Privacy-Preservig Biometric Authentication
Inner Product Enctyption

by Artem Tsvik

February 2020

Task 1

Elliptic Curve Math

- Implemented base operations on Elliptic Curve
- Checked bilinearity property of ate pairing

$$(r_1, r_2, r_3) \leftarrow \mathbb{Z}_{2^{256}}$$

$$(T_1, T_2, T_3) = (G_1^{r_1}, G_1^{r_2}, G_1^{r_3}),\ (T_4, T_5, T_6) = (G_2^{r_1}, G_2^{r_2}, G_2^{r_3})$$

$$TP = T_1 + T_2 - T_3,\ TQ = T_4 + T_5 - T_6,\ tf = r_1 + r_2 - r_3$$

in result

$$e(G_1, G_2)^{tf^2} = e(TP, TQ)$$

Task 2

Hamming Distance via Inner Product

- Created "hamming_distance" function
- Generated 2 vectors $(v_1, v_2) \leftarrow \{-1, 1\}^{16}$:
  $v_1 = [-1, -1, -1, 1, -1, -1, 1, 1, -1, 1, 1, 1, 1, 1, -1, -1]$
  $v_2 = [1, 1, -1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, -1]$
- and calculated hamming distance between v_1 and v_2
  result : 0.625

Task 3

Inner Product Encryption

- Generated 2 vectors v_1, v_2
- Implemented main functions :
  - Master key generation
  - Decryption key generation
  - Encryption
  - Decryption

Task 3

Inner Product Encryption (Continue)

Python and C++ runtime table for the functions at 1.6GHz processor running Windows 10 with 4GB of memory.

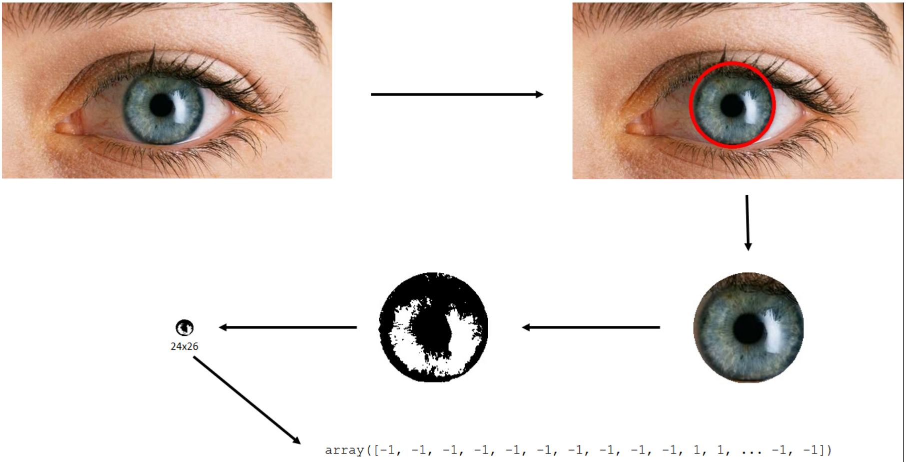
	Python	C++
Master key generation	9.95s	226ms
Decryption key generation	12.14s	210ms
Encryption	3.37s	106ms
Decryption	12.93s	239ms

Task 4

- Used library OpenCV

- Implemented functions for highlighting iris on some photos and convert it to bio vector

Task 4



Task 5*

- Master key generation (run on Device)

CIPE.Setup

$\vec{s} = (s_1, \ldots, s_n) \leftarrow \mathbb{Z}_q^n$
$\vec{t} = (t_1, \ldots, t_n) \leftarrow \mathbb{Z}_q^n$
$\vec{u} = (u_1, \ldots, u_{n+2}) \leftarrow \mathbb{Z}_q^{n+2}$
$\vec{v} = (v_1, \ldots, v_{n+2}) \leftarrow \mathbb{Z}_q^{n+2}$
For $i = 1$ to n do
$gen1\_h_i = (g^{s_i}, h^{t_i})$
For $i = 1$ to $n + 2$ do
$gen2\_h_i = (\overline{g}^{u_i}, \overline{h}^{v_i})$
$msk = (\{gen1\_h_i, s_i, t_i\}_{i=1}^{n}, \{gen2\_h_i, u_i, v_i\}_{i=1}^{n+2})$

return msk

Task 5*

- Decryption key generation (run on Device during registration)

$\textbf{CIPE.Conv.DK}(msk, \overline{c} = (c_1, \ldots, c_n))$
$(r_0, r_1) \leftarrow \mathbb{Z}_q^2$
For i to $n+2$ do
$\overline{h_i} = \overline{g}^{u_i r_0} \overline{h}^{v_i r_1}$
$d_{0,1} = \overline{g}^{r_0}$, $d_{0,2} = \overline{h}^{r_1}$
$d_{0,3} = (\prod_{j=1}^{n} c_j^{-s_j}) \cdot \overline{h}_1$, $d_{0,4} = (\prod_{j=1}^{n} c_j^{-t_j}) \cdot \overline{h}_2$
For i to n do
$d_{0,i+4} = c_i \cdot \overline{h}_{i+2}$
$DK = (d_{0,1}, \ldots, d_{0,n+4})$
return DK

Task 5*

- Encryption (run on Device during authentication)

$$\textbf{CIPE.Conv.Enc}(msk, \overline{c} = (c_1, ..., c_n))$$

$$(r_2, r_3) \leftarrow \mathbb{Z}_q^2$$

$$c_{0,3} = g^{r_2}, \; c_{0,4} = h^{r_3}$$

For i to n do

$$c_{0,i+4} = c_i \cdot g^{s_i r_2} \cdot h^{t_i r_3}$$

$$c_{0,1} = (\textstyle\prod_{j=3}^{n+4} c_{0,j}^{-u_j}), \; c_{0,2} = (\textstyle\prod_{j=3}^{n+4} c_{0,j}^{-v_j})$$

$$CT = (c_{0,1}, ..., c_{0,n+4})$$

return CT

Task 5*

	Python	C++
Original		
Master key generation	9.95s	226ms
Decryption key generation	12.14s	210ms
Encryption	3.37s	106ms
Decryption	12.93s	239ms
Modified		
Master key generation	9.15s	217ms
Decryption key generation	15.19s	258ms
Encryption	3.89s	128ms
Decryption	12.26s	210ms

Thank You