Biometrics Authentication: Formalization and Instantiation

Keng-Yu Chen

November 1, 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 $\mathsf{FE.Dec}$:

- $\mathsf{FE.Setup}(1^\lambda) \rightarrow \mathsf{msk}, \mathsf{pp}$: 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 $\mathbf{c_y}$.
- $\mathsf{FE.Dec}(\mathsf{pp}, f_{\mathbf{x}}, \mathbf{c_y}) \to z$: It outputs a value $z \in \mathbb{F}$.

Correctness: The fh-IPFE scheme FE is *correct* if $\forall(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and $\mathbf{x}, \mathbf{y} \in \mathbb{F}^k$, we have

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

Definition 2 (Two-Input Inner Product Functional Encryption (adapted from [PP22])). A *two-input inner product functional encryption* (2i-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 $\mathsf{FE.Dec}$:

- $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{sk}, \mathsf{ek}_1, \mathsf{ek}_2$: It outputs a secret key sk and two encryption keys $\mathsf{ek}_1, \mathsf{ek}_2$.
- $\mathsf{FE.KeyGen}(\mathsf{sk}, \mathbf{A}) \to \mathsf{dk}_{\mathbf{A}}$: It generates the functional decryption key $\mathsf{dk}_{\mathbf{A}}$ for a diagonal matrix $\mathbf{A} \in \mathbb{F}^{k \times k}$,
- $\mathsf{FE.Enc}(\mathsf{ek}_i, \mathbf{x}) \to \mathbf{c_x}$: Given an encryption key, either ek_1 or ek_2, it encrypts the input vector $\mathbf{x} \in \mathbb{F}^k$ to the ciphertext $\mathbf{c_x}$.
- $\mathsf{FE.Dec}(\mathsf{dk}_{\mathbf{A}}, \mathbf{c_x}, \mathbf{c_y}) \to z$: It outputs a value $z \in \mathbb{F}$.

Correctness: The 2i-IPFE scheme FE is *correct* if $\forall(\mathsf{sk}, \mathsf{ek}_1, \mathsf{ek}_2) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, $\mathbf{A} \in \mathbb{F}^{k \times k}$, and $\mathbf{x}, \mathbf{y} \in \mathbb{F}^k$, we have

$$\mathsf{FE.Dec}(\mathsf{FE.KeyGen}(\mathsf{sk}, \mathbf{A}), \mathsf{FE.Enc}(\mathsf{ek}_1, \mathbf{x}), \mathsf{FE.Enc}(\mathsf{ek}_2, \mathbf{y})) = \mathbf{x}\mathbf{A}\mathbf{y}^T \in \mathbb{F}.$$

Definition 3 (Two-Client Inner Product Functional Encryption (adapted from [PP22])). A *two-client inner product functional encryption* (2c-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 $\mathsf{FE.Dec}$:

- $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{sk}, \mathsf{ek}_1, \mathsf{ek}_2$: It outputs a secret key sk and two encryption keys $\mathsf{ek}_1, \mathsf{ek}_2$.
- $\mathsf{FE.KeyGen}(\mathsf{sk}, \mathbf{A}) \to \mathsf{dk}_{\mathbf{A}}$: It generates the functional decryption key $\mathsf{dk}_{\mathbf{A}}$ for a diagonal matrix $\mathbf{A} \in \mathbb{F}^{k \times k}$,
- $\mathsf{FE.Enc}(\ell, \mathsf{ek}_i, \mathbf{x}) \to \mathbf{c_x}$: Given a label ℓ and an encryption key, either ek_1 or ek_2, it encrypts the input vector $\mathbf{x} \in \mathbb{F}^k$ to the ciphertext $\mathbf{c_x}$.
- $\mathsf{FE.Dec}(\mathsf{dk}_{\mathbf{A}}, \mathbf{c_x}, \mathbf{c_y}) \to z$: It outputs a value $z \in \mathbb{F}$.

Correctness: The 2c-IPFE scheme FE is *correct* if $\forall(\mathsf{sk}, \mathsf{ek}_1, \mathsf{ek}_2) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, $\mathbf{A} \in \mathbb{F}^{k \times k}$, label ℓ, and $\mathbf{x}, \mathbf{y} \in \mathbb{F}^k$, we have

$$\mathsf{FE.Dec}(\mathsf{FE.KeyGen}(\mathsf{sk}, \mathbf{A}), \mathsf{FE.Enc}(\ell, \mathsf{ek}_1, \mathbf{x}), \mathsf{FE.Enc}(\ell, \mathsf{ek}_2, \mathbf{y})) = \mathbf{x}\mathbf{A}\mathbf{y}^T \in \mathbb{F}.$$

2 Formalization

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

- $\mathsf{Setup}(1^\lambda) \to \mathsf{esk}, \mathsf{psk}, \mathsf{csk}$: 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 ($r = 0$).

We discuss two usage models that employs the authentication scheme Π.

2.1 Usage Model – Device-of-User

In the model described in Figure 1 (an overview), Figure 2 (on enrollment), and Figure 3 (on authentication), users authenticate themselves to a server through their own devices and biometric scanners that are shared among different users. A key distribution service distributes keys for them. In practice, this model applies to the situation when the users access an online service run by the server.

- User: The user who enrolls its biometric data and authenticates itself to the server. We assume the user's biometric distribution is $\mathcal{B} \in \mathbb{B}$.

- Scanner: A machine to extract the user's biometric data by querying the oracle $\mathcal{O}_\mathcal{B}$.

- Device: A device belonging to the user. In practice, it can be a desktop or a mobile phone. It processes the Enroll and Probe functions for User with keys esk and psk. It queries $\mathcal{O}_\mathcal{B}$ for biometric data through the Scanner.

- KDS: A key distribution service. It runs Setup to generate keys and distribute them to Device and Server.

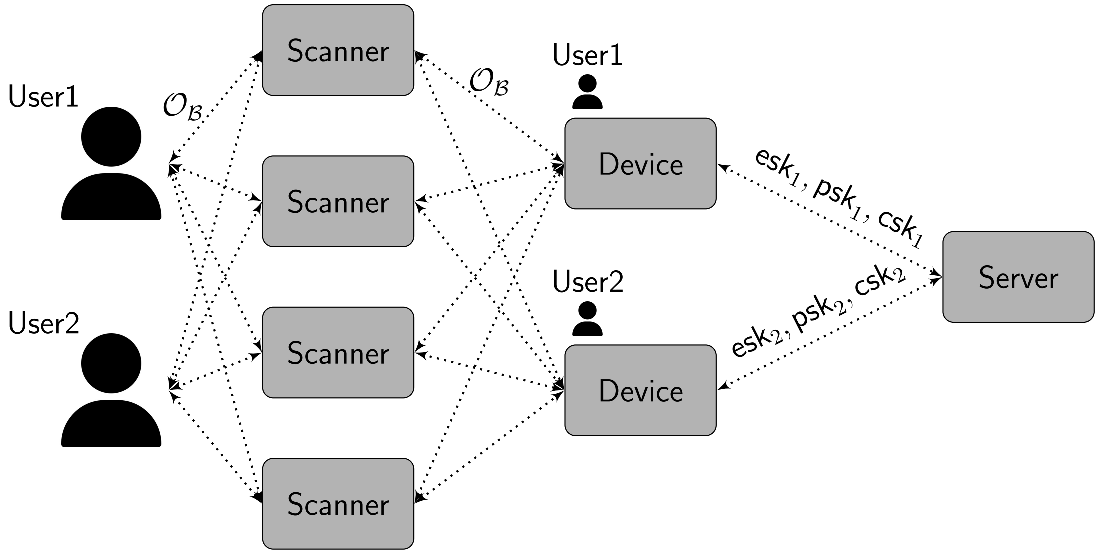

Figure 1: An Overview of the Device-of-User Usage Model

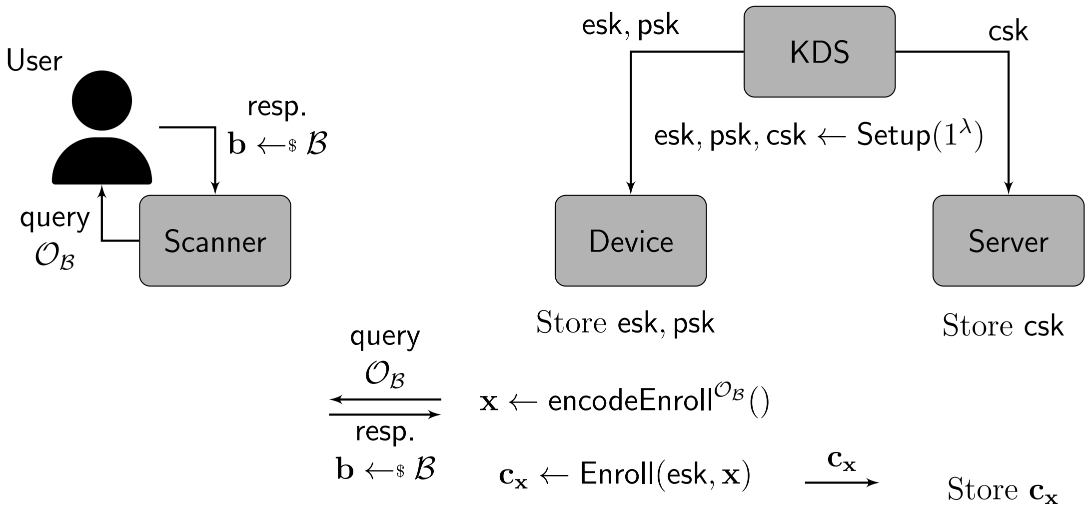

Figure 2: Device-of-User Usage Model on Enrollment

- Server: The server responsible for authenticating the user. It stores the comparison key csk and the user's enrollment message $\mathbf{c_x}$. On authentication, it compares the probe message with the registered enrollment message and returns the result.

The Device-of-User model is similar to the use case in [EM23], except that Scanner and KDS are part of Device, which is a secure hardware token in their model.

2.2 Usage Model – Device-of-Domain

In the model described in Figure 4 (an overview), Figure 5 (on enrollment), and Figure 6 (on authentication), users first enroll themselves at an enrollment station and then authenticate themselves to a server through devices that belong to a domain. A key distribution service distributes enrollment keys to the enrollment station, probe

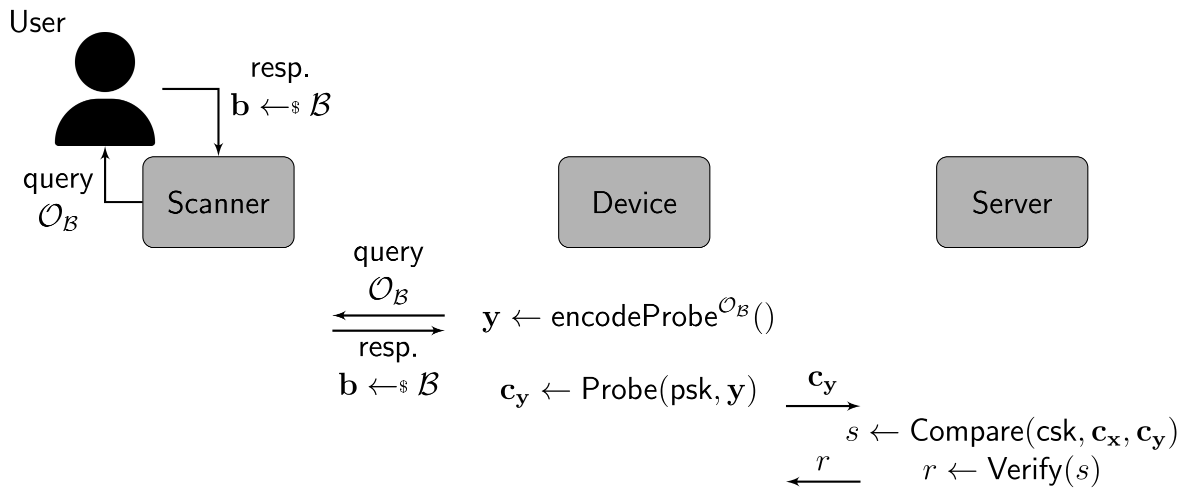

Figure 3: Device-of-User Usage Model on Authentication

keys to the domain, and comparison keys to the server. In practice, a domain can be a department in an organization, and this models applies to the situation when a user wants to access a public service of a department, such as a restricted area or instruments.

- User: The user who enrolls its biometric data at an enrollment station and authenticates itself to the server. We assume the user's biometric distribution is $\mathcal{B} \in \mathbb{B}$.
- Domain: A domain that owns several devices, all of which share one enrollment key esk, one probe key psk and one comparison key csk. Only the probe key is stored at each device of a domain. The enrollment key is stored at the enrollment station, and the comparison key is stored at the server. In practice, a domain can be a department, and users enroll and authenticate themselves before accessing a restricted service of this department.
- Scanner: A machine to extract the user's biometric data by querying the oracle $\mathcal{O}_{\mathcal{B}}$.
- Station: An enrollment station responsible for collecting the user's biometric data to enroll them for a domain on the server.
- Device: A device belonging to a domain. In practice, it can be a device checking identities for a restricted area or an instrument. It owns a probe key psk and processes the Probe function for enrolled users of this domain.
- KDS: A key distribution service. It runs Setup to generate keys and distribute them to Station, Domain, and Server.
- Server: The server responsible for authenticating the user. It stores the comparison key csk for each domain and the user's enrollment message $\mathbf{c_x}$. On authentication, it compares the probe message with the registered enrollment message and returns the result.

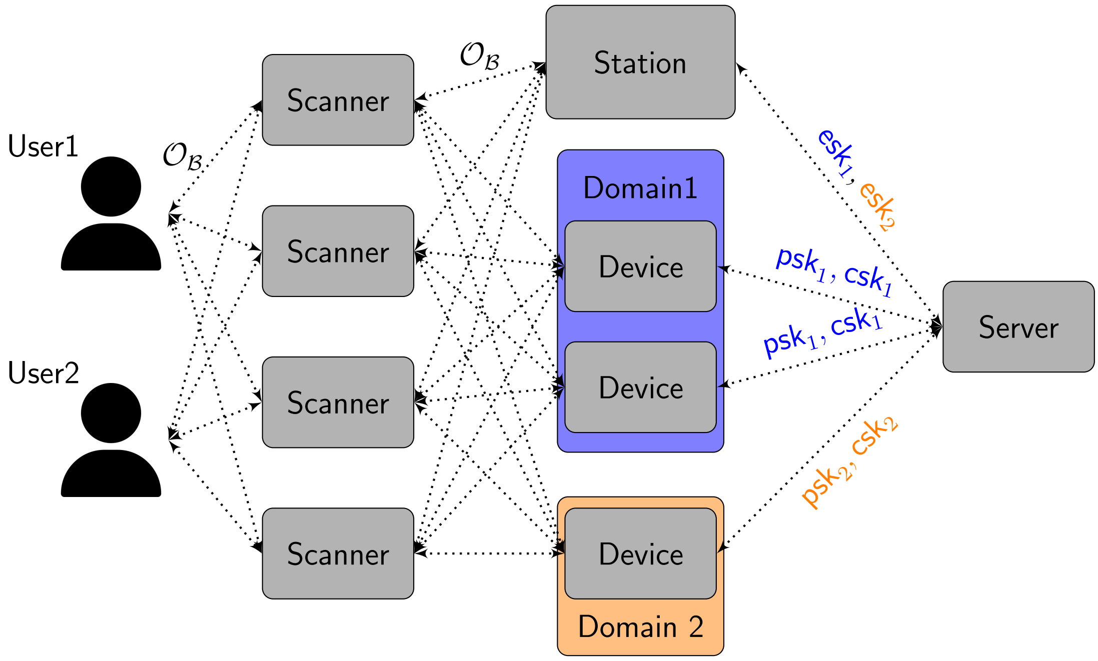

Figure 4: An overview of the Device-of-Domain Usage Model

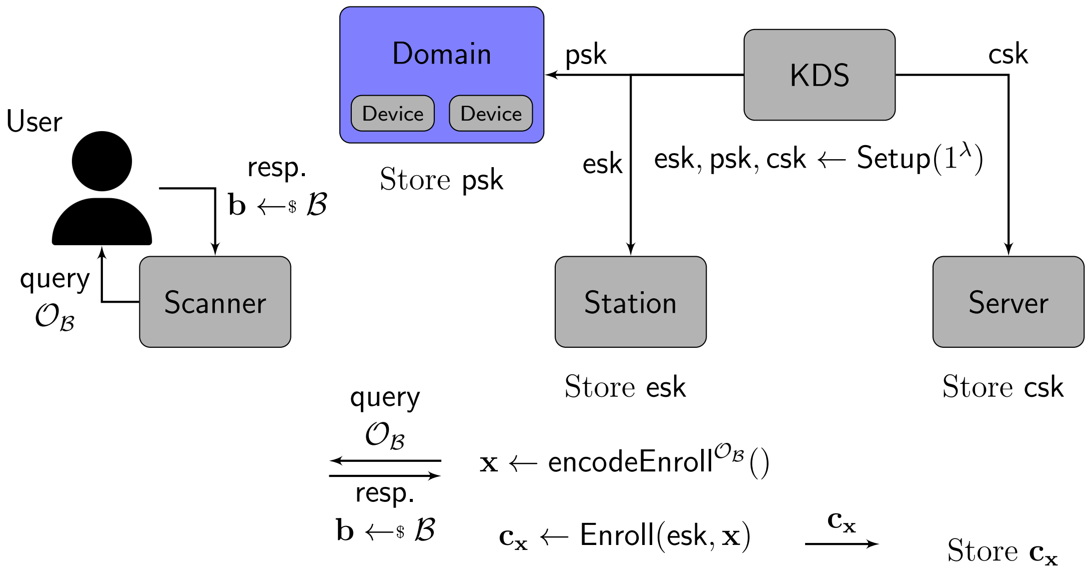

Figure 5: Device-of-Domain Usage Model on Enrollment

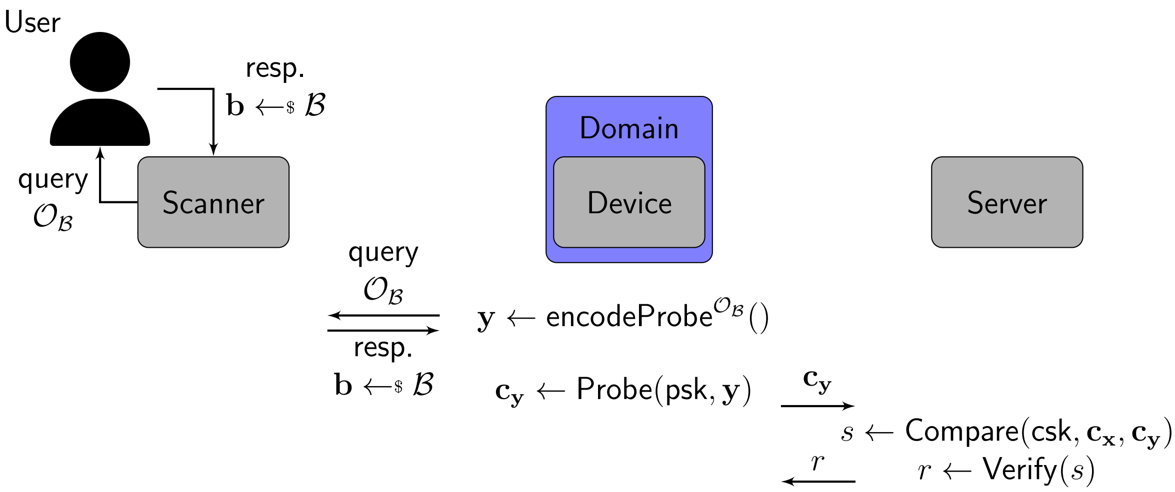

Figure 6: Device-of-Domain Usage Model on Authentication

2.3 Instantiation with an fh-IPFE Scheme

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.

- $\mathsf{Setup}(1^\lambda)$: It calls $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{msk}, \mathsf{pp}$ and outputs $\mathsf{esk} \leftarrow (\mathsf{msk}, \mathsf{pp})$, $\mathsf{psk} \leftarrow (\mathsf{msk}, \mathsf{pp})$ and $\mathsf{csk} \leftarrow \mathsf{pp}$.
- $\mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$: For a template vector $\mathbf{b} = (b_1, b_2, \cdots, b_k)$ sampled from $\mathcal{O}_{\mathcal{B}}$, the function encodes it as $\mathbf{x} = (x_1, x_2, \cdots, x_{k+2}) = (b_1, b_2, \cdots, b_k, 1, \|\mathbf{b}\|^2)$.
- $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$: 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 $\mathbf{y} = (y_1, y_2, \cdots, y_{k+2}) = (-2b'_1, -2b'_2, \cdots, -2b'_k, \|\mathbf{b}'\|^2, 1)$.
- $\mathsf{Probe}(\mathsf{psk}, \mathbf{y})$: 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

$$s = \mathsf{FE.Dec}(\mathsf{pp}, \mathbf{c_x}, \mathbf{c_y}) = \mathbf{x}\mathbf{y}^T = \sum_{i=1}^{k} -2b_i b'_i + \|\mathbf{b}\|^2 + \|\mathbf{b}'\|^2 = \|\mathbf{b} - \mathbf{b}'\|^2.$$

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.

Instantiated with an fh-IPFE scheme in this way, the comparison secret key csk is public, and the enrollment secret key esk and probe secret key psk are the same. Anyone with access to the enrollment message $\mathbf{c_x}$ and either one of esk, psk, or a probe oracle $\mathsf{Probe}(\mathsf{psk}, \cdot)$ can probe some $\mathbf{y}' \in \mathbf{F}^{k+2}$ and find $\mathbf{x}\mathbf{y}'^T$ to get partial or full information about $\mathbf{x}$. Even if the adversary can only sample random ciphertexts $\mathbf{c_y}$ without knowing $\mathbf{y}$, if the field size q is not large enough, one can find a forged $\mathbf{c}_{\mathbf{y}^*}$ such that $\mathbf{x}\mathbf{y}^{*T} < \tau$ to impersonate the user by sampling many times offline.

Therefore, Server must store $\mathbf{c_x}$ securely, to avoid such an attack from an adversary who can access the probe oracle; Device must protect its probe function, to avoid such an attack from a malicious Server.

In the Device-of-Domain model, we assume the probe oracle is public, just as everyone can try accessing a public service. A malicious $\mathsf{Station}$ or Server, who has the enrollment message $\mathbf{c_x}$, can utilize this attack to retrieve information about User.

2.4 Instantiation with a 2i-IPFE Scheme

Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be a 2i-IPFE scheme we defined in Definition 2. Following the scheme in Section 2.3, we can instantiate a biometric authentication scheme using FE.

- $\mathsf{Setup}(1^\lambda)$: It calls $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{sk}, \mathsf{ek}_1, \mathsf{ek}_2$, $\mathsf{FE.KeyGen}(sk, \mathbf{I}_{k+2}) \to \mathsf{dk}_\mathbf{I}$, where $\mathbf{I}_{k+2}$ is an identity matrix of size $(k+2)\times(k+2)$. It outputs $\mathsf{esk} \leftarrow \mathsf{ek}_1$, $\mathsf{psk} \leftarrow \mathsf{ek}_2$, and $\mathsf{csk} \leftarrow \mathsf{dk}_\mathbf{I}$
- $\mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}(), \mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$: The same as the scheme in 2.3.
- $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$: It calls $\mathsf{FE.Enc}(\mathsf{ek}_1, \mathbf{x}) \to \mathbf{c_x}$ and outputs $\mathbf{c_x}$.
- $\mathsf{Probe}(\mathsf{psk}, \mathbf{y})$: It calls $\mathsf{FE.Enc}(\mathsf{ek}_2, \mathbf{y}) \to \mathbf{c_y}$ and outputs $\mathbf{c_y}$.
- $\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$: It calls $\mathsf{FE.Dec}(\mathsf{dk}_\mathbf{I}, \mathbf{c_x}, \mathbf{c_y}) \to s$ 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

$$s = \mathsf{FE.Dec}(\mathsf{dk}_\mathbf{I}, \mathbf{c_x}, \mathbf{c_y}) = \mathbf{x}\mathbf{I}_{k+2}\mathbf{y}^T = \mathbf{x}\mathbf{y}^T = \|\mathbf{b}-\mathbf{b}'\|^2.$$

just as the scheme in Section 2.3

Unlike the previous scheme, instantiated with a 2i-IPFE scheme in this way, the comparison secret key csk is now secret, and the enrollment secret key esk and probe secret key psk are distinct. Without csk, one cannot compare an enrollment message $\mathbf{c_x}$ and a probe message $\mathbf{c_y}$. We can also transmit $\mathbf{c_x}$ in a public channel and store it in a public storage, under necessary security requirements of the 2i-IPFE scheme, such as indistinguishability of $\mathbf{c_x}$.

In the Device-of-Domain model, the indistinguishability of $\mathbf{c_x}$ is against an adversary who has a probe oracle $\mathsf{Probe}(\mathsf{psk}, \cdot)$. If Server is malicious, then it can use csk to distinguish $\mathbf{c_x}$ enrolled by different samples. Therefore, we must limit the adversary's ability. For example, we can require the adversary to distinguish biometric vectors sampled from distributions in a pre-defined pool, and the adversary can only probe vectors randomly sampled from a distribution in the pool. We can also limit the rate of the probe oracle.

2.5 Instantiation with a 2c-IPFE Scheme

Note that if labels remain constant, a 2c-IPFE scheme is reduced to a 2i-IPFE scheme. Therefore, we can consider utilizing the label to represent each domain in the Device-of-Domain model. Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be a 2c-IPFE scheme we defined in Definition 3. Following the scheme in Section 2.4, we can instantiate a biometric authentication scheme using FE.

- $\mathsf{Setup}(1^\lambda)$: It calls $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{sk}, \mathsf{ek}_1, \mathsf{ek}_2$, $\mathsf{FE.KeyGen}(sk, \mathbf{I}_{k+2}) \to \mathsf{dk}_\mathbf{I}$, where $\mathbf{I}_{k+2}$ is an identity matrix of size $(k+2) \times (k+2)$. For keys used for Domain ℓ, it outputs $\mathsf{esk} \leftarrow (\ell, \mathsf{ek}_1)$, $\mathsf{psk} \leftarrow (\ell, \mathsf{ek}_2)$, and $\mathsf{csk} \leftarrow \mathsf{dk}_\mathbf{I}$.

  Note that when the previous 2i-IPFE-based scheme in Section 2.4 is applied to a Device-of-Domain model, we assume that Setup is run once for each domain to generate different $\mathsf{esk}, \mathsf{psk}, \mathsf{csk}$. In the scheme in this section, however, Setup is run only once for all the domains, and each domain shares the same csk and the same $\mathsf{esk}, \mathsf{psk}$ except different labels.

- $\mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}(), \mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$: The same as the scheme in 2.4.

- $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$: It calls $\mathsf{FE.Enc}(\ell, \mathsf{ek}_1, \mathbf{x}) \to \mathbf{c_x}$ and outputs $\mathbf{c_x}$.

- $\mathsf{Probe}(\mathsf{psk}, \mathbf{y})$: It calls $\mathsf{FE.Enc}(\ell, \mathsf{ek}_2, \mathbf{y}) \to \mathbf{c_y}$ and outputs $\mathbf{c_y}$.

- $\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$: It calls $\mathsf{FE.Dec}(\mathsf{dk}_\mathbf{I}, \mathbf{c_x}, \mathbf{c_y}) \to s$ 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, if the labels of $\mathbf{c_x}$ and $\mathbf{c_y}$ are the same (they are of the same domain), we have

$$s = \mathsf{FE.Dec}(\mathsf{dk}_\mathbf{I}, \mathbf{c_x}, \mathbf{c_y}) = \mathbf{x}\mathbf{I}_{k+2}\mathbf{y}^T = \|\mathbf{b} - \mathbf{b}'\|^2.$$

just as the scheme in Section 2.4

When the Device-of-Domain model is instantiated with a 2c-IPFE scheme in this way, the enrollment secret key esk and probe secret key psk are now shared among all the devices, regardless of their domains. Therefore, to let a malicious or broken Domain not threaten other honest ones, one needs to make sure given esk or psk, $\mathbf{c_x}$ still does not leak information about $\mathbf{x}$. This is different from the scheme in Section 2.4, where we only need seurity against an adversary who has a probe oracle $\mathsf{Probe}(\mathsf{psk}, \cdot)$.

If Server and Domain are both malicious, then the adversary can use csk to distinguish $\mathbf{c_x}$ and even recover $\mathbf{x}$. Therefore, we assume at most one party of them can be malicious at the same time. Note that this is the same as the 2i-IPFE-based scheme, where only one of Server and Domain can be malicious.

3 Security Games

From now on, we consider a family of biometric distributions $\mathbb{B}$. Removing a person $\mathcal{B}$ from $\mathbb{B}$ is written as $\mathbb{B} \setminus \mathcal{B}$. To model the knowledge about the biometric distributions, we offer an oracle $\mathcal{O}_{\mathsf{samp}}(\cdot)$ to all adversaries in this section.

- $\mathcal{O}_{\mathsf{samp}}(\cdot)$: On input an index i,
  - If i was not queried before, it first samples a biometric distribution $\mathcal{B}_i \in \mathbb{B}$ and then outputs a biometric sample $\mathbf{b} \leftarrow_\$ \mathcal{B}_i$.
  - If i has been queried before, it outputs a biometric sample $\mathbf{b} \leftarrow_\$ \mathcal{B}_i$.

3.1 Unforgeability against Malicious Scanner (UF-MSC)

In the game of Unforgeability against Malicious Scanner, we model the ability of a malicious $\mathsf{Scanner}$ in the Device-of-User model who has access to Server's database of registered enrollments and tries to impersonate User. The adversary $\mathcal{A}$ is given the enrollment message $\mathbf{c_x}$ and oracles $\mathcal{O}_{\mathcal{B}}$ and $\mathcal{O}^q_{\mathsf{auth}}$, and it tries to find a valid probe message $\tilde{\mathbf{z}}$. The whole game $\mathsf{UF\text{-}MSC}$ is defined in Algorithm 1.

Algorithm 1 $\mathsf{UF\text{-}MSC}_{\Pi,\mathbb{B}}(\mathcal{A})$

1: $\mathcal{B} \leftarrow_\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
2: $\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$
4: $\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
5: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{B}}, \mathcal{O}^q_{\mathsf{auth}}}(\mathbf{c_x})$
6: $s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
7: **return** $\mathsf{Verify}(s)$

Algorithm 2 $\mathsf{UF\text{-}MSC}'_{\Pi,\mathbb{B}}(\mathcal{A}')$

1: $\mathcal{B} \leftarrow_\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
2: $\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$
4: $\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
5: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}^q_{\mathsf{auth}}}()$
6: $s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
7: **return** $\mathsf{Verify}(s)$

The given oracle is defined as follows:

- $\mathcal{O}_{\mathcal{B}}$: It outputs a biometric sample $\mathbf{b} \leftarrow_\$ \mathcal{B}$.
- $\mathcal{O}^q_{\mathsf{auth}}(\mathsf{csk}, \mathbf{c_x}, \cdot)$: This is a resource-limited oracle. If it has been queried over q times in total, it aborts. Otherwise, on input $\mathbf{z}$, it outputs $\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{z}))$.

To consider potential false positives of biometrics match, we consider the plain $\mathsf{UF\text{-}MSC}'$ game in Algorithm 2, in which the adversary does not have any knowledge about the template.

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

$$\mathbf{Adv}^{\mathsf{UF\text{-}MSC}}_{\Pi,\mathbb{B},\mathcal{A}} := \Pr[\mathsf{UF\text{-}MSC}_{\Pi,\mathbb{B}}(\mathcal{A}) \to 1] - q \cdot \sup_{\text{PPT } \mathcal{A}'} \Pr[\mathsf{UF\text{-}MSC}'_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1].$$

The q factor is to avoid trivial attacks. Let $\mathcal{A}'$ be a $\mathsf{UF\text{-}MSC}'$ adversary. An adversary $\mathcal{A}$ can run $\mathcal{A}'() \to \tilde{\mathbf{z}}$ and $\mathcal{O}^q_{\mathsf{auth}}(\mathsf{csk}, \mathbf{c}_\mathbf{x}, \tilde{\mathbf{z}})$ up to q times and then outputs a successful forged probe message if there is any. The probability that $\mathcal{A}$ wins is

$$\begin{aligned}\Pr[\mathsf{UF\text{-}MSC}_{\Pi,\mathbb{B}}(\mathcal{A}) \to 1] &= 1 - (1 - \Pr[\mathsf{UF\text{-}MSC}'_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1])^q \\ &\le q \cdot \Pr[\mathsf{UF\text{-}MSC}'_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1]\end{aligned}$$

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *unforgeable against malicious scanner (UF-MSC)* if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{UF\text{-}MSC}}_{\Pi,\mathbb{B},\mathcal{A}} = \mathsf{negl}.$$

Note that if csk is an empty or public string, then the scheme cannot achieve UF-MSC security when the false positive rate is not negligible, as the adversary can run the $\mathsf{Compare}(\mathsf{csk}, \mathbf{c}_\mathbf{x}, \cdot)$ over q times. Also note that in the Device-of-Domain model, the probe oracle $\mathsf{Probe}(\mathsf{psk}, \cdot)$ is assumed to be public, so the scheme is never UF-MSC since the adversary can generate valid probe messages itself. Therefore, we only consider UF-MSC security in the Device-of-User model.

3.2 Unforgeability against Malicious Device (UF-MDV)

In the game of Unforgeability against Malicious Device, we model the ability of a malicious Device who has access to Server's database of registered enrollments and tries to impersonate User. The adversary $\mathcal{A}$ is given the keys $\mathsf{esk}, \mathsf{psk}$ (only psk in Device-of-Domain model), enrollment message $\mathbf{c}_\mathbf{x}$, and oracle $\mathcal{O}^q_{\mathsf{auth}}$ and tries to find a valid probe message $\tilde{\mathbf{z}}$. The whole game $\mathsf{UF\text{-}MDV}$ is defined in Algorithm 3. Similarly to Section 3.1, we also consider the plain $\mathsf{UF\text{-}MDV}'$ game in Algorithm 4.

Algorithm 3 $\mathsf{UF\text{-}MDV}_{\Pi,\mathbb{B}}(\mathcal{A})$

$\mathcal{B} \leftarrow_\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
$\mathbf{c}_\mathbf{x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
In Device-of-User model:
  $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}^q_{\mathsf{auth}}}(\mathsf{esk}, \mathsf{psk}, \mathbf{c}_\mathbf{x})$
In Device-of-Domain model:
  $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}^q_{\mathsf{auth}}}(\mathsf{psk}, \mathbf{c}_\mathbf{x})$
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c}_\mathbf{x}, \tilde{\mathbf{z}})$
return $\mathsf{Verify}(s)$

Algorithm 4 $\mathsf{UF\text{-}MDV}'_{\Pi,\mathbb{B}}(\mathcal{A}')$

$\mathcal{B} \leftarrow_\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
$\mathbf{c}_\mathbf{x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
In Device-of-User model:
  $\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}^q_{\mathsf{auth}}}()$
In Device-of-Domain model:
  $\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}_{\mathsf{Probe}}, \mathcal{O}^q_{\mathsf{auth}}}()$
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c}_\mathbf{x}, \tilde{\mathbf{z}})$
return $\mathsf{Verify}(s)$

Note that in Device-of-Domain model, a probe oracle is given to $\mathsf{UF\text{-}MDV}'$ adversary.

- $\mathcal{O}_{\mathsf{Probe}}(\mathsf{psk}, \cdot)$: On input $\mathbf{y}'$, it outputs the probe message $\mathsf{Probe}(\mathsf{psk}, \mathbf{y}')$.

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

$$\mathbf{Adv}^{\mathsf{UF\text{-}MDV}}_{\Pi,\mathbb{B},\mathcal{A}} := \Pr[\mathsf{UF\text{-}MDV}_{\Pi,\mathbb{B}}(\mathcal{A}) \to 1] - q \cdot \sup_{\text{PPT } \mathcal{A}'} \Pr[\mathsf{UF\text{-}MDV'}_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1].$$

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *unforgeable against malicious device (UF-MDV)* if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{UF\text{-}MDV}}_{\Pi,\mathbb{B},\mathcal{A}} = \mathsf{negl}.$$

3.3 Unforgeability against Malicious Domain (UF-MDM)

In the game of Unforgeability against Malicious Domain, we model the ability of a malicious Domain in the Device-of-Domain model who tries to access an honest Domain through a User who has enrolled in both of them. The adversary $\mathcal{A}$ is given the enrollment message $\mathbf{c_x}$ of the honest Domain and oracles $\mathcal{O}_{\mathsf{Probe}}$, $\mathcal{O}'_{\mathsf{Probe}}$, and $\mathcal{O}^q_{\mathsf{auth}}$, and it tries to find a valid probe message $\tilde{\mathbf{z}}$. The whole game $\mathsf{UF\text{-}MDM}$ is defined in Algorithm 5. Similarly to Section 3.1, we also consider the plain $\mathsf{UF\text{-}MDM'}$ game in Algorithm 6.

Algorithm 5 $\mathsf{UF\text{-}MDM}_{\Pi,\mathbb{B}}(\mathcal{A})$

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

Algorithm 6 $\mathsf{UF\text{-}MDM}'_{\Pi,\mathbb{B}}(\mathcal{A}')$

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

The $\mathcal{O}'_{\mathsf{Probe}}$ oracle is to model the ability that the malicious Domain can let User probe with a contrived key.

- $\mathcal{O}'_{\mathsf{Probe}}(\cdot)$: On input psk', it first samples $\mathbf{y}' \leftarrow_{\$} \mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$ and outputs $\mathsf{Probe}(\mathsf{psk}', \mathbf{y}')$.

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

$$\mathbf{Adv}^{\mathsf{UF\text{-}MDM}}_{\Pi,\mathbb{B},\mathcal{A}} := \Pr[\mathsf{UF\text{-}MDM}_{\Pi,\mathbb{B}}(\mathcal{A}) \to 1] - q \cdot \sup_{\text{PPT } \mathcal{A}'} \Pr[\mathsf{UF\text{-}MDM}'_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1].$$

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *unforgeable against malicious domain (UF-MDM)* if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{UF\text{-}MDM}}_{\Pi,\mathbb{B},\mathcal{A}} = \mathsf{negl}.$$

3.4 Unforgeability against Malicious Station (UF-MST)

In the game of Unforgeability against Malicious Station, we model the ability of a malicious Station in the Device-of-Domain model who tries to impersonate User. The adversary $\mathcal{A}$ is given the enrollment key esk, enrollment message $\mathbf{c_x}$, and oracles $\mathcal{O}_{\mathsf{Probe}}, \mathcal{O}'_{\mathsf{Enroll}}$, and $\mathcal{O}^q_{\mathsf{auth}}$, and it tries to find a valid probe message $\tilde{\mathbf{z}}$. The whole game UF-MST is defined in Algorithm 7. Similarly to Section 3.1, we also consider the plain UF-MST′ game in Algorithm 8.

Algorithm 7 $\mathsf{UF\text{-}MST}_{\Pi,\mathbb{B}}(\mathcal{A})$

1: $\mathcal{B} \leftarrow_\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
2: $\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
4: $\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
5: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Probe}}, \mathcal{O}'_{\mathsf{Enroll}}, \mathcal{O}^q_{\mathsf{auth}}}(\mathbf{c_x})$
6: $s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
7: **return** $\mathsf{Verify}(s)$

Algorithm 8 $\mathsf{UF\text{-}MST}'_{\Pi,\mathbb{B}}(\mathcal{A}')$

1: $\mathcal{B} \leftarrow_\$ \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
2: $\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
4: $\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
5: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}_{\mathsf{Probe}}, \mathcal{O}^q_{\mathsf{auth}}}()$
6: $s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
7: **return** $\mathsf{Verify}(s)$

The $\mathcal{O}'_{\mathsf{Enroll}}(\cdot)$ oracle is to model the ability that the malicious Station can let User enroll with a contrived key.

- $\mathcal{O}'_{\mathsf{Enroll}}(\cdot)$: On input esk', it first samples $\mathbf{x}' \leftarrow_\$ \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$ and outputs $\mathsf{Enroll}(\mathsf{esk}', \mathbf{x}')$.

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

$$\mathbf{Adv}^{\mathsf{UF\text{-}MST}}_{\Pi,\mathbb{B},\mathcal{A}} := \Pr[\mathsf{UF\text{-}MST}_{\Pi,\mathbb{B}}(\mathcal{A}) \to 1] - q \cdot \sup_{\text{PPT } \mathcal{A}'} \Pr[\mathsf{UF\text{-}MST}'_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1].$$

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *unforgeable against malicious station (UF-MST)* if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{UF\text{-}MST}}_{\Pi,\mathbb{B},\mathcal{A}} = \mathsf{negl}.$$

3.5 Indistinguishable against Malicious Server (IND-MSV)

In the game of Indistinguishable against Malicious Server, we model the ability of a malicious Server who tries to identify the user. The adversary $\mathcal{A}$ is given oracles to two biometric distributions $\mathcal{B}^{(0)}, \mathcal{B}^{(1)}$, the comparison key csk, an enrollment message $\mathbf{c_x}$, and a list of t probe messages $\{\mathbf{c_y}^{(i)}\}_{i=1}^t$. It tries to guess from either $\mathcal{B}^{(0)}$ or $\mathcal{B}^{(1)}$ these messages are generated. The whole game is defined in Algorithm 9.

Note that in Device-of-Domain model, a probe oracle is given to the adversary.

- $\mathcal{O}^{\mathsf{samp}}_{\mathsf{Probe}}(\cdot)$: On input an index i, it first samples $\mathbf{y}' \leftarrow_\$ \mathsf{encodeProbe}^{\mathcal{O}_{\mathsf{samp}}(i)}$, which uses $\mathcal{O}_{\mathsf{samp}}(i)$ to answer biometric queries, and outputs $\mathsf{Probe}(\mathsf{psk}, \mathbf{y}')$.

Algorithm 9 $\mathsf{IND\text{-}MSV}_{\Pi,\mathbb{B}}(\mathcal{A})$

$b \leftarrow_\$ \{0,1\}$
$\mathcal{B}^{(0)} \leftarrow_\$ \mathbb{B}, \quad \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}^{(0)}$
$\mathcal{B}^{(1)} \leftarrow_\$ \mathbb{B}, \quad \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}^{(1)}$
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}^{(b)}}}()$
$\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
for $i = 1$ to t **do**
    $\mathbf{y}^{(i)} \leftarrow \mathsf{encodeProbe}^{\mathcal{O}_{\mathcal{B}^{(b)}}}()$
    $\mathbf{c_y}^{(i)} \leftarrow \mathsf{Probe}(\mathsf{psk}, \mathbf{y}^{(i)})$
end for
In Device-of-User Model:
    $\tilde{b} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{B}}^{(0)},\mathcal{O}_{\mathcal{B}}^{(1)}}(\mathsf{csk}, \mathbf{c_x}, \{\mathbf{c_y}^{(i)}\}_{i=1}^{t})$
In Device-of-Domain Model:
    $\tilde{b} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{B}}^{(0)},\mathcal{O}_{\mathcal{B}}^{(1)},\mathcal{O}_{\mathsf{Probe}}^{\mathsf{samp}}}(\mathsf{csk}, \mathbf{c_x}, \{\mathbf{c_y}^{(i)}\}_{i=1}^{t})$
return $1_{\tilde{b}=b}$

We provide $\mathcal{O}_{\mathsf{Probe}}^{\mathsf{samp}}(\cdot)$ instead of $\mathcal{O}_{\mathsf{Probe}}(\mathsf{psk}, \cdot)$. This is to avoid the trivial attack where the adversary probes samples from the oracles $\mathcal{O}_{\mathcal{B}^{(0)}}$ and $\mathcal{O}_{\mathcal{B}^{(1)}}$ and compare the results with $\mathbf{c_x}$.

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

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

An authentication scheme Π associated with a family of distributions $\mathbb{B}$ is called *indistinguishable against malicious server (IND-MSV)* if for any PPT adversary $\mathcal{A}$,

$$\mathbf{Adv}_{\Pi,\mathbb{B},\mathcal{A}}^{\mathsf{IND\text{-}MSV}} = \mathsf{negl}.$$

References

[Boy04] Xavier Boyen. “Reusable cryptographic fuzzy extractors”. In: *Proceedings of the 11th ACM Conference on Computer and Communications Security*. CCS ’04. Washington DC, USA: Association for Computing Machinery, 2004, pp. 82–91. ISBN: 1581139616. DOI: 10.1145/1030083.1030096. URL: https://doi.org/10.1145/1030083.1030096.

[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.