorithms
• Data Mining
• Medical Diagnostics
• Face Recognition
• Policy Checking
• Banking
1-OUT-OF-N OBLIVIOUS TRANSFER PROTOCOL

The client has a number from 1 to \( n \) and the server has \( n \) secrets. They should communicate and after that the client should learn \( S_i \), but the server should not learn \( i \) and the client should not learn other secrets than \( S_i \).

- \( \text{Init} \) – some initialization data.
- \( Q(i) \) – query token for number \( i \) which hides \( i \) from the server.
- \( R(Q(i), \text{Init}) \) – response token which contains enough info to invoke \( S_i \), but nothing else.
WHITE BOX CRYPTOGRAPHY

Key -> plaintext

Random Data

WBC

cyphertext

plaintext -> cyphertext
WHITE-BOX AS AN ALTERNATIVE TO PUBLIC KEY

In order to be able to encrypt/decrypt messages one should have both encryption and decryption white boxes. Having only one of those white boxes one can encrypt (or decrypt) messages but s/he cannot decrypt (or encrypt) them. So white box cryptography can be used as an alternative to the public-key cryptography where encryption (decryption) white-box is public and the key is private.
The client has a deterministic finite automata (or a regular expression) $\Gamma$ instead of the function $F$ and the server has a text string $x$. And their objective is to check whether the string $x$ matches to the regular expression $\Gamma$ (i.e. $x \in L(\Gamma)$) allowing both parties to learn the answer so that neither the client nor the server learned any additional information about the input of each other.
OUR CONTRIBUTION

• Usage of white-box based OT (to make operations significantly more efficient compared with the existing approaches while preserving the same security level).

• No restriction on DFA’s input alphabet.
**Preliminaries**

- Client
  - $\Gamma(Q, \Sigma, \Delta, s_1, F)$
  - $Q$ – set of states
  - $\Sigma$ – input alphabet
  - $\Delta: Q \times \Sigma \rightarrow Q$ – transitions table
  - $s_1$ – initial state
  - $F$ – set of accepting states

- Server
  - $x \in \Sigma^n$
  - $n$ – the length of the string known by the client

$$\Gamma(x) = \left( \Delta \left[ ... \Delta \left[ \Delta[s_1, x[1]], x[2] \right] ..., x[n] \right] \right) \in F$$
BRIEF DESCRIPTION OF THE PROTOCOL

a) **Client:** Create a special matrix $M_{\Gamma}$ of size $n \times |Q| \times |\Sigma|$ intended for evaluation of $\Gamma$ on $n$-length strings.

b) **Client:** Create garbled DFA matrix $GM_{\Gamma}$ by permuting each row of the $M_{\Gamma}$ matrix, then encrypting each cell of the matrix using one time pad. As a result, to calculate $\Gamma(X)$ it will be enough for the server to have the $GM_{\Gamma}$ and a key for each position $i;1 \leq i \leq n$ of the string $x$ corresponding to the letter of that position.

c) **Server:** For each letter of string $x$ create an white-box based OT query token and send them all along with the OT initialization data to the client.

d) **Client:** For each OT query token create an OT response token for the corresponding key and send them together with the garbled DFA matrix $GM_{\Gamma}$ to the Server.

e) **Server:** From OT response tokens invoke the keys for positions of the string $x$, then compute $\Gamma(X)$ using those keys and $GM_{\Gamma}$ garbled DFA matrix.
# Complexity for $\Sigma = \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>Round Complexity</th>
<th>Client Computations</th>
<th>Server Computations</th>
<th>Network Communication Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Asymmetric</td>
<td>Symmetric</td>
<td>Asymmetric</td>
</tr>
<tr>
<td>Troncoso</td>
<td>$O(n)$</td>
<td>$O(n</td>
<td>Q</td>
<td>)$</td>
</tr>
<tr>
<td>Frikken</td>
<td>2</td>
<td>$O(n +</td>
<td>Q</td>
<td>)$</td>
</tr>
<tr>
<td>Gennaro</td>
<td>$O(\min{</td>
<td>Q</td>
<td>, n})$</td>
<td>$O(n</td>
</tr>
<tr>
<td>Yao</td>
<td>1</td>
<td>$O(n)$</td>
<td>$O(n</td>
<td>Q</td>
</tr>
<tr>
<td>Ishai</td>
<td>1</td>
<td>$O(n)$</td>
<td>None</td>
<td>$O(n</td>
</tr>
<tr>
<td>Mohassell</td>
<td>1</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>This Protocol</td>
<td>1</td>
<td>None</td>
<td>$O(n)$</td>
<td>None</td>
</tr>
</tbody>
</table>
BENCHMARKS FOR $\Sigma = \{0,1\}$

Garbled DFA creation (sec):

| $|Q|$ | $n$ | 10   | 100  | 1000 | 10000 | 100000 |
|-----|-----|------|------|------|-------|--------|
| 10  |     | <0.001 | 0.002 | 0.017 | 0.17  | 1.7    |
| 100 |     | 0.002 | 0.017 | 0.17  | 1.7   | 17     |
| 1000|     | 0.017 | 0.17  | 1.7   | 17    | >100   |
| 10000|    | 0.17  | 1.7   | 17    | >100  | >100   |
| 100000|   | 1.7   | >100  | >100  | >100  | >100   |
| 1000000|  | 17    | >100  | >100  | >100  | >100   |
BENCHMARKS FOR $\Sigma = \{0,1\}$

OT phase and garbled DFA evaluation (sec):

<table>
<thead>
<tr>
<th>$n$</th>
<th>OT phase (server)</th>
<th>OT phase (client)</th>
<th>Garbled DFA evaluation (server)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>100</td>
<td>0.002</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>1000</td>
<td>0.021</td>
<td>0.007</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>10000</td>
<td>0.21</td>
<td>0.07</td>
<td>0.002</td>
</tr>
<tr>
<td>100000</td>
<td>2.1</td>
<td>0.7</td>
<td>0.02</td>
</tr>
<tr>
<td>1000000</td>
<td>21</td>
<td>7</td>
<td>0.2</td>
</tr>
</tbody>
</table>
OTHER SECURITY RELATED SCENARIOS

• The client wants to hide the result $\Gamma(x)$ from the server.

• The client or both parties want to learn whether $x$ has substring $y$ which matches to the regular expression $\Gamma$ (i.e. $\Gamma(y) = 1$).

• The client or both parties want to learn all positions of substrings of $x$ which match $\Gamma$.

• The client or both parties want to learn the number of substrings of $x$ which match $\Gamma$. 
THANK YOU