Biometrics Authentication: Formalization and Instantiation

Keng-Yu Chen

September 30, 2024

This report formalizes the biometric authentication scheme, including its structure, usage, and security analysis with a security game model.

1 Preliminaries

In this report, we assume

- λ is the security parameter.
- $[m]$ denotes the set of integers $\{1, 2, \cdots, m\}$.
- $\mathbb{Z}_q$ is the finite field modulo a prime number q.
- A function $f(n)$ is called *negligible* iff for any integer c, $f(n) < \frac{1}{n^c}$ for all sufficiently large n. We write it as $f(n) = \mathsf{negl}$, and we may also use negl to represent an arbitrary negligible function.
- poly is the class of polynomial funcions. We may also use poly to represent an arbitrary polynomial function.
- We write sampling a value r from a distribution $\mathcal{D}$ as $r \leftarrow\!\!\$\ \mathcal{D}$. If S is a finite set, then $r \leftarrow\!\!\$\ S$ means sampling r uniformly from S.
- The distribution $\mathcal{D}^t$ denotes t identical and independent distributions of $\mathcal{D}$.
- A PPT algorithm denotes a probabilistic polynomial time algorithm. Unless otherwise specified, all algorithms run in PPT.

Definition 1 (Functional Hiding Inner Product Functional Encryption)**.** A *functional hiding inner product functional encryption* (fh-IPFE) scheme FE for a field $\mathbb{F}$ and input length k is composed of PPT algorithms $\mathsf{FE.Setup}$, $\mathsf{FE.KeyGen}$, $\mathsf{FE.Enc}$, and $\mathsf{FE.Dec}$:

- $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{msk}, \mathsf{pp}$: It outputs the public parameter pp and the master secret key msk.

- $\mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pp}, \mathbf{x}) \to f_{\mathbf{x}}$: It generates the functional decryption key $f_{\mathbf{x}}$ for an input vector $\mathbf{x} \in \mathbb{F}^k$.
- $\mathsf{FE.Enc}(\mathsf{msk}, \mathsf{pp}, \mathbf{y}) \to \mathbf{c_y}$: It encrypts the input vector $\mathbf{y} \in \mathbb{F}^k$ to the ciphertext $\mathbf{c_y}$.
- $\mathsf{FE.Dec}(\mathsf{pp}, f_{\mathbf{x}}, \mathbf{c_y}) \to z \in \mathbb{F}$: It outputs a value z.

Correctness: The fh-IPFE scheme FE is *correct* if $\forall \mathsf{msk}, \mathsf{pp} \leftarrow \mathsf{FE.Setup}$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{F}^k$, we have

$$\mathsf{FE.Dec}(\mathsf{pp}, \mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pp}, \mathbf{x}), \mathsf{FE.Enc}(\mathsf{msk}, \mathsf{pp}, \mathbf{y})) = \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{F}.$$

2 Formalization

In general, an authentication shceme Π associated with a family of biometric distributions $\mathbb{B}$ is composed of the following algorithms.

- $\mathsf{USetup}(1^\lambda) \to \mathsf{esk}, \mathsf{psk}, \mathsf{csk}$: It outputs the enrollment secret key esk, probe secret key psk, and compare secret key csk.
- $\mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}() \to \mathbf{x}$: Given an oracle $\mathcal{O}_{\mathcal{B}}$, which samples biometric data from the distribution $\mathcal{B} \in \mathbb{B}$, it encodes biometric samples as $\mathbf{x}$, the input format for enrollment.
- $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x}) \to \mathbf{c_x}$: It outputs the enrollment message $\mathbf{c_x}$ from $\mathbf{x}$.
- $\mathsf{encodeProbe}^{\mathcal{O}_{\mathcal{B}}}() \to \mathbf{y}$: Given an oracle $\mathcal{O}_{\mathcal{B}}$, which samples biometric data from the distribution $\mathcal{B} \in \mathbb{B}$, it encodes biometric samples as $\mathbf{y}$, the input format for probe.
- $\mathsf{Probe}(\mathsf{psk}, \mathbf{y}) \to \mathbf{c_y}$: It outputs the probe message $\mathbf{c_y}$ from $\mathbf{y}$.
- $\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y}) \to s$: It compares the enrollment message $\mathbf{c_x}$ and probe message $\mathbf{c_y}$ and outputs a score s.
- $\mathsf{Verify}(s) \to r \in \{0, 1\}$: It is a deterministic algorithm that reads the comparison score s and determines whether this is a successful authentication ($r = 1$) or not ($r = 0$).

The usage model we consider is described in Figure 1. We assume the user's biometric distribution is $\mathcal{B} \in \mathbb{B}$.

For example, let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be an fh-IPFE scheme we defined in Definition 1. Following [EM23], we can instantiate a biometric authentication scheme using FE with the distance metric the Euclidean distance. Let the biometric templates $\mathbf{b}$ and $\mathbf{b}'$ be sampled from some distribution $\mathcal{B} \subseteq [m]^k$, and let the associated field of FE be $\mathbb{Z}_q$ where q is a prime number larger than the maximum possible Euclidean distance $m^2 \cdot k$. The scheme is instantiated as follows.

User		**Server**
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{USetup}(1^\lambda)$		
$\mathbf{b} \leftarrow_\$ \mathcal{B}, \mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$		
$\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$		
	$\xrightarrow{\mathsf{csk}, \mathbf{c_x}}$	
$\mathbf{b}' \leftarrow_\$ \mathcal{B}, \mathbf{y} \leftarrow \mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$		
$\mathbf{c_y} \leftarrow \mathsf{Probe}(\mathsf{psk}, \mathbf{y})$		
	$\xrightarrow{\mathbf{c_y}}$	
		$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$
		$r \leftarrow \mathsf{Verify}(s)$
	$\xleftarrow{r}$	

Figure 1: Authentication Model with User and Server

- $\mathsf{USetup}(1^\lambda)$: It calls $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{msk}, \mathsf{pp}$ and outputs $(\mathsf{esk}, \mathsf{psk}) \leftarrow ((\mathsf{msk}, \mathsf{pp}), (\mathsf{msk}, \mathsf{pp}))$ and $\mathsf{csk} \leftarrow \mathsf{pp}$.
- $\mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}() \to \mathbf{x}$: For a template vector $\mathbf{b} = (b_1, b_2, \cdots, b_k)$ sampled from $\mathcal{O}_\mathcal{B}$, the function encodes it as $\mathbf{x} = (x_1, x_2, \cdots, x_{k+2}) = (b_1, b_2, \cdots, b_k, 1, \|\mathbf{b}\|^2)$. The auxiliary oracle is empty.
- $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$: It calls $\mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pp}, \mathbf{x}) \to f_\mathbf{x}$ and outputs $\mathbf{c_x} \leftarrow f_\mathbf{x}$.
- $\mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$: For a template vector $\mathbf{b}' = (b'_1, b'_2, \cdots, b'_k)$ sampled from $\mathcal{O}_\mathcal{B}$, the function encodes it as $\mathbf{y} = (y_1, y_2, \cdots, y_{k+2}) = (-2b'_1, -2b'_2, \cdots, -2b'_k, \|\mathbf{b}'\|^2, 1)$. The auxiliary oracle is empty.
- $\mathsf{Probe}(\mathsf{psk}, \mathbf{y})$: It calls $\mathsf{FE.Enc}(\mathsf{msk}, \mathsf{pp}, \mathbf{y}) \to \mathbf{c_y}$ and outputs $\mathbf{c_y}$.
- $\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$: It calls $\mathsf{FE.Dec}(\mathsf{pp}, \mathbf{c_x}, \mathbf{c_y}) \to s$ and outputs the value s.
- $\mathsf{Verify}(s)$: If $\sqrt{s} < \tau$, a pre-defined threshold for comparing the closeness of two templates, then it outputs $r = 1$; otherwise, it outputs $r = 0$.

By the correctness of the functional encryption scheme FE, we have

$$s = \mathsf{FE.Dec}(\mathsf{pp}, \mathbf{c_x}, \mathbf{c_y}) = \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{k} -2b_i b'_i + \|\mathbf{b}\|^2 + \|\mathbf{b}'\|^2 = \|\mathbf{b} - \mathbf{b}'\|^2.$$

which is the square of the Euclidean distance between two templates $\mathbf{b}$ and $\mathbf{b}'$. Therefore, if two templates $\mathbf{b}$ and $\mathbf{b}'$ are close enough such that $\|\mathbf{b} - \mathbf{b}'\| < \tau$, the scheme results in $r = 1$, a successful authentication.

3 Security Games

From now on, we consider a family of biometric distributions $\mathbb{B} = \{\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_N\}$, representing N people. Removing a person $\mathcal{B}$ from $\mathbb{B}$ is written as $\mathbb{B} \setminus \mathcal{B}$.

3.1 Forgery Game

In the forgery game, we model the ability of an adversary who has access to the server's database of registered enrollments and tries to forge the user. The adversary $\mathcal{A}$ is given the enrollment message $\mathbf{c_x}$ and oracle $\mathcal{O}$ and tries to find a valid probe message $\tilde{\mathbf{z}}$. The whole game is defined in Figure 2.

$\underline{\mathsf{Forg}_{\Pi,\mathbb{B}}(\mathcal{A}^{\mathcal{O}})}$
$\mathcal{B} \leftarrow_{\$} \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{USetup}(1^\lambda)$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$
$\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
$\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}}(\mathbf{c_x})$
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
return $\mathsf{Verify}(s)$

Figure 2: The Forgery Game

$\underline{\mathsf{Forg'}_{\Pi,\mathbb{B}}(\mathcal{A'}^{\mathcal{O}})}$
$\mathcal{B} \leftarrow_{\$} \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{USetup}(1^\lambda)$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$
$\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
$\tilde{\mathbf{z}} \leftarrow \mathcal{A'}^{\mathcal{O}}()$
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
return $\mathsf{Verify}(s)$

Figure 3: The Plain Forgery Game

The given oracle $\mathcal{O}$ can be any or more of the following three oracles:

- $\mathcal{O}_{\mathsf{samp}}(\cdot)$: On input an index i, it outputs a biometric sample $\mathbf{b}$ sampled from the distribution $\mathcal{B}_i \in \mathbb{B}$.
- $\mathcal{O}_{\mathsf{Enroll}}(\mathsf{esk}, \cdot)$: On input $\mathbf{x}$, it outputs the enrollment message $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$.
- $\mathcal{O}_{\mathsf{Probe}}(\mathsf{psk}, \cdot)$: On input $\mathbf{y}$, it outputs the probe message $\mathsf{Probe}(\mathsf{psk}, \mathbf{y})$.
- $\mathcal{O}_{\mathsf{Compare}}(\mathsf{csk}, \cdot, \cdot)$: On input $\mathbf{c_x}$ and $\mathbf{c_y}$, it outputs the comparison $\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$.

Note that if the enrollment secret key esk, probe secret key psk, or the comparison secret key csk is an empty or public string in the scheme, then the corresponding oracles are naturally and implicitly given since the adversary can compute them herself.

To consider potential false positives of biometrics match, we consider the plain forgery game in Figure 3, in which the adversary does not have any knowledge about the template.

We define the advantage of an adversary $\mathcal{A}$ in the forgery game of a scheme Π associated with a family of distributions $\mathbb{B}$ as

$$\mathbf{Adv}^{\mathsf{Forg}}_{\Pi,\mathbb{B},\mathcal{A}^{\mathcal{O}}} := \Pr[\mathsf{Forg}_{\Pi,\mathbb{B}}(\mathcal{A}^{\mathcal{O}}) \to 1] - \max_{\text{PPT } \mathcal{A}'} \Pr[\mathsf{Forg'}_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1].$$

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *forgery secure* if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{Forg}}_{\Pi,\mathbb{B},\mathcal{A}^{\mathcal{O}}} = \mathsf{negl}.$$

3.2 Simulation Game

In the simulation game, we model the ability of the server who tries to learn something more than the comparison result of the enrollment and probe messages. The adversary $\mathcal{A}$ is given an enrollment message and a list of t probe messages, and she needs to guess whether these are real messages or simulation results of a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_{\mathsf{Enroll}}, \mathcal{S}_{\mathsf{Probe}})$ based on the $\mathsf{Compare}$ function. Intuitively, the simulator receives a list of $\mathsf{Compare}$ results of real enrollment and probe messages, and it returns some manual enrollment or probe messages that look similar to real ones. The whole game is defined in Figure 4.

$$
\begin{array}{l}
\underline{\mathsf{SIM-Real}_{\Pi,\mathbb{B}}(\mathcal{A}^{\mathcal{O}})} \\
\mathcal{B} \leftarrow_{\$} \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B} \\
\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{USetup}(1^{\lambda}) \\
\mathbf{b} \leftarrow_{\$} \mathcal{B}, \mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}() \\
\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x}) \\
\{\mathbf{y}^{(i)}\}_{i=1}^{t} \leftarrow \{\mathsf{encodeProbe}^{\mathcal{O}_{\mathcal{B}}}()\}_{i=1}^{t} \\
\{\mathbf{c_y}^{(i)}\}_{i=1}^{t} \leftarrow \left\{\mathsf{Probe}(\mathsf{psk}, \mathbf{y}^{(i)})\right\}_{i=1}^{t} \\
b \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{csk}, \mathbf{c_x}, \{\mathbf{c_y}^{(i)}\}_{i=1}^{t}) \\
\mathbf{return}\ b
\end{array}
$$

$$
\begin{array}{l}
\underline{\mathsf{SIM-Ideal}_{\Pi,\mathbb{B}}(\mathcal{A}^{\tilde{\mathcal{O}}}, \mathcal{S} = (\mathcal{S}_0, \mathcal{S}_{\mathsf{Enroll}}, \mathcal{S}_{\mathsf{Probe}}))} \\
\mathcal{B} \leftarrow_{\$} \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B} \\
\mathsf{esk}, \mathsf{psk}, \mathsf{csk}, \leftarrow \mathsf{USetup}(1^{\lambda}) \\
\mathbf{b} \leftarrow_{\$} \mathcal{B}, \mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}() \\
\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x}) \\
\{\mathbf{b}^{(i)}\}_{i=1}^{t} \leftarrow_{\$} \mathcal{B}^{t} \\
\{\mathbf{y}^{(i)}\}_{i=1}^{t} \leftarrow \{\mathsf{encodeProbe}^{\mathcal{O}_{\mathcal{B}}}()\}_{i=1}^{t} \\
\{\mathbf{c_y}^{(i)}\}_{i=1}^{t} \leftarrow \left\{\mathsf{Probe}(\mathsf{psk}, \mathbf{y}^{(i)})\right\}_{i=1}^{t} \\
\mathsf{C} \leftarrow \{\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y}^{(i)})\}_{i=1}^{t} \\
\mathbf{c_x}', \{\mathbf{c_y}^{(i)\prime}\}_{i=1}^{t} \leftarrow \mathcal{S}_0(\mathsf{csk}, \mathsf{C}) \\
b \leftarrow \mathcal{A}^{\tilde{\mathcal{O}}}(\mathsf{csk}, \mathbf{c_x}', \{\mathbf{c_y}^{(i)\prime}\}_{i=1}^{t}) \\
\mathbf{return}\ b
\end{array}
$$

Figure 4: The Simulation Game

The oracle $\mathcal{O}$ can be any or more of the following oracles:

- $\mathcal{O}_{\mathsf{samp}}(\cdot)$: On input an index i, it outputs a biometric sample $\mathbf{b}$ sampled from the distribution $\mathcal{B}_i \in \mathbb{B}$.
- $\mathcal{O}_{\mathsf{Enroll}}(\mathsf{esk}, \cdot)$: On input $\mathbf{x}$, it outputs the enrollment message $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$.
- $\mathcal{O}_{\mathsf{Probe}}(\mathsf{psk}, \cdot)$: On input $\mathbf{y}$, it outputs the probe message $\mathsf{Probe}(\mathsf{psk}, \mathbf{y})$.

The oracle $\tilde{\mathcal{O}}$ is stateful and memorizes all the previous queries. It includes two simulators $\mathcal{S}_{\mathsf{Enroll}}, \mathcal{S}_{\mathsf{Probe}}$ and is given by any or more of the following oracles:

- $\mathcal{O}_{\mathsf{samp}}(\cdot)$: On input an index i, it outputs a biometric sample $\mathbf{b}$ sampled from the distribution $\mathcal{B}_i \in \mathbb{B}$.
- $\tilde{\mathcal{O}}_{\mathsf{Enroll}}(\mathsf{csk}, \mathsf{esk}, \cdot)$: On input $\mathbf{x}^{(j)}$, it updates the collection of comparison results C by adding the results with all previous probe messages $\mathbf{c_y}^{(i)}$. Then it calls the simulator $\mathcal{S}_{\mathsf{Enroll}}(\mathsf{csk}, \mathsf{C})$ and returns whatever the simulator returns.
- $\tilde{\mathcal{O}}_{\mathsf{Probe}}(\mathsf{csk}, \mathsf{psk}, \cdot)$: On input $\mathbf{y}^{(j)}$, it updates the collection of comparison results C by adding the results with all previous enrollment messages $\mathbf{c_x}^{(i)}$. Then it calls the simulator $\mathcal{S}_{\mathsf{Probe}}(\mathsf{csk}, \mathsf{C})$ and returns whatever the simulator returns.

$$\begin{array}{l}\tilde{\mathcal{O}}_{\mathsf{Enroll}}(\mathsf{csk}, \mathsf{esk}, \mathbf{x}^{(j)}) \\ \hline \mathbf{c_x}^{(j)} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x}^{(j)}) \\ \mathsf{C} \leftarrow \mathsf{C} \cup \{\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(j)}, \mathbf{c_y}^{(i)})\}_i \\ \mathbf{c_x}' \leftarrow \mathcal{S}_{\mathsf{Enroll}}(\mathsf{csk}, \mathsf{C}) \\ \textbf{return } \mathbf{c_x}'\end{array} \qquad \begin{array}{l}\tilde{\mathcal{O}}_{\mathsf{Probe}}(\mathsf{csk}, \mathsf{psk}, \mathbf{y}^{(j)}) \\ \hline \mathbf{c_y}^{(j)} \leftarrow \mathsf{Probe}(\mathsf{psk}, \mathbf{y}^{(j)}) \\ \mathsf{C} \leftarrow \mathsf{C} \cup \{\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(i)}, \mathbf{c_y}^{(j)})\}_i \\ \mathbf{c_y}' \leftarrow \mathcal{S}_{\mathsf{Probe}}(\mathsf{csk}, \mathsf{C}) \\ \textbf{return } \mathbf{c_y}'\end{array}$$

Figure 5: Choice of the Oracle $\tilde{\mathcal{O}}$

The details of these oracles are given in Figure 5.

We define the advantage of an adversary $\mathcal{A}$ and a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_{\mathsf{Enroll}}, \mathcal{S}_{\mathsf{Probe}})$ in the simulation game of a scheme Π associated with a family of distributions $\mathbb{B}$ as

$$\mathbf{Adv}^{\mathsf{SIM}}_{\Pi,\mathbb{B},\mathcal{A}^{\mathcal{O},\tilde{\mathcal{O}}},\mathcal{S}} := |\Pr[\mathsf{SIM}-\mathsf{Real}_{\Pi,\mathbb{B}}(\mathcal{A}^{\mathcal{O}}) \to 1] - \Pr[\mathsf{SIM}-\mathsf{Ideal}_{\Pi,\mathbb{B}}(\mathcal{A}^{\tilde{\mathcal{O}}}, \mathcal{S}) \to 1]|.$$

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *simulation secure* if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_{\mathsf{Enroll}}, \mathcal{S}_{\mathsf{Probe}})$ such that

$$\mathbf{Adv}^{\mathsf{SIM}}_{\Pi,\mathbb{B},\mathcal{A}^{\mathcal{O},\tilde{\mathcal{O}}},\mathcal{S}} = \mathsf{negl}.$$

3.3 Identification Game

In the identification game, we model the ability of the server who tries to identify the user. The adversary $\mathcal{A}$ is given an enrollment message $\mathbf{c_x}^{(b)}$ and two distinct distributions $\mathcal{B}^{(0)}, \mathcal{B}^{(1)}$ that can be efficiently sampled. She tries to guess from which $\mathbf{c_x}^{(b)}$ is generated. The whole game is defined in Figure 6.

$$\begin{array}{l}\mathsf{Id}_{\Pi,\mathbb{B}}(\mathcal{A}^{\mathcal{O}}) \\ \hline b \leftarrow_{\$} \{0,1\} \\ \mathcal{B}^{(0)}, \mathcal{B}^{(1)} \leftarrow_{\$} \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \{\mathcal{B}^{(0)}, \mathcal{B}^{(1)}\} \\ \mathsf{esk}, \mathsf{psk}, \mathsf{csk}, \leftarrow \mathsf{USetup}(1^\lambda) \\ \mathbf{x}^{(b)} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathbf{B}^{(b)}}}() \\ \mathbf{c_x}^{(b)} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x}^{(b)}) \\ \tilde{b} \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{csk}, \mathbf{c_x}^{(b)}, \mathcal{B}^{(0)}, \mathcal{B}^{(1)}) \\ \textbf{return } 1_{\tilde{b}=b}\end{array}$$

Figure 6: The Identification Game

The given oracle $\mathcal{O}$ can be any or more of the following three oracles:

- $\mathcal{O}_{\mathsf{samp}}(\cdot)$: On input an index i, it outputs a biometric sample $\mathbf{b}$ sampled from the distribution $\mathcal{B}_i \in \mathbb{B}$.
- $\mathcal{O}_{\mathsf{Enroll}}(\mathsf{esk}, \cdot)$: On input $\mathbf{x}$, it outputs the enrollment message $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$.

We define the advantage of an adversary $\mathcal{A}$ in the identification game of a scheme Π associated with a family of distributions $\mathbb{B}$ as

$$\mathbf{Adv}^{\mathsf{Id}}_{\Pi,\mathbb{B},\mathcal{A}^{\mathcal{O}}} := |\Pr[\mathsf{Id}_{\Pi}(\mathcal{A}^{\mathcal{O}}) \to 1] - \frac{1}{2}|.$$

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *identification secure* if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{Id}}_{\Pi,\mathbb{B},\mathcal{A}^{\mathcal{O}}} = \mathsf{negl}.$$

3.4 Reusability Game

In the reusability game, we model the ability of a malicious application who tries to forge the user in another application. The adversay $\mathcal{A}$ is given the enrollment message $\mathbf{c_x}$ and oracle $\mathcal{O}$ and tries to find a valid probe message $\tilde{z}$. The whole game is defined in Figure 7.

$\mathsf{Reuse}_{\Pi,\mathbb{B}}(\mathcal{A}^{\mathcal{O}})$

$\mathcal{B} \leftarrow\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{USetup}(1^\lambda)$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
$\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
$\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}}(\mathbf{c_x})$
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
return $\mathsf{Verify}(s)$

Figure 7: The Reusability Game

$\mathsf{Reuse}'_{\Pi,\mathbb{B}}(\mathcal{A}'^{\mathcal{O}})$

$\mathcal{B} \leftarrow\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{USetup}(1^\lambda)$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
$\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
$\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}}()$
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
return $\mathsf{Verify}(s)$

Figure 8: The Plain Reusability Game

The given oracle $\mathcal{O}$ can be any or more of the following three oracles:

- $\mathcal{O}^{\mathsf{Re}}_{\mathsf{Enroll}}(\cdot,\cdot)$: On input esk and pp, it first samples $\mathbf{x} \leftarrow\$ \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$ and outputs $\mathsf{Enroll}(\mathsf{esk}, \mathsf{pp}, \mathbf{x})$.
- $\mathcal{O}^{\mathsf{Re}}_{\mathsf{Probe}}(\cdot,\cdot)$: On input psk and pp, it first samples $\mathbf{y} \leftarrow\$ \mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$ and outputs $\mathsf{Probe}(\mathsf{psk}, \mathsf{pp}, \mathbf{y})$.

Note that the resuability game is defined in a similar way as the forgery game in Section 3.1. The difference is their use cases and oracles. In the forgery game, the targeted adversary is an eavesdropper who has access to the server's database and the user's device to execute $\mathcal{O}_{\mathsf{Enroll}}$ and $\mathcal{O}_{\mathsf{Probe}}$. If the comparison secret key csk is public, or the comparison function is not restricted, he can also execute $\mathcal{O}_{\mathsf{compare}}$. In contrast, the reusability game targets a malicious application, who can ask the user to enroll and probe with crafted secret keys. We assume the malicious application cannot access other functions in other applications.

4 Analysis of Games

4.1 Id Games $\leq$ Forgery Games

We provide a reduction to show that, given the following assumption, if an adversary $\mathcal{A}_{\mathsf{Forg}}$ can break the forgery game with oracles $\mathcal{O}_{\mathsf{samp}}$, $\mathcal{O}_{\mathsf{Enroll}}$ and $\mathcal{O}_{\mathsf{Compare}}$, then there exists an adversary $\mathcal{A}_{\mathsf{Id}}$ who can break the identification game if she is given the oracle $\mathcal{O}_{\mathsf{samp}}$ and $\mathcal{O}_{\mathsf{Enroll}}$.

Assumption 1. *Let* $\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{USetup}(1^\lambda)$*,* $\mathbf{x}^{(0)}, \mathbf{x}^{(1)} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$*,* ${\mathbf{c}_{\mathbf{x}}}^{(0)} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x}^{(0)})$*,* ${\mathbf{c}_{\mathbf{x}}}^{(1)} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x}^{(1)})$*. Then there exists a constant* τ *such that for all* $\mathbf{z}$ *where*

$$\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, {\mathbf{c}_{\mathbf{x}}}^{(0)}, \mathbf{z})) = 1$$

we have

$$\Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, {\mathbf{c}_{\mathbf{x}}}^{(1)}, \mathbf{z})) = 1] \geq \tau$$

Theorem 1. *Given Assumption* **??** *with the constant* $\tau \geq$*. Let* $\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathcal{O}\mathsf{Forg}}}$ *be an adversay in the Forgery game with oracles* $\mathcal{O}_{\mathsf{Forg}} = (\mathcal{O}'_{\mathsf{samp}}, \mathcal{O}'_{\mathsf{Enroll}}, \mathcal{O}'_{\mathsf{Compare}})$*. Then there exists an adversary* $\mathcal{A}_{\mathsf{Id}}$ *in the identification game with oracles* $\mathcal{O}_{\mathsf{Id}} = (\mathcal{O}_{\mathsf{samp}}, \mathcal{O}_{\mathsf{Enroll}})$ *such that*

$$\mathbf{Adv}_{\Pi,\mathbb{B},\mathcal{A}_{\mathsf{Id}}^{\mathcal{O}_{\mathsf{Id}}}} \geq \frac{1}{4}[\mathbf{Adv}_{\Pi,\mathbb{B},\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}} - 1 - \tau]$$

Proof. Consider the following algorithm and an algorithm $\mathcal{A}_{\mathsf{Forg}}$ who can break the forgery game with

$\mathcal{A}_{\mathsf{Id}}^{\mathcal{O}_{\mathsf{Id}}}(\mathsf{csk}, {\mathbf{c}_{\mathbf{x}}}^{(b)}, \mathcal{B}^{(0)}, \mathcal{B}^{(1)})$

$\mathbf{x}' \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}^{(0)}}}()$
  When $\mathcal{O}_{\mathbf{B}^{(0)}}$ is called, sample $\mathbf{b} \leftarrow \mathcal{B}^{(0)}$ and return $\mathbf{b}$.
$\mathbf{c}'_{\mathbf{x}} \leftarrow \mathcal{O}_{\mathsf{Enroll}}(\mathbf{x}')$
$\tilde{\mathbf{z}} \leftarrow \mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}(\mathbf{c}'_{\mathbf{x}})$
  When $\mathcal{O}'_{\mathsf{samp}}(i)$ is called, return $\mathcal{O}_{\mathsf{samp}}(i)$
  When $\mathcal{O}'_{\mathsf{Enroll}}(\mathbf{x})$ is called, return $\mathcal{O}_{\mathsf{Enroll}}(\mathbf{x})$.
  When $\mathcal{O}'_{\mathsf{Compare}}(\mathbf{c}_{\mathbf{x}}, \mathbf{c}_{\mathbf{y}})$ is called, return $\mathsf{Compare}(\mathsf{csk}, \mathbf{c}_{\mathbf{x}}, \mathbf{c}_{\mathbf{y}})$.
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, {\mathbf{c}_{\mathbf{x}}}^{(b)}, \tilde{\mathbf{z}})$
If $\mathsf{Verify}(s) = 1$
  return 0
Else
  return a random bit $b' \leftarrow_{\$} \{0, 1\}$

Figure 9: The Identification Game Adversary

Note that ${\mathbf{c}_{\mathbf{x}}}'$ and $\mathcal{O}'_{\mathsf{Enroll}}, \mathcal{O}'_{\mathsf{Compare}}$ are simulated perfectly as the real forgery game with disbribution $\mathcal{B}^{(0)}$, so

$$\Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, {\mathbf{c}_{\mathbf{x}}}', \tilde{\mathbf{z}})) \to 1] = \Pr[\mathsf{Forg}_{\Pi,\mathbb{B}}(\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}) \to 1].$$

Therefore,

$$\Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(0)}, \tilde{\mathbf{z}})) \to 1] \geq \Pr[\mathsf{Forg}_{\Pi,\mathbb{B}}(\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}) \to 1] \cdot \tau.$$

Let the challenge bit be b. If $b = 0$, the probability that $\mathcal{A}_{\mathsf{Id}}^{\mathcal{O}_{\mathsf{Enroll}}}$ returns b is

$$\Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(0)}, \tilde{\mathbf{z}})) \to 1] + (1 - \Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(0)}, \tilde{\mathbf{z}})) \to 1]) \cdot \frac{1}{2}$$

$$= \frac{1}{2} + \frac{1}{2}\Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(0)}, \tilde{\mathbf{z}})) \to 1] \geq \frac{1}{2} + \frac{\tau}{2}\Pr[\mathsf{Forg}_{\Pi,\mathbb{B}}(\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}) \to 1].$$

If $b = 1$, the probability that $\mathcal{A}_{\mathsf{Id}}^{\mathcal{O}_{\mathsf{Enroll}}}$ returns b is

$$\begin{aligned}
&\Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(1)}, \tilde{\mathbf{z}})) \to 0] \cdot \frac{1}{2} \\
=&(1 - \Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(1)}, \tilde{\mathbf{z}})) \to 1]) \cdot \frac{1}{2}
\end{aligned}$$

Note that $\mathcal{A}_{\mathsf{Forg}}$ is not given any knowledge about $\mathcal{B}^{(1)}$. The distribution of $\mathbf{c}'_{\mathbf{x}}$ is identical to $\mathbf{c}'_{\mathbf{x}} \leftarrow \mathcal{O}'_{\mathsf{Enroll}}(\mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}'}}())$ where $\mathcal{B}' \leftarrow\!\!\$\ \mathbb{B}$, which she can sample herself in the plain forgery game. Therefore,

$$\Pr[\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}^{(1)}, \tilde{\mathbf{z}})) \to 1] \leq \max_{\text{PPT } \mathcal{A}'} \Pr[\mathsf{Forg'}_{\Pi,\mathbb{B}}(\mathcal{A'}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}) \to 1].$$

In summary,

$$\begin{aligned}
\Pr[\mathcal{A}_{\mathsf{Id}}^{\mathcal{O}_{\mathsf{Id}}} \to b] &= \frac{1}{2}\Pr[\mathcal{A}_{\mathsf{Id}}^{\mathcal{O}_{\mathsf{Id}}} \to 0 \mid b = 0] + \frac{1}{2}\Pr[\mathcal{A}_{\mathsf{Id}}^{\mathcal{O}_{\mathsf{Id}}} \to 1 \mid b = 1] \\
&\geq \frac{1}{4} + \frac{\tau}{4}\Pr[\mathsf{Forg}_{\Pi,\mathbb{B}}(\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}) \to 1] + \frac{1}{4} - \frac{1}{4}\max_{\text{PPT } \mathcal{A}'} \Pr[\mathsf{Forg'}_{\Pi,\mathbb{B}}(\mathcal{A'}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}) \to 1] \\
&= \frac{1}{2} + \frac{1}{4}\mathbf{Adv}_{\Pi,\mathbb{B},\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}} - \frac{1-\tau}{4}\Pr[\mathsf{Forg}_{\Pi,\mathbb{B}}(\mathcal{A}_{\mathsf{Forg}}^{\mathcal{O}_{\mathsf{Forg}}}) \to 1]
\end{aligned}$$

□

References

[MR14] Avradip Mandal and Arnab Roy. *Relational Hash*. Cryptology ePrint Archive, Paper 2014/394. 2014. URL: https://eprint.iacr.org/2014/394.

[Lee+18] Joohee Lee et al. *Instant Privacy-Preserving Biometric Authentication for Hamming Distance*. Cryptology ePrint Archive, Paper 2018/1214. 2018. URL: https://eprint.iacr.org/2018/1214.

[PP22] Paola de Perthuis and David Pointcheval. *Two-Client Inner-Product Functional Encryption, with an Application to Money-Laundering Detection*. Cryptology ePrint Archive, Paper 2022/441. 2022. DOI: 10.1145/3548606.3559374. URL: https://eprint.iacr.org/2022/441.

[EM23] Johannes Ernst and Aikaterini Mitrokotsa. *A Framework for UC Secure Privacy Preserving Biometric Authentication using Efficient Functional Encryption*. Cryptology ePrint Archive, Paper 2023/481. 2023. URL: https://eprint.iacr.org/2023/481.