GSHADE: Faster Privacy-Preserving Distance Computation and Biometric Identification

Thomas Schneider (TU Darmstadt)

based on joint works with
Michael Zohner (TU Darmstadt)
Julien Bringer, Hervé Chabanne, Mélanie Favre, Alain Patey (Morpho)
Gilad Asharov, Yehuda Lindell (Bar-Ilan University)

Workshop on PETs for Biometric Data, Haifa, Jan 15, 2015
Privacy-Preserving Biometric Identification

Task: Check if query is *similar* to an entry in the DB.
- without revealing the query to the server
- without revealing the DB to the client
Secure Two-Party Computation

This Talk: Passive Adversaries
Example Privacy-Preserving Applications

Auctions [NaorPS99], ...

Remote Diagnostics [BrickellPSW07], ...

DNA Searching [Troncoso-PastorizaKC07], ...

Biometric Identification [ErkinFGKLT09], ...

Medical Diagnostics [BarniFKLSS09], ...
Oblivious Transfer (OT)

OT is fundamental of many secure computation protocols.
Yao’s Garbled Circuits Protocol [Yao’86]

\[ f(\cdot, \cdot) \quad \text{e.g., } \mathbf{x} < \mathbf{y} \]

private data \( \mathbf{x} = x_1, .., x_n \)

\[ \widetilde{\mathbf{C}} \]

\[ \tilde{\mathbf{y}} \]

\[ (\tilde{\mathbf{x}}; \bot) \leftarrow \text{OT}(\mathbf{x}; (\tilde{x}^0, \tilde{x}^1)) \]

\[ f(\mathbf{x}, \mathbf{y}) = \widetilde{\mathbf{C}}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \]

private data \( \mathbf{y} = y_1, .., y_n \)

- Circuit

- Garbled Circuit \( \widetilde{\mathbf{C}} \)

- Garbled Values

- Garbled Table

OT on keys per Alice’s input bit

\

\[ E(\tilde{x}_1^0, \tilde{y}_1^0; \tilde{c}_1^{g(0,0)}) \]
\[ E(\tilde{x}_1^0, \tilde{y}_1^1; \tilde{c}_1^{g(0,1)}) \]
\[ E(\tilde{x}_1^1, \tilde{y}_1^0; \tilde{c}_1^{g(1,0)}) \]
\[ E(\tilde{x}_1^1, \tilde{y}_1^1; \tilde{c}_1^{g(1,1)}) \]
The GMW Protocol
[Goldreich/Micali/Wigderson’87]

Secret share inputs:

\[ a = a_1 \oplus a_2 \]
\[ b = b_1 \oplus b_2 \]

Non-Interactive XOR gates: \( c_1 = a_1 \oplus b_1 ; c_2 = a_2 \oplus b_2 \)

Interactive AND gates:

\[ c_1, b_1 \]
\[ d_1 \]
\[ c_2, b_2 \]
\[ d_2 \]

Recombine outputs:

\[ d = d_1 \oplus d_2 \]

Two OTs on bits per AND gate
Overview of this talk: Secure Computation

Special Purpose Protocols

- Public Key Crypto
  >>
  Symmetric Crypto

Generic Protocols

- One-Time Pad
  >>
  One-Time Pad

Part 1: Efficient OT Extensions

Part 2: GSHADE

Boolean Circuit

- GMW
- Yao

Symmetric Crypto

- GMW
- Yao

Public Key Crypto

- GMW
- Yao

OT

- GMW
- Yao
Part 1: Efficient OT Extensions


http://encrypto.de/code/OTExtension
OT - Bad News

- [ImpagliazzoRudich’89]: there’s no black-box reduction from OT to OWFs

- Several OT protocols based on public-key cryptography
  - e.g., [NaorPinkas’01] yields ~1,000 OTs per second

- Since public-key crypto is expensive, OT was believed to be inefficient
OT - Good News

- [Beaver’95]: OTs can be pre-computed (only OTP in online phase)

- OT Extensions (similar to hybrid encryption):
  - use symmetric crypto to stretch few “real” OTs into longer/many OTs
    - [Beaver’96]: OT on long strings from short seeds
    - [IshaiKilianNissimPetrank’03]: many OTs from few OTs
OT Extension of [IKNP’03] (1)

- Alice inputs $m$ pairs of $\ell$-bit pairs $(x_{i,0}, x_{i,1})$

- Bob inputs $m$-bit string $r$ and obtains $x_{i,r_i}$ in $i$-th OT
OT Extension of [IKNP’03] (2)

- Alice and Bob perform $k$ “real” OTs on random seeds with reverse roles ($k$: security parameter)
OT Extension of [IKNP’03] (3)

- Bob generates a random $m \times k$ bit matrix $T$ and masks his choices $r$

- The matrix is masked with the stretched seeds of the “real” OTs

\[
T \in_R \{0, 1\}^{m \times k} \\
\text{for } 1 \leq j \leq k: \\
u_{j,0} = \text{PRG}(s_{j,0}) \oplus T[j] \\
u_{j,1} = \text{PRG}(s_{j,1}) \oplus T[j] \oplus r
\]

\[
V[j] = u_{j,c_j} \oplus \text{PRG}(s_{j,c_j})
\]

PRG: pseudo-random generator (instantiated with AES)
OT Extension of [IKNP’03] (4)

- Transpose matrices $V$ and $T$

- Alice masks her inputs and obliviously sends them to Bob

\[
V' = V^T \quad \text{and} \quad T' = T^T
\]

\[
\begin{align*}
\text{for } 1 \leq i \leq m: \\
y_{i,0} &= x_{i,0} \oplus H(i, V'[i]) \\
y_{i,1} &= x_{i,1} \oplus H(i, V'[i] \oplus c)
\end{align*}
\]

\[
(y_{i,0}, y_{i,1}), 1 \leq i \leq m \\
\implies x_{i,r_i} = y_{i,r_i} \oplus H(i, T'[i])
\]

$H$: correlation robust function (instantiated with hash function)
Computation Complexity of OT Extension

For 1 ≤ j ≤ k:

\[ c_j \in_R \{0,1\} \]
\[ (s_{j,0}, s_{j,1}) \in_R \{0,1\}^{2k} \]

\[ r = (r_1, \ldots, r_m) \in \{0,1\}^m \]

\[ e_{j,c_j} \rightarrow OT \]
\[ (s_{j,0}, s_{j,1}) \rightarrow (s_{j,0}, s_{j,1}) \]

\[ T \in_R \{0,1\}^{m \times k} \]

\[ u_{j,0} = PRG(s_{j,0}) \oplus T[j] \]
\[ u_{j,1} = PRG(s_{j,1}) \oplus T[j] \oplus r \]

\[ V[i] = u_{j,c_j} \oplus PRG(s_{j,c_j}) \]

\[ V^T = V^T \]
\[ T' = T^T \]

Time distribution for 10 Mio. OTs (in 21s):

- "real" OTs: 10%
- H (SHA-1): 33%
- PRG (AES): 42%
- Transpose: 14%
- Misc (Snd/Rcv/XOR): 1%

Non-crypto part is bottleneck!!!
Algorithmic Optimization
Efficient Bit-Matrix Transposition

- Naive matrix transposition performs $mk$ load/process/store operations

- Eklundh's algorithm reduces number of operations to $O(m \log_2 k)$ swaps
  - Swap whole registers instead of bits
  - Transposing 10 times faster
Algorithmic Optimization
Parallelized OT Extension

- OT extension can easily be parallelized by splitting the $T$ matrix into sub-matrices.

- Since columns are independent, OT is highly parallelizable.
Communication Complexity of OT Extension

$m$ pairs $(x_{i,0}, x_{i,1}) \in \{0,1\}^2$

$r = (r_1, \ldots, r_m) \in \{0,1\}^m$

for $1 \leq j \leq k$:

$\ell \in R \{0,1\}$

$(s_{j,0}, s_{j,1}) \in R \{0,1\}^{2k}$

$s_{j,c_j}$

OT

$T \in R \{0,1\}^{m \times k}$

for $1 \leq j \leq k$:

$u_{j,0} = PRG(s_{j,0}) \oplus T[j]$  

$u_{j,1} = PRG(s_{j,1}) \oplus T[j] \oplus r$

for $1 \leq j \leq k$:

$V[j] = u_{j,c_j} \oplus PRG(s_{j,c_j})$

$V' = V^T$

$T' = T^T$

for $1 \leq i \leq m$:

$y_{i,0} = x_{i,0} \oplus H(i, V'[i])$

$y_{i,1} = x_{i,1} \oplus H(i, V'[i] \oplus c)$

$(y_{i,0}, y_{i,1}), 1 \leq i \leq m$

for $1 \leq i \leq m$:

$x_{i,c_i} = y_{i,c_i} \oplus H(i, T'[i])$

Yao: $\ell = k = 80$

GMW: $\ell = 1, k = 80$

Per OT:

$2\ell$

Bits sent

$2k$
Protocol Optimization
General OT Extension

- Instead of generating a random $T$ matrix, we derive it from $s_{j,0}$

- Reduces data sent by Bob by factor 2
Specific OT Functionalities

- Secure computation protocols often require a specific OT functionality
  - Yao with free XORs requires strings $x_0, x_1$ to be XOR-correlated
  - GMW with multiplication triples can use random strings

- Correlated OT: random $x_0$ and $x_1 = x_0 \oplus x$
  - e.g., for Yao

- Random OT: random $x_0$ and $x_1$
  - e.g., for GMW
Specific OT Functionalities
Correlated OT Extension (C-OT)

- Choose $x_{i,0}$ as random output of $H$ (modeled as RO here)

- Compute $x_{i,1}$ as $x_{i,0} \oplus x_i$ to obliviously transfer XOR-correlated values

- Reduces data sent by Alice by factor 2
Specific OT Functionalities
Random OT Extension (R-OT)

- Choose $x_{i,0}$ and $x_{i,1}$ as random outputs of $H$ (modeled as RO here)

- No data sent by Alice
### Performance Evaluation

**Conclusion**

- **OT** is very efficient

- **Communication** is the **bottleneck** for OT (even without using AES-NI)

---

#### Performance for 10 Mio. OTs on 80-bit strings

<table>
<thead>
<tr>
<th>Method</th>
<th>Gigabit LAN</th>
<th>WiFi 802.11g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig</td>
<td>20.6</td>
<td>30.7</td>
</tr>
<tr>
<td>EMT</td>
<td>14.4</td>
<td>30.5</td>
</tr>
<tr>
<td>G-OT</td>
<td>13.9</td>
<td>29.4</td>
</tr>
<tr>
<td>C-OT</td>
<td>10.6</td>
<td>14.4</td>
</tr>
<tr>
<td>R-OT</td>
<td>10.0</td>
<td>14.2</td>
</tr>
<tr>
<td>2T</td>
<td>5.0</td>
<td>14.2</td>
</tr>
<tr>
<td>4T</td>
<td>2.6</td>
<td>14.2</td>
</tr>
</tbody>
</table>

---

Gigabit LAN: Gigabit LAN performance for 10 Mio. OTs on 80-bit strings.

WiFi 802.11g: WiFi 802.11g performance for 10 Mio. OTs on 80-bit strings.
J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, M. Zohner: 
GSHADE: Faster privacy-preserving distance computation and biometric identification. 
In ACM IH&MMSEC’14.
Task: Check if query is similar to an entry in the DB.
- without revealing the query to the server
- without revealing the DB to the client
Use-Cases

Biometric Access Control / Border Control

Anonymous Biometric Credentials

Secure Biometric Database Intersection
The SCiFI Algorithm
[Osadchy/Pinkas/Jarrous/Moskovitch S&P’10]

Compute Hamming distance of $\ell=900$ bit strings and compare with threshold.
## Privacy-Preserving Biometric Identification: Classification

<table>
<thead>
<tr>
<th>Technique</th>
<th>Public-Key Crypto</th>
<th>Boolean / Hybrid</th>
<th>OT-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hamming (HD)</td>
<td>[OPJM10]</td>
<td>[HEKM11]</td>
<td>[BCP13] SHADE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[SZ13]</td>
<td>GSHADE</td>
</tr>
<tr>
<td>Euclidean</td>
<td>[EFG+09]</td>
<td>[SSW09]</td>
<td>GSHADE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[HKS+10]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[BG11]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[HMEK11]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[SZ13]</td>
<td></td>
</tr>
<tr>
<td>Normalized HD</td>
<td>-</td>
<td>[BG11]</td>
<td>GSHADE</td>
</tr>
</tbody>
</table>
Secure Hamming Dist. computation from OT [BringerChabannePatey'13]

Goal: compute $\text{HD}(X,Y) = \sum (x_i \oplus y_i), \ i=1..\ell$

for $i=1..\ell$:

choose $r_i \in_R \mathbb{Z}_{\ell+1}$

$r_i + y_i ; r_i + (1-y_i)$

$\text{OT}$

$x_i$

$t_i = r_i + (x_i \oplus y_i)$

$R = \sum r_i$

$T = \sum t_i = R + \text{HD}(X,Y)$

Continue with generic MPC protocol (e.g., Yao or GMW)

from $T - R = \text{HD}(X,Y)$ …
GSHADE: Optimizations and Generalization of SHADE

- For multiple HD computations: $\text{HD}(X,Y_1)$, $\text{HD}(X,Y_2)$, ...:
  Same number of OTs, but on longer strings

- Can use correlated OT (C-OT) to improve communication

- Generalize to larger class of functions $f(X,Y) = f_X(X) + f_Y(Y) + \Sigma f_i(x_i,Y)$
  - Hamming Distance: $f_X=f_Y=0$, $f_i(x_i,Y)=x_i \oplus y_i$
  - Squared Euclidean Distance (for faces & fingerprints):
    $f_X(X) = \Sigma x_i^2$, $f_Y(Y) = \Sigma y_i^2$, $f_i(x_i,Y) = -2x_iy_i$
  - Normalized Hamming Distance (for irises) $\sum_{i=1}^{\ell} \frac{m_i m'_i (x_i \oplus y_i)}{\sum_{i=1}^{\ell} (m_i m'_i)}$
  - Squared Mahalanobis Distance
    (for hand shapes, keystrokes, signatures) $(X - Y)^T M (X - Y)$
GSHADE Protocol

Goal: compute \( f(X, Y) = f_X(X) + f_Y(Y) + \Sigma f_i(x_i, Y) \)

choose \( r_i \in \mathbb{R}^\mathbb{Z}_m \)

for \( i=1..\ell \):

\[
\begin{align*}
  r_i + f_i(0, Y); \ r_i + f_i(1, Y) \\
  R = - f_Y(Y) + \Sigma r_i \\
  T = f_X(X) + \Sigma t_i
\end{align*}
\]

Continue with generic MPC from \( T - R = f(X, Y) = \ldots \)
Performance of GSHADE

Compare biometric sample with DB of **5,000** entries.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Distance</th>
<th>Time in s</th>
<th>Communication in MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCiFI</td>
<td>Hamming</td>
<td>1.0</td>
<td>6.2</td>
</tr>
<tr>
<td>Eigenfaces</td>
<td>Euclidean</td>
<td>5.0</td>
<td>83.6</td>
</tr>
<tr>
<td>FingerCodes</td>
<td>Euclidean</td>
<td>6.7</td>
<td>67.5</td>
</tr>
<tr>
<td>IrisCodes</td>
<td>Normalized Hamming</td>
<td>9.1</td>
<td>56.4</td>
</tr>
</tbody>
</table>
Performance for SCiFI

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime in s</td>
<td>244.0</td>
<td>42.9</td>
<td>46</td>
<td>9.9</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.5</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>DB</td>
<td>=100</td>
<td></td>
<td>DB</td>
</tr>
<tr>
<td></td>
<td>8.8</td>
<td>10.5</td>
<td>9.9</td>
<td>63.4</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>Communication in MB</td>
<td>7.3</td>
<td>2.6</td>
<td>8.3</td>
<td>8.3</td>
</tr>
<tr>
<td></td>
<td>1.7</td>
<td>5.7</td>
<td>2.6</td>
<td>7.3</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

IDBI=100        IDBI=320  IDBI=50,000
Performance for Eigenfaces

- HE [EFG+09]
- HE+GC [HKS+10]
- GMW [SZ13]
- GSHADE+GMW [BCF+14]

Runtime in s

<table>
<thead>
<tr>
<th>DB</th>
<th>HE</th>
<th>HE+GC</th>
<th>GMW</th>
<th>GSHADE+GMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>320</td>
<td>40.0</td>
<td>79.6</td>
<td>139.6</td>
<td>1.3</td>
</tr>
<tr>
<td>1,000</td>
<td>17.7</td>
<td>26.3</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>

Communication in MB

<table>
<thead>
<tr>
<th>DB</th>
<th>HE</th>
<th>HE+GC</th>
<th>GMW</th>
<th>GSHADE+GMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>320</td>
<td>7.3</td>
<td>9.2</td>
<td>7.7</td>
<td>17</td>
</tr>
<tr>
<td>1,000</td>
<td>9.4</td>
<td>9.4</td>
<td>9.4</td>
<td>446</td>
</tr>
</tbody>
</table>
Performance for Iriscodes

- HE+GC [BG11]
- GSHADE+GMW [BCF+14]

**Runtime in s**
- |DB|=320: HE+GC 17.6, GSHADE+GMW 0.5
- |DB|=10,000: HE+GC 212.6, GSHADE+GMW 17.2

**Communication in MB**
- |DB|=320: HE+GC 87.5, GSHADE+GMW 37.6
- |DB|=10,000: HE+GC 1.7, GSHADE+GMW 4.9
Performance for Fingercodes

<table>
<thead>
<tr>
<th>DB | Communication in MB</th>
<th>DB | Runtime in s</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>13.8, 1.8</td>
</tr>
<tr>
<td>1,024</td>
<td>17.5, 2.2</td>
</tr>
</tbody>
</table>

- HE+GC [HMEK11]
- GSHADE+GMW [BCF+14]
Summary

Conclusion

- OT is very efficient due to OT extensions
- Applications can be built efficiently directly on OT

Future Work

- Further optimize *communication* of OT / secure computation
- Other applications based directly on OT / GSHADE for other distances
- Extend to stronger adversary models
Thanks for your attention.

Questions?

Contact: http://encrypto.de