Efficient Privacy-Preserving Biometric Identification

Yan Huang    Lior Malka    David Evans    Jonathan Katz

http://www.mightbeevil.org/secure-biometrics/

Feb 9, 2011
Motivating Scenario: Private No-Fly Checking
Threat Models

- *Semi-honest* adversary
  - Must follow the protocol correctly
- *Malicious* adversary
  - Can deviate arbitrarily from the protocol

In both threat models, an adversary attempts to break either the *correctness* or the *privacy* property of the protocol.
Threat Models

- **Semi-honest adversary**
  - Must follow the protocol correctly

- **Malicious adversary**
  - Can deviate arbitrarily from the protocol

In both threat models, an adversary attempts to break either the *correctness* or the *privacy* property of the protocol.
Filterbank-based Fingerprint Recognition [Jain et al., 2000]

Also used by Barni et al. [2010].
Non-private Protocol

\[ V = \{ v_1, v_2, \cdots, v_M \} \]

\[ v' = \{ v'_1, v'_2, \cdots, v'_N \} \]

Euclidean Distance

\[ d = \{ d_1, d_2, \cdots, d_M \} \]

Finding Minimum

\[ d^* = \min_{1 \leq i \leq M} (d_i) \]

Retrieve Identity

Record\((i^*)\), if \(d^* = d_{i^*} < \varepsilon\); ⊥, otherwise.
Privacy-preserving Protocol

\[ V = \{ v_1, v_2, \ldots, v_M \} \]

\[ v' = \{ v'_1, v'_2, \ldots, v'_N \} \]

Euclidean Distance

Finding Minimum

Retrieve Identity

Homomorphic Encryption

Garbled Circuits

Backtracking Protocol

Record\((i^*)\), if \(d^* = d_{i^*} < \varepsilon\); \(\bot\), otherwise.
Privacy-preserving Protocol

\[ V = \{v_1, v_2, \ldots, v_M\} \]

\[ v' = \{v'_1, v'_2, \ldots, v'_N\} \]

Euclidean Distance

Finding Minimum

\[ d = \{d_1, d_2, \ldots, d_M\} \]

\[ d^* = \min_{1 \leq i \leq M} (d_i) \]

Retrieve Identity

Record\((i^*)\), if \(d^* = d_{i^*} < \varepsilon\); \(\bot\), otherwise.
Euclidean Distance

Let \( d_i \) be the distance between \( \mathbf{v}_i = [v_{i,j}]_{1 \leq j \leq N} \) and \( \mathbf{v}' = [v'_j]_{1 \leq j \leq N} \)

\[
d_i = \| \mathbf{v}_i - \mathbf{v}' \|^2 = \sum_{j=1}^{N} (v_{i,j} - v'_j)^2
\]

\[
= \sum_{j=1}^{N} v_{i,j}^2 + \sum_{j=1}^{N} (-2v_{i,j} \cdot v'_j) + \sum_{j=1}^{N} v'_j^2
\]

For privacy, want to compute \([d_i]_{pk}\).
Additive Homomorphic Encryption

\[
\begin{align*}
[a]_{pk} \quad & \quad \Rightarrow \quad [a + b \mod p]_{pk} = [a]_{pk} \cdot [b]_{pk} \\
[b]_{pk} \quad & \\
\end{align*}
\]

\[
\begin{align*}
[a]_{pk} \quad & \quad \Rightarrow \quad [c \cdot a \mod p]_{pk} = [a]_c^{\text{ pk}} \\
\multicolumn{2}{c}{c} \\
\end{align*}
\]

We used Paillier cryptosystem [Catalano et al., 2001, Paillier, 1999] in our prototype.
Additive Homomorphic Encryption

\[
\begin{align*}
[a] & \quad \Rightarrow [a + b \mod p] = [a] \cdot [b] \\
[b] & \\
\end{align*}
\]

\[
\begin{align*}
[a] & \quad \Rightarrow [c \cdot a \mod p] = [a]^c \\
c & \\
\end{align*}
\]

We used Paillier cryptosystem [Catalano et al., 2001, Paillier, 1999] in our prototype.
Private Euclidean Distance

$$\|d_i\| = \begin{bmatrix} \sum_{j=1}^{N} v_{i,j}^2 + \sum_{j=1}^{N} (-2v_{i,j}v'_j) + \sum_{j=1}^{N} v'^2_j \end{bmatrix}$$

$$= [S_{i,1}] \cdot [S_{i,2}] \cdot [S_3]$$

$$[S_{i,2}] = \begin{bmatrix} \sum_{j=1}^{N} (-2v_{i,j}v'_j) \end{bmatrix} = \prod_{j=1}^{N} \left[ -2v_{i,j} \right] v'_j$$
Improving the Efficiency

- **Modular exponentiation is slow.** For every $i$, computing $[S_{i,2}]$ requires $N$ modular exponentiations. Overall, it involves $MN$ modular exponentiations.

- **Encode many messages in one homomorphic encryption**

\[
\begin{align*}
\{a_1, a_2, a_3, a_4\} & \rightarrow \{a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4\}
\end{align*}
\]

Packing was introduced by Sadeghi et al. [2009] to save bandwidth, but is exploited more aggressively here to save computation also.
Padding 0’s to Ensure Correctness

\[
\begin{align*}
51,28,72 & \quad 51,28,72 \\
+ & \quad 39,92,22 \\
\quad & 39,92,22 \\
\hline
91,20,94 & \quad 91,20,94
\end{align*}
\]

\[
\begin{align*}
051,028,072 & \quad 051,028,072 \\
+ & \quad 039,092,022 \\
\quad & 039,092,022 \\
\hline
090,120,094 & \quad 090,120,094
\end{align*}
\]
Vertical Partitioning to Speedup Computing $[S_{i,2}]$

$$[S_{i,2}] = \prod_{j=1}^{N} \left[ -2v_{i,j} \right]^{v'_j}$$

$$\begin{bmatrix}
-2v_{1,1} & -2v_{1,2} & \cdots & -2v_{1,N} \\
-2v_{2,1} & -2v_{2,2} & \cdots & -2v_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
-2v_{\kappa,1} & -2v_{\kappa,2} & \cdots & -2v_{\kappa,N}
\end{bmatrix}$$
Vertical Partitioning to Speedup Computing $[S_{i,2}]$

$$[S_{i,2}] = \prod_{j=1}^{N} \left[ -2v_{i,j} \right]^{v_j'}$$

$$[S_{1,2} \parallel S_{2,2} \parallel \cdots \parallel S_{\kappa,2}] = \prod_{1 \leq j \leq N} \left[ -2v_{1,j}v_j' \parallel -2v_{2,j}v_j' \parallel \cdots \parallel -2v_{\kappa,j}v_j' \right]$$

$$\begin{bmatrix}
-2v_{1,1} & -2v_{1,2} & \cdots & -2v_{1,N} \\
-2v_{2,1} & -2v_{2,2} & \cdots & -2v_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
-2v_{\kappa,1} & -2v_{\kappa,2} & \cdots & -2v_{\kappa,N}
\end{bmatrix}$$
Vertical Partitioning to Speedup Computing \([S_{i,2}]\)

\[
[S_{i,2}] = \prod_{j=1}^{N} \left[ -2v_{i,j} \right]^{v'_j}
\]

\[
[S_{1,2}||S_{2,2}|| \cdots ||S_{\kappa,2}] = \prod_{1 \leq j \leq N} \left[ -2v_{1,j}v'_j \| -2v_{2,j}v'_j \| \cdots \| -2v_{\kappa,j}v'_j \right]
\]

\[
\left[ -2v_{1,j}v'_j \| -2v_{2,j}v'_j \| \cdots \| -2v_{\kappa,j}v'_j \right] = \left[ -2v_{1,j} \| -2v_{2,j} \| \cdots \| -2v_{\kappa,j} \right]^{v'_j}
\]

\[
\begin{bmatrix}
-2v_{1,1} & -2v_{1,2} & \cdots & -2v_{1,N} \\
-2v_{2,1} & -2v_{2,2} & \cdots & -2v_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
-2v_{\kappa,1} & -2v_{\kappa,2} & \cdots & -2v_{\kappa,N}
\end{bmatrix}
\]
Vertical Partitioning to Speedup Computing $[S_{i,2}]$

$$[S_{i,2}] = \prod_{j=1}^{N} [-2v_{i,j}] v_j'$$

$$[S_{1,2}||S_{2,2}|| \cdots || S_{\kappa,2}] = \prod_{1 \leq j \leq N} \left[ [-2v_{1,j}v_j' || -2v_{2,j}v_j' || \cdots || -2v_{\kappa,j}v_j'] \right]$$

$$\left[ -2v_{1,j}v_j' || -2v_{2,j}v_j' || \cdots || -2v_{\kappa,j}v_j' \right] = \left[ -2v_{1,j} || -2v_{2,j} || \cdots || -2v_{\kappa,j} \right] v_j'$$

\[
\begin{bmatrix}
-2v_{1,1} & -2v_{1,2} & \cdots & -2v_{1,N} \\
-2v_{2,1} & -2v_{2,2} & \cdots & -2v_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
-2v_{\kappa,1} & -2v_{\kappa,2} & \cdots & -2v_{\kappa,N}
\end{bmatrix}
\]
Effects of Packing

![Graph showing the relationship between Paillier Encryption Security Parameter and number of times more efficient for Time and Bandwidth.]
Sharing the Secrets

The server generates nonce masks \( r = [r_1, r_2, \ldots, r_M] \) and sends

\[
[d'_1 \| \cdots \| d'_M]_{pk} = [(d_1 + r_1) \| (d_2 + r_2) \| \cdots \| (d_M + r_M)]_{pk}
\]

where \( pk \) is the client’s public key.

Make the sampling range of \( r_i \) large enough so that \( d'_i \) and \( d_i \) is statistically indistinguishable.
Privacy-preserving Protocol

\[ V = \{v_1, v_2, \ldots, v_M\} \]

\[ v' = \{v'_1, v'_2, \ldots, v'_N\} \]

\[ d' = \{d'_1, d'_2, \ldots, d'_M\} \]

\[ r = \{r_1, r_2, \ldots, r_M\} \]

\[ d^* = \min_{1 \leq i \leq M} (d_i) \]

**Finding Minimum**

**Euclidean Distance**

**Retrieve Identity**

**Garbled Circuits**

\[ \text{Record}(i^*), \text{if } d^* = d_{i^*} < \varepsilon; \quad \bot, \text{otherwise.} \]
Efficient oblivious transfer protocol combining schemes from both [Naor and Pinkas, 2001] and [Ishai et al., 2003]

Standard garbled circuits [Yao, 1986] combined with free-XOR technique [Kolesnikov and Schneider, 2008]
Finding the Minimum Difference

Goal

Given \( d' = d + r \) and \( r \), securely compute \( d^* = \min_{1 \leq i \leq M} (d_i, \epsilon) \).
Reducing the Bit-width

\[ 2M(\ell - k) \text{ non-free gates in total.} \]
Privacy-preserving Protocol

\[ V = \{v_1, v_2, \ldots, v_M\} \]

\[ v' = \{v'_1, v'_2, \ldots, v'_N\} \]

**Euclidean Distance**

**Finding Minimum**

\[ d^* = \min_{1 \leq i \leq M} (d_i) \]

**Retrieve Identity**

**Backtracking Protocol**

**Record** \( i^* \), if \( d^* = d_{i^*} < \varepsilon \);
\( \perp \), otherwise.
Finding the Record

- Ultimate goal is to retrieve the record associated with $d^*$
- Prior work [Kolesnikov et al., 2009] accomplished this by relaying indices throughout the \( M\text{-to-}1 \) Min circuit
- We achieve this with a \textit{backtracking} protocol
  1. No need to propagate ID numbers
  2. Obtain record without an extra secure information retrieval by ID
  3. Use labels obtained in garbled circuit execution
The 2-to-1 Min

$\lambda_i^0 / \lambda_i^1$

MUX

GT

2-to-1 Min

$k$ bits

1 bit

$k$ bits

$k$ bits
Mini Example — The Server

“Radu”  “Adrian”  “Doug”  “Yan”
Mini Example — The Server

Random Permutation
Selection Wires in the $M$-to-1 Min Tree
Backtracking — The Sender

\[
\text{Enc}_{n_2, \lambda_2^0}(\text{“Yan”}) \quad \text{Enc}_{n_2, \lambda_2^1}(\text{“Radu”}) \quad \text{Enc}_{n_3, \lambda_3^0}(\text{“Adrian”}) \quad \text{Enc}_{n_3, \lambda_3^1}(\text{“Doug”})
\]

\[
\text{Enc}_{n_1, \lambda_1^0}(n_2) \quad \text{Enc}_{n_1, \lambda_1^1}(n_3)
\]

\[
\text{Enc}_{\lambda}(n_1)
\]

\[n_1, n_2, n_3\text{ are random nonces known only to the sender.}\]
Backtracking — The Receiver

$d_{Yan}$

$d_{Radu}$

$k$ bits

$\lambda_2^0/\lambda_2^1$

$2$-to-$1$ Min

$d_{Adrian}$

$k$ bits

$\lambda_3^0/\lambda_3^1$

$2$-to-$1$ Min

$d_{Doug}$

$k$ bits

$\lambda_e^0/\lambda_e^1$

$2$-to-$1$ Min

$\varepsilon$
Backtracking — The Receiver

Client knows $\lambda_0^{\varepsilon}, \lambda_0^1, \lambda_2^1, \lambda_3^0$ from circuit evaluation,
Client knows $\lambda_\epsilon^0, \lambda_1^0, \lambda_2^1, \lambda_3^0$ from circuit evaluation, so is able to infer $n_1$
Backtracking — The Receiver

Client knows $\lambda_0^0, \lambda_1^0, \lambda_2^1, \lambda_3^0$ from circuit evaluation, so is able to infer $n_1, n_2$
Client knows $\lambda_0^0, \lambda_2^1, \lambda_2^0, \lambda_3^0$ from circuit evaluation, so is able to infer $n_1, n_2$, and Radu.
System Recap

\[ V = \{v_1, v_2, \ldots, v_M\} \]

\[ v' = \{v'_1, v'_2, \ldots, v'_N\} \]

Euclidean Distance

Distance

Finding Minimum

OT Circuit

Retrieve Identity

Backtracking

\[ d = \{d_1, d_2, \ldots, d_M\} \]

\[ d^* = \min_{1 \leq i \leq M} (d_i) \]

\[ \text{Record}(i^*), \text{if} \ d^* = d_{i^*} < \varepsilon; \]
\[ \bot, \text{otherwise.} \]
4.6× faster and uses 58% less bandwidth than Barni et al. [2010], even though we compute the global minimum
Thank you!

Software available for download at:

http://www.mightbeevil.org/secure-biometrics/
References I


