# Biometrics Authentication: Formalization and Instantiation

Keng-Yu Chen

December 4, 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

- $\lambda$ 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 $\mathsf{negl}$ to represent an arbitrary negligible function.
- $\mathsf{poly}$ is the class of polynomial funcions. We may also use $\mathsf{poly}$ 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.

We introduce three types of inner product functional encryption schemes: function hiding functional encryption, two-input functional encryption, and two-client functional encryption. We will instantiate our biometric authentication scheme using these primitives.

**Definition 1** (Function Hiding Inner Product Functional Encryption)**.** A *function hiding inner product functional encryption* (fh-IPFE) scheme $\mathsf{FE}$ 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 $\mathsf{pp}$ and the master secret key $\mathsf{msk}$.
- $\mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pp}, \mathbf{x}) \to f_\mathbf{x}$: 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 $\mathsf{FE}$ 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

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

Instantiation using an fh-IPFE scheme is given in Section 2.3.

**Definition 2** (Two-Input Inner Product Functional Encryption (adapted from [PP22]))**.** A *two-input inner product functional encryption* (2i-IPFE) scheme $\mathsf{FE}$ 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 $\mathsf{sk}$ 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 $\mathsf{ek}_1$ or $\mathsf{ek}_2$, 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 $\mathsf{FE}$ 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

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

Instantiation using a 2i-IPFE is given in Section 2.4.

**Definition 3** (Two-Client Inner Product Functional Encryption (adapted from [PP22]))**.** A *two-client inner product functional encryption* (2c-IPFE) scheme $\mathsf{FE}$ 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 $\mathsf{sk}$ 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 $\mathbf{A} \in \mathbb{F}^{k\times k}$,

- $\mathsf{FE.Enc}(\ell, \mathsf{ek}_i, \mathbf{x}) \rightarrow \mathbf{c_x}$: Given a label $\ell$ and an encryption key, either $\mathsf{ek}_1$ or $\mathsf{ek}_2$, it encrypts the input vector $\mathbf{x} \in \mathbb{F}^k$ to the ciphertext $\mathbf{c_x}$.
- $\mathsf{FE.Dec}(\mathsf{dk}_\mathbf{A}, \mathbf{c_x}, \mathbf{c_y}) \rightarrow z$: It outputs a value $z \in \mathbb{F}$.

Correctness: The 2c-IPFE scheme $\mathsf{FE}$ 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 $\ell$, and $\mathbf{x}, \mathbf{y} \in \mathbb{F}^k$, we have

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

Instantiation using a 2c-IPFE is given in Section 2.5.
We also consider an instantiation using a relational hash scheme.

**Definition 4** (Relational Hash (adapted from [MR14]))**.** Let $R_\lambda$ be a relation over sets $X_\lambda, Y_\lambda$, and $Z_\lambda$. A *relational hash* scheme $\mathsf{RH}$ for $R_\lambda$ consists of PPT algorithms $\mathsf{RH.KeyGen}$, $\mathsf{RH.HASH}_1$, $\mathsf{RH.HASH}_2$, and $\mathsf{RH.Verify}$:

- $\mathsf{RH.KeyGen}(1^\lambda) \rightarrow \mathsf{pk}$: It outputs a public hash key $\mathsf{pk}$.
- $\mathsf{RH.Hash}_1(\mathsf{pk}, \mathbf{x}) \rightarrow \mathbf{h_x}$: Given a hash key $\mathsf{pk}$ and $\mathbf{x} \in X_\lambda$, it outputs a hash $\mathbf{h_x}$.
- $\mathsf{RH.Hash}_2(\mathsf{pk}, \mathbf{y}) \rightarrow \mathbf{h_y}$: Given a hash key $\mathsf{pk}$ and $\mathbf{y} \in Y_\lambda$, it outputs a hash $\mathbf{h_y}$.
- $\mathsf{RH.Verify}(\mathsf{pk}, \mathbf{h_x}, \mathbf{h_y}) \rightarrow r \in \{0,1\}$: Given a hash key $\mathsf{pk}$, two hashes $\mathbf{h_x}$ and $\mathbf{h_y}$, and $\mathbf{z} \in Z_\lambda$, it verifies whether the relation among $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ holds.

Correctness: The relational hash scheme $\mathsf{RH}$ is *correct* if $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in X_\lambda \times Y_\lambda \times Z_\lambda$,

$$\Pr\left[\begin{cases} \mathsf{pk} \leftarrow \mathsf{RH.KeyGen}(1^\lambda) \\ \mathbf{h_x} \leftarrow \mathsf{RH.Hash}_1(\mathsf{pk}, \mathbf{x}) \\ \mathbf{h_y} \leftarrow \mathsf{RH.Hash}_2(\mathsf{pk}, \mathbf{y}) \end{cases} : \mathsf{RH.Verify}(\mathsf{pk}, \mathbf{h_x}, \mathbf{h_y}, \mathbf{z}) = R(\mathbf{x}, \mathbf{y}, \mathbf{z})\right] = 1 - \mathsf{negl}.$$

Instantiation using a relational hash is given in Section 2.6.

## 2 Formalization

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

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{esk}, \mathsf{psk}, \mathsf{csk}$: It outputs the enrollment secret key $\mathsf{esk}$, probe secret key $\mathsf{psk}$, and compare secret key $\mathsf{csk}$.
- $\mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}() \rightarrow \mathbf{x}$: 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.
- $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x}) \rightarrow \mathbf{c_x}$: 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.
- $\mathsf{Probe}(\mathsf{psk}, \mathbf{y}) \to \mathbf{c}_{\mathbf{y}}$: It outputs the probe message $\mathbf{c}_{\mathbf{y}}$ from $\mathbf{y}$.
- $\mathsf{Compare}(\mathsf{csk}, \mathbf{c}_{\mathbf{x}}, \mathbf{c}_{\mathbf{y}}) \to s$: It compares the enrollment message $\mathbf{c}_{\mathbf{x}}$ and probe message $\mathbf{c}_{\mathbf{y}}$ 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 $\Pi$.

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

The Device-of-User model, when instantiated by an fh-IPFE scheme (Section 2.3), is analogous to the use case presented in [EM23]. In their model, a user possesses a personal device, such as a smartphone or laptop, and a secure hardware device that runs an initial setup and stores all the keys, which corresponds to our KDS. On enrollemnt and authentication, the user inputs biometric templates onto the device, which corresponds to our Scanner. Subsequently, the device transmits the template to the secure hardware for the enrollment or probing processes, which are equivalent to our Device. In addition, they incorporate a two-factor authentication mechanism. The secure hardware also executes a digital signature scheme and sign the probe message on authentication.

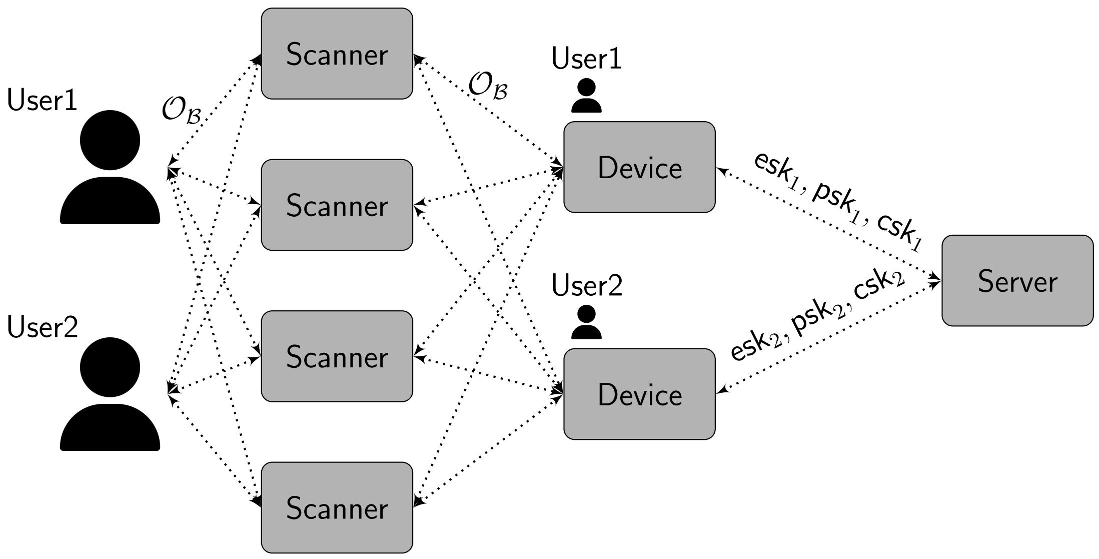

User1
User2
$\mathcal{O}_{\mathcal{B}}$
$\mathcal{O}_{\mathcal{B}}$
Scanner
Scanner
Scanner
Scanner
User1
Device
User2
Device
$\mathsf{esk}_1, \mathsf{psk}_1, \mathsf{csk}_1$
$\mathsf{esk}_2, \mathsf{psk}_2, \mathsf{csk}_2$
Server

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

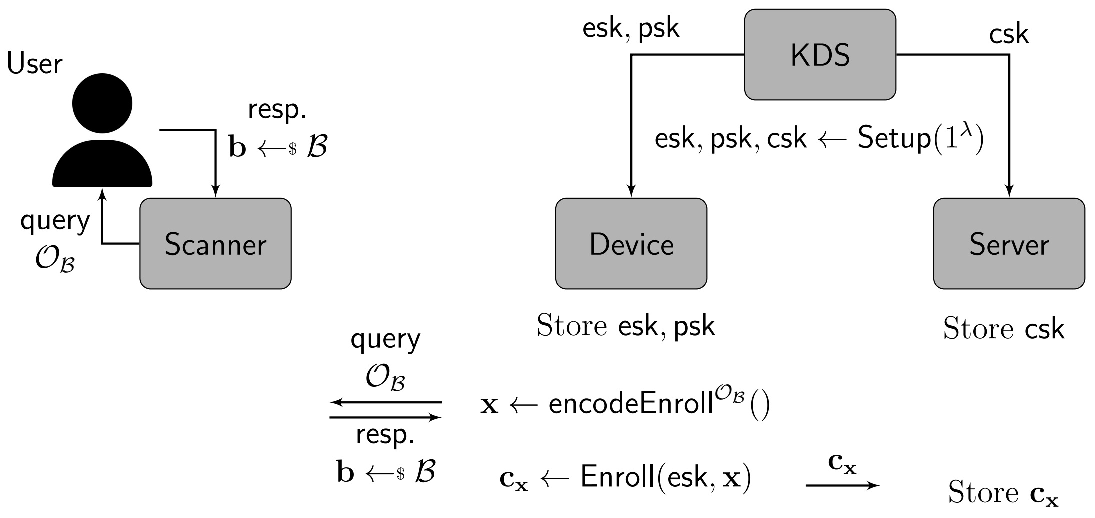
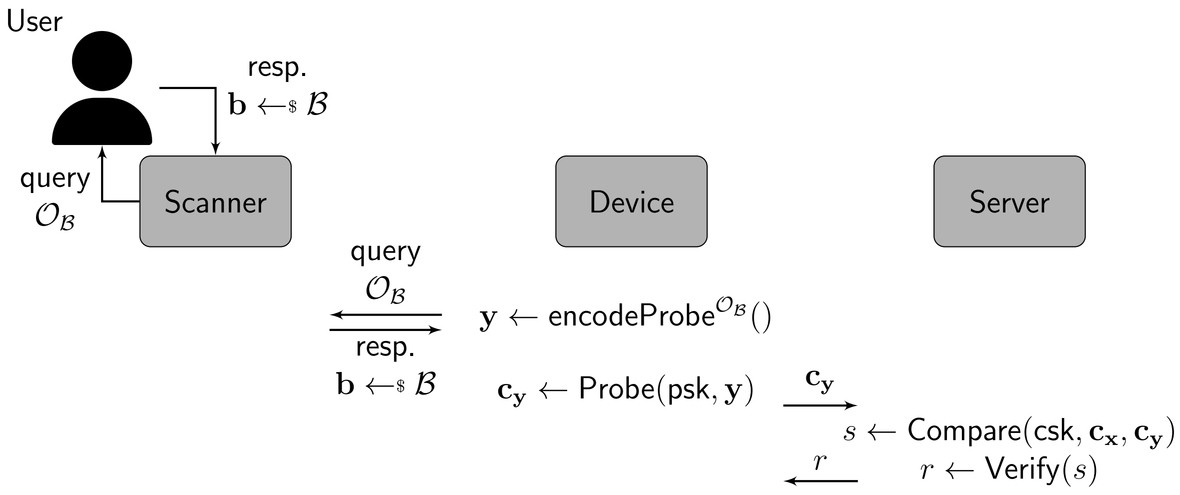

User
resp.
$\mathbf{b} \leftarrow_\$ \mathcal{B}$
query
$\mathcal{O}_{\mathcal{B}}$
Scanner
esk, psk
KDS
csk
$\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
Device
Server
Store esk, psk
Store csk
query
$\mathcal{O}_{\mathcal{B}}$
resp.
$\mathbf{b} \leftarrow_\$ \mathcal{B}$
$\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$
$\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
$\mathbf{c_x}$
Store $\mathbf{c_x}$

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



User
resp.
$\mathbf{b} \leftarrow_{\$} \mathcal{B}$
query
$\mathcal{O}_{\mathcal{B}}$
Scanner
Device
Server
query
$\mathcal{O}_{\mathcal{B}}$
resp.
$\mathbf{b} \leftarrow_{\$} \mathcal{B}$
$\mathbf{y} \leftarrow \mathsf{encodeProbe}^{\mathcal{O}_{\mathcal{B}}}()$
$\mathbf{c_y} \leftarrow \mathsf{Probe}(\mathsf{psk}, \mathbf{y})$
$\mathbf{c_y}$
$s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$
$r$
$r \leftarrow \mathsf{Verify}(s)$

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

## 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 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 $\mathsf{esk}$, one probe key $\mathsf{psk}$ and one comparison key $\mathsf{csk}$. 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 $\mathsf{psk}$ and processes the $\mathsf{Probe}$ function for enrolled users of this domain.

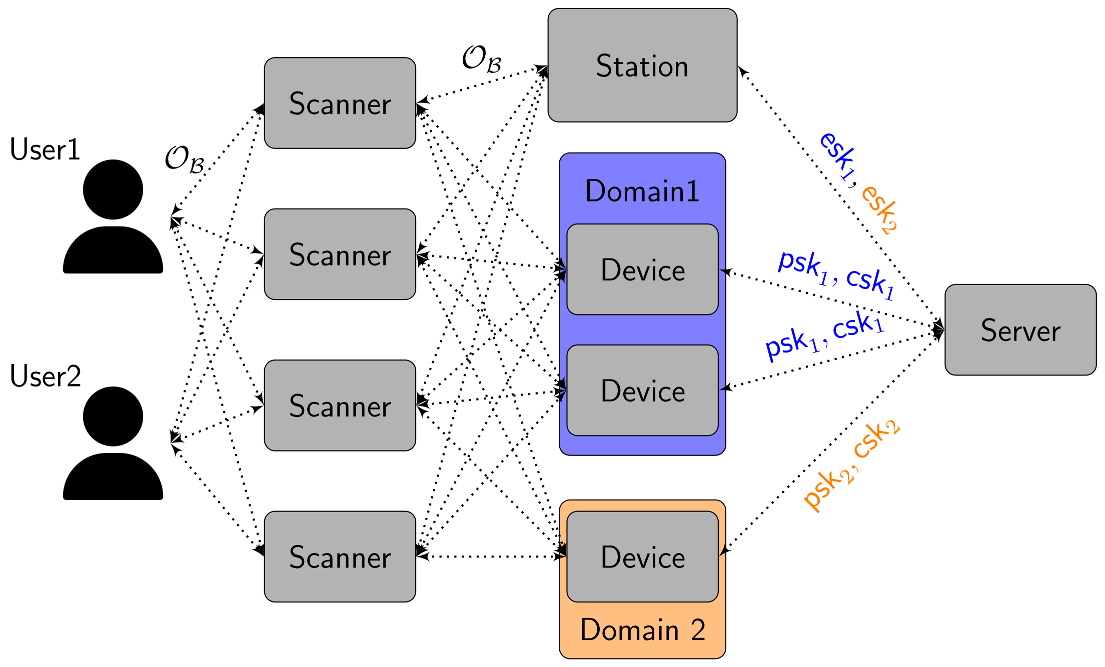


Figure 4: An overview of the Device-of-Domain Usage Model

- 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.

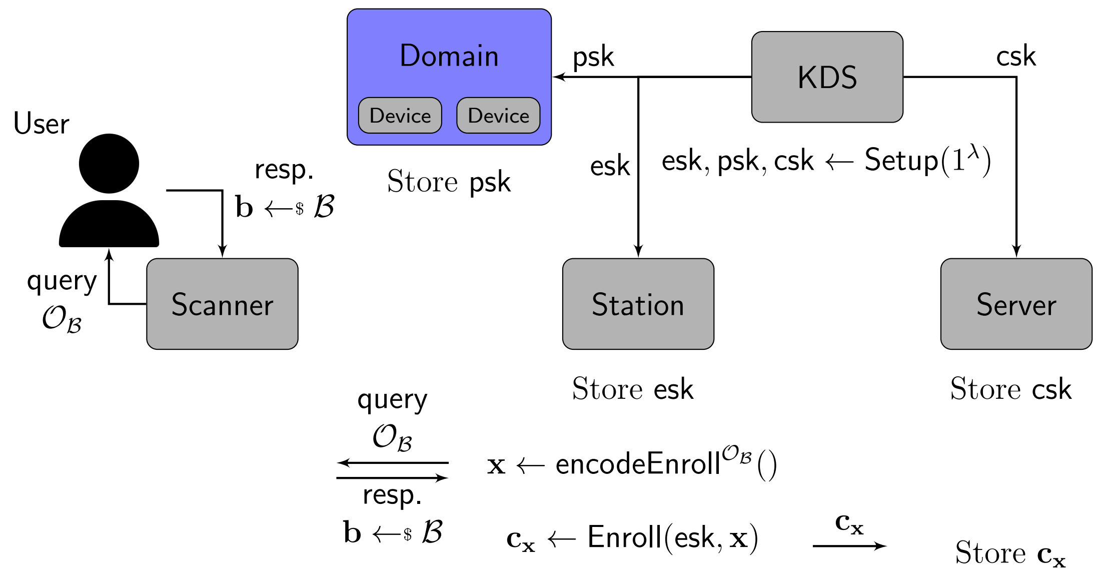

Domain
Device
Device
psk
KDS
csk
User
resp.
$\mathbf{b} \leftarrow_\$ \mathcal{B}$
Store psk
esk
esk, psk, csk $\leftarrow$ Setup$(1^\lambda)$
query
$\mathcal{O}_\mathcal{B}$
Scanner
Station
Server
Store esk
Store csk
query
$\mathcal{O}_\mathcal{B}$
$\mathbf{x} \leftarrow$ encodeEnroll$^{\mathcal{O}_\mathcal{B}}()$
resp.
$\mathbf{b} \leftarrow_\$ \mathcal{B}$
$\mathbf{c_x} \leftarrow$ Enroll(esk, $\mathbf{x}$)
$\mathbf{c_x}$
Store $\mathbf{c_x}$

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

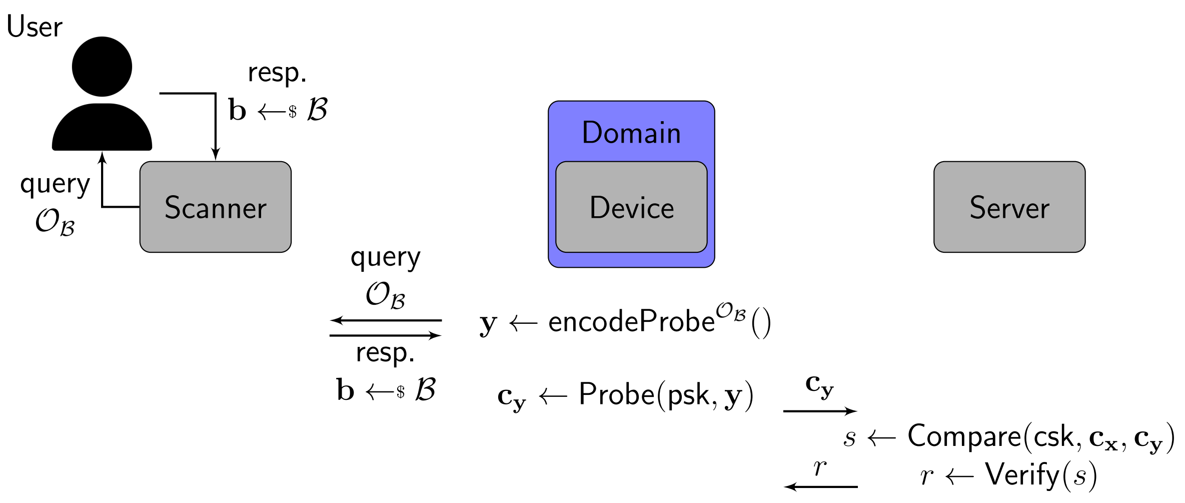

User
resp.
$\mathbf{b} \leftarrow_\$ \mathcal{B}$
Domain
Device
Server
query
$\mathcal{O}_\mathcal{B}$
Scanner
query
$\mathcal{O}_\mathcal{B}$
$\mathbf{y} \leftarrow$ encodeProbe$^{\mathcal{O}_\mathcal{B}}()$
resp.
$\mathbf{b} \leftarrow_\$ \mathcal{B}$
$\mathbf{c_y} \leftarrow$ Probe(psk, $\mathbf{y}$)
$\mathbf{c_y}$
$s \leftarrow$ Compare(csk, $\mathbf{c_x}$, $\mathbf{c_y}$)
$r$
$r \leftarrow$ Verify$(s)$

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 $\mathsf{FE}$ with the distance metric the Euclidean distance. Let the biometric distribution $\mathcal{B} \subseteq [m]^k$, and let the associated field of $\mathsf{FE}$ 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 $\sqrt{s} \leq \tau$, 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 $\mathsf{FE}$, 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 $\|\mathbf{b} - \mathbf{b}'\| \leq \tau$, the scheme results in $r = 1$, a successful authentication.

Instantiated with an fh-IPFE scheme in this way, the comparison secret key $\mathsf{csk}$ is public, and the enrollment secret key $\mathsf{esk}$ and probe secret key $\mathsf{psk}$ are the same. Anyone with access to the enrollment message $\mathbf{c_x}$ and either one of $\mathsf{esk}$, $\mathsf{psk}$, 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 $\mathbf{c_{y^*}}$ such that $\mathbf{x}\mathbf{y}^{*T} \leq \tau$ to impersonate the user by sampling many times offline.

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

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 $\mathsf{Server}$, who has the enrollment message $\mathbf{c_x}$, can utilize this attack to retrieve information about $\mathsf{User}$.

## 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 $\mathsf{FE}$.

- $\mathsf{Setup}(1^\lambda)$: 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 $\sqrt{s} \leq \tau$, 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 $\mathsf{FE}$, 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 $\mathsf{csk}$ is now secret, and the enrollment secret key $\mathsf{esk}$ and probe secret key $\mathsf{psk}$ are distinct. Without $\mathsf{csk}$, 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 $\mathsf{Server}$ is malicious, then it can use $\mathsf{csk}$ 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 $\mathsf{FE}$.

- $\mathsf{Setup}(1^\lambda)$: It calls $\mathsf{FE.Setup}(1^\lambda) \rightarrow \mathsf{sk}, \mathsf{ek}_1, \mathsf{ek}_2$, $\mathsf{FE.KeyGen}(sk, \mathbf{I}_{k+2}) \rightarrow \mathsf{dk}_{\mathbf{I}}$, where $\mathbf{I}_{k+2}$ is an identity matrix of size $(k+2) \times (k+2)$. For keys used for $\mathsf{Domain}$ $\ell$, 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 $\mathsf{Setup}$ is run once for each domain to generate different $\mathsf{esk}, \mathsf{psk}, \mathsf{csk}$. In the scheme in this section, however, $\mathsf{Setup}$ is run only once for all the domains, and each domain shares the same $\mathsf{csk}$ 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 $\mathsf{FE.Enc}(\ell, \mathsf{ek}_1, \mathbf{x}) \rightarrow \mathbf{c_x}$ and outputs $\mathbf{c_x}$.

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

- $\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$: It calls $\mathsf{FE.Dec}(\mathsf{dk}_{\mathbf{I}}, \mathbf{c_x}, \mathbf{c_y}) \rightarrow s$ and outputs the value $s$.

- $\mathsf{Verify}(s)$: If $\sqrt{s} \leq \tau$, 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 $\mathsf{FE}$, 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 $\mathsf{esk}$ and probe secret key $\mathsf{psk}$ are now shared among all the devices, regardless of their domains. Therefore, to let a malicious or broken $\mathsf{Domain}$ not threaten other honest ones, one needs to make sure given $\mathsf{esk}$ or $\mathsf{psk}$, $\mathbf{c_x}$ 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 $\mathsf{Server}$ and $\mathsf{Domain}$ are both malicious, then the adversary can use $\mathsf{csk}$ 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 $\mathsf{Server}$ and $\mathsf{Domain}$ can be malicious.

## 2.6 Instantiation with a Relational Hash Scheme

Let $\mathsf{RH} = (\mathsf{RH.KeyGen}, \mathsf{RH.Hash}_1, \mathsf{RH.Hash}_2, \mathsf{RH.Verify})$ be a relational hash scheme we defined in Definition 4 for the relation $R$ of Hamming distance proximity parametrized by a constant $\tau$.

$$R = \{(\mathbf{x}, \mathbf{y}) \mid \mathsf{HD}(\mathbf{x}, \mathbf{y}) \leq \tau \wedge \mathbf{x}, \mathbf{y} \in \{0,1\}^k\}$$

Note that here we ignore the third parameter $Z$. Following [EM23] and [MR14], we can instantiate a biometric authentication scheme using $\mathsf{RH}$. Let the biometric distribution $\mathcal{B} \subseteq \{0,1\}^k$.

- $\mathsf{Setup}(1^\lambda)$: It calls $\mathsf{RH.Setup}(1^\lambda) \to \mathsf{pk}$ and outputs $\mathsf{esk} \leftarrow \mathsf{pk}$, $\mathsf{psk} \leftarrow \mathsf{pk}$, and $\mathsf{csk} \leftarrow \mathsf{pk}$.
- $\mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$: For a template vector $\mathbf{b}$ sampled from $\mathcal{O}_\mathcal{B}$, it direclty outputs $\mathbf{x} \leftarrow \mathbf{b}$.
- $\mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$: It calls $\mathsf{RH.Hash}_1(\mathsf{pk}, \mathbf{x}) \to \mathbf{h_x}$ and outputs $\mathbf{c_x} \leftarrow \mathbf{h_x}$.
- $\mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$: For a template vector $\mathbf{b}'$ sampled from $\mathcal{O}_\mathcal{B}$, it directy outputs $\mathbf{y} \leftarrow \mathbf{b}'$.
- $\mathsf{Probe}(\mathsf{psk}, \mathbf{y})$: It calls $\mathsf{RH.Hash}_2(\mathsf{pk}, \mathbf{y}) \to \mathbf{h_y}$ and outputs $\mathbf{c_y} \leftarrow \mathbf{h_y}$.
- $\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{c_y})$: It calls $\mathsf{RH.Verify}(\mathsf{pk}, \mathbf{h_x}, \mathbf{h_y}) \to s$ and outputs the value $s$.
- $\mathsf{Verify}(s)$: It direclty returns $r \leftarrow s$.

By the correctness of the relational hash scheme $\mathsf{RH}$, we have (except for a negligible probability),

$$r = 1 \Leftrightarrow (\mathbf{x}, \mathbf{y}) \in R \Leftrightarrow \mathsf{HD}(\mathbf{b}, \mathbf{b}') \leq \tau$$

## 3 Security Games

To rigorously analyze the security of an authentication scheme, we simulate biometric distributions of users by assuming the existence of a family $\mathbb{B}$ of distributions. We require that all distributions in $\mathbb{B}$ are efficiently samplable and has an excessively large size for a PPT adversary to enumerate. We then provide interfaces for all algorithms to interact with $\mathbb{B}$.

- $\mathsf{BioSamp}()$: Generate a random distribution $\mathcal{B}$ of $\mathbb{B}$. By this we mean providing either parameters of an efficiently samplable distribution or a PPT algorithm as the sampler. For simplicity, we write $\mathcal{B} \leftarrow \mathsf{BioSamp}()$ as $\mathcal{B} \leftarrow_\$ \mathbb{B}$.
- $\mathsf{BioDelete}(\mathcal{B})$: Delete $\mathcal{B}$ from $\mathbb{B}$. Consequently, no further access to $\mathsf{BioSamp}$ can derive $\mathcal{B}$. For simplicity, we write $\mathsf{BioDelete}(\mathcal{B})$ as $\mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$.
- $\mathcal{O}_{\mathsf{samp}}(\cdot)$: On input an index $i$,
  - If $i$ was not queried before, it first samples a biometric distribution $\mathcal{B}_i \in \mathbb{B}$ by $\mathsf{BioSamp}$ 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

To describe the unforgeability of an authentication scheme, we model the ability of an adversary who tries to impersonate $\mathsf{User}$. The adversary $\mathcal{A}$ is given the sampling oracle $\mathcal{O}_{\mathsf{samp}}$ and auxiliary information $\mathsf{option}$, which depends on our threat model. The adversary tries to find a valid probe message $\tilde{\mathbf{z}}$. The whole game $\mathsf{UF}_{\Pi,\mathbb{B},\mathsf{option}}$ is defined in Algorithm 1.

**Algorithm 1** $\mathsf{UF}_{\Pi,\mathbb{B},\mathsf{option}}(\mathcal{A})$

```
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: for $i = 1$ to $L$ do
4:     $\mathsf{esk}_i, \mathsf{psk}_i, \mathsf{csk}_i \leftarrow \mathsf{Setup}(1^\lambda)$        ▷ Keys for Domain $i \in [L]$
5: end for
6: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
7: $\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
8: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{samp}}}(\mathsf{option})$
9: $s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \tilde{\mathbf{z}})$
10: return $\mathsf{Verify}(s)$
```

The auxiliary information $\mathsf{option}$ can be nothing or include $\mathbf{c_x}, \mathsf{esk}, \mathsf{psk}, \mathsf{esk}_i, \mathsf{psk}_i$ for $i \in [L]$ or the following oracles:

- $\mathcal{O}_\mathcal{B}$: It outputs a biometric sample $\mathbf{b} \leftarrow_{\$} \mathcal{B}$.
- $\mathcal{O}_{\mathsf{Enroll}}(\mathsf{esk}, \cdot)$: 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 $\mathsf{Probe}(\mathsf{psk}, \mathbf{y}')$. If this oracle and $\mathcal{O}_\mathcal{B}$ are given at the same time, we require the adversary to return some $\tilde{\mathbf{z}}$ that is not equal to any previous answer of $\mathcal{O}_{\mathsf{Probe}}$.
- $\mathcal{O}^q_{\mathsf{log}}(\mathsf{csk}, \mathbf{c_x}, \cdot)$: 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 $\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c_x}, \mathbf{z}))$. We omit $q$ in the superscript when we allow unbounded number of queries.
- $\mathcal{O}^q_{\mathsf{auth}}(\mathsf{csk}, \cdot, \cdot)$: This is a resource-limited oracle. If it has been queried over $q$ times in total, it aborts. Otherwise, on input $\tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}}$, it outputs $\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}}))$. We omit $q$ in the superscript when we allow unbounded number of queries.
- $\mathcal{O}^{(i)}_{\mathsf{Enroll}}(\mathsf{esk}_i, \cdot)$: On input $\mathbf{x}'$, it outputs the enrollment message $\mathsf{Enroll}(\mathsf{esk}_i, \mathbf{x}')$.
- $\mathcal{O}^{(i)}_{\mathsf{Probe}}(\mathsf{psk}_i, \cdot)$: On input $\mathbf{y}'$, it outputs the probe message $\mathsf{Probe}(\mathsf{psk}_i, \mathbf{y}')$.
- $\mathcal{O}'_{\mathsf{Enroll}}(\cdot)$: On input $\mathsf{esk}'$, it first samples $\mathbf{x}' \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$ and outputs $\mathsf{Enroll}(\mathsf{esk}', \mathbf{x}')$.
- $\mathcal{O}'_{\mathsf{Probe}}(\cdot)$: On input $\mathsf{psk}'$, it first samples $\mathbf{y}' \leftarrow \mathsf{encodeProbe}^{\mathcal{O}_\mathcal{B}}()$ and outputs $\mathsf{Probe}(\mathsf{psk}', \mathbf{y}')$. This oracle and $\mathsf{psk}$ should not be given at the same time.

If $\mathsf{option}$ does not include $\mathcal{O}_{\mathsf{Probe}}$, we define the advantage of an adversary $\mathcal{A}$ with $\mathsf{option}$ in the $\mathsf{UF}$ game of a scheme $\Pi$ associated with a family of distributions $\mathbb{B}$ as

$$\mathbf{Adv}^{\mathsf{UF}}_{\Pi,\mathbb{B},\mathcal{A},\mathsf{option}} := \Pr[\mathsf{UF}_{\Pi,\mathbb{B},\mathsf{option}}(\mathcal{A}) \to 1]$$

When $\mathsf{option}$ includes $\mathsf{psk}$ or $\mathcal{O}_{\mathsf{Probe}}$, to consider potential non-negligible false positive rates of biometrics match, we define the plain $\mathsf{UF}'$ game in Algorithm 2, where the adversary has only $\mathcal{O}_{\mathsf{samp}}$ and $\mathcal{O}^{q}_{\mathsf{log}}$ and tries to find a vector $\tilde{\mathbf{z}}$ close to $\mathbf{x}$.

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

```
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: $\mathbf{c}_{\mathbf{x}} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
5: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}_{\mathsf{samp}}, \mathcal{O}^{q}_{\mathsf{log}}}()$
6: $\tilde{\mathbf{c}_{\mathbf{z}}} \leftarrow \mathsf{Probe}(\mathsf{psk}, \tilde{\mathbf{z}})$
7: $s \leftarrow \mathsf{Compare}(\mathsf{csk}, \mathbf{c}_{\mathbf{x}}, \tilde{\mathbf{c}_{\mathbf{z}}})$
8: **return** $\mathsf{Verify}(s)$
```

The advantage of an adversary $\mathcal{A}$ is defined as

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

An authentication scheme $\Pi$ associated with a family $\mathbb{B}$ of distributions is called *option-unforgeable* ($\mathsf{option}$-UF) if for any PPT adversary $\mathcal{A}$,

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

Note that if a $\mathsf{UF}$ game adversary can simulate a $\mathsf{UF}'$ game and $\mathsf{csk}$ is empty or public, the scheme cannot achieve UF security when the false positive rate is not negligible, as the adversary can run $\mathsf{Verify}(\mathsf{Compare}(\mathsf{csk}, \mathbf{c}_{\mathbf{x}}, \cdot))$ over $q$ times to boost the false positive rates.

For the rest of this report, if the scheme, the family distribution, and the auxiliary information $\mathsf{option}$ are clear from context, we omit the subscript and write the game as $\mathsf{UF}(\mathcal{A})$. This abbreviation also holds for all other games.

## 3.2 Indistinguishable against Malicious Server (IND-MSV)

In the game of Indistinguishable against Malicious Server, we model the ability of a malicious $\mathsf{Server}$ 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 $\mathsf{csk}$, an enrollment message $\mathbf{c}_{\mathbf{x}}$, and a list of $t$ probe messages $\{\mathbf{c}_{\mathbf{y}}{}^{(i)}\}_{i=1}^{t}$ . It tries to guess from either $\mathcal{B}^{(0)}$ or $\mathcal{B}^{(1)}$ these messages are generated. The whole game is defined in Algorithm 3.

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

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

**Algorithm 3** $\mathsf{IND\text{-}MSV}_{\Pi,\mathbb{B}}(\mathcal{A})$

```
1: $b \leftarrow_{\$} \{0,1\}$
2: $\mathcal{B}^{(0)} \leftarrow_{\$} \mathbb{B}, \quad \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}^{(0)}$
3: $\mathcal{B}^{(1)} \leftarrow_{\$} \mathbb{B}, \quad \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}^{(1)}$
4: $\mathsf{esk}, \mathsf{psk}, \mathsf{csk} \leftarrow \mathsf{Setup}(1^\lambda)$
5: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}^{(b)}}}()$
6: $\mathbf{c_x} \leftarrow \mathsf{Enroll}(\mathsf{esk}, \mathbf{x})$
7: for $i = 1$ to $t$ do
8:     $\mathbf{y}^{(i)} \leftarrow \mathsf{encodeProbe}^{\mathcal{O}_{\mathcal{B}^{(b)}}}()$
9:     $\mathbf{c_y}^{(i)} \leftarrow \mathsf{Probe}(\mathsf{psk}, \mathbf{y}^{(i)})$
10: end for
11: $\tilde{b} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{B}^{(0)}}, \mathcal{O}_{\mathcal{B}^{(1)}}}(\mathsf{csk}, \mathbf{c_x}, \{\mathbf{c_y}^{(i)}\}_{i=1}^{t})$
12: return $1_{\tilde{b}=b}$
```

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

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

## 4 Security Analysis

Given an fh-IPFE scheme $\mathsf{FE}$, we define the $\mathsf{fh\text{-}IND}$ game in algorithm 4.

**Algorithm 4** $\mathsf{fh\text{-}IND}_{\mathsf{FE}}(\mathcal{A})$

```
1: $b \leftarrow_{\$} \{0,1\}$
2: $\mathsf{msk}, \mathsf{pp} \leftarrow \mathsf{FE.Setup}(1^\lambda)$
3: $\tilde{b} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Enc}}}(\mathsf{pp})$
4: return $1_{\tilde{b}=b}$
```

- $\mathcal{O}_{\mathsf{KeyGen}}(\cdot,\cdot)$: On input pair $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, it outputs $\mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pp}, \mathbf{x}^{(b)})$.
- $\mathcal{O}_{\mathsf{Enc}}(\cdot,\cdot)$: On input pair $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$, it outputs $\mathsf{FE.Enc}(\mathsf{msk}, \mathsf{pp}, \mathbf{y}^{(b)})$.

To avoid trivial attacks, we consider *admissible adversaries.*

**Definition 5** (Admissible Adversary)**.** Let $\mathcal{A}$ be an adversary in an $\mathsf{fh\text{-}IND}$ game, and let $(\mathbf{x}_1^{(0)}, \mathbf{x}_1^{(1)}), \cdots, (\mathbf{x}_{Q_K}^{(0)}, \mathbf{x}_{Q_K}^{(1)})$ be its queries to $\mathcal{O}_{\mathsf{KeyGen}}$ and $(\mathbf{y}_1^{(0)}, \mathbf{y}_1^{(1)}), \cdots, (\mathbf{y}_{Q_E}^{(0)}, \mathbf{y}_{Q_E}^{(1)})$ be its queries to $\mathcal{O}_{\mathsf{Enc}}$. We say $\mathcal{A}$ is *admissible* if $\forall i \in [Q_K], \forall j \in [Q_E]$,

$$\mathbf{x}_i^{(0)} \mathbf{y}_j^{(0)^T} = \mathbf{x}_i^{(1)} \mathbf{y}_j^{(1)^T}$$

**Definition 6** (fh-IND Security)**.** An fh-IPFE scheme $\mathsf{FE}$ is called fh-IND secure if for any admissible adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the $\mathsf{fh\text{-}IND}$ game in Algorithm 4 is

$$\mathbf{Adv}^{\mathsf{fh\text{-}IND}}_{\mathsf{FE},\mathcal{A}} := \left|\Pr[\mathsf{fh\text{-}IND}_{\mathsf{FE}}(\mathcal{A}) \to 1] - \frac{1}{2}\right| = \mathsf{negl}.$$

We also define the $\mathsf{RUF}$ game in algorithm 5 for a real number $\gamma$.

**Algorithm 5** $\mathsf{RUF}^{\gamma}_{\mathsf{FE}}(\mathcal{A})$

```
1: $\mathbf{r} \leftarrow_{\$} \mathbb{F}^{k+2}$
2: $\mathsf{msk}, \mathsf{pp} \leftarrow \mathsf{FE.Setup}(1^{\lambda})$
3: $\mathbf{c} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pp}, \mathbf{r})$
4: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}'_{\mathsf{KeyGen}}, \mathcal{O}'_{\mathsf{Enc}}}(\mathsf{pp}, \mathbf{c})$
5: if $\tilde{\mathbf{z}}$ is equal to any output of $\mathcal{O}'_{\mathsf{Enc}}$ then
6:     return 0
7: end if
8: $s \leftarrow \mathsf{FE.Dec}(\mathsf{pp}, \mathbf{c}, \tilde{\mathbf{z}})$
9: return $1_{s \leq \gamma}$
```

- $\mathcal{O}'_{\mathsf{KeyGen}}(\cdot)$: On input $\mathbf{x}'$, it outputs $\mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pp}, \mathbf{x}')$.
- $\mathcal{O}'_{\mathsf{Enc}}(\cdot)$: On input $\mathbf{y}'$, it outputs $\mathsf{FE.Enc}(\mathsf{msk}, \mathsf{pp}, \mathbf{y}')$. The adversary is required to return $\tilde{\mathbf{z}}$ that is not equal to any output of this oracle.

**Definition 7** (RUF Security)**.** An fh-IPFE scheme $\mathsf{FE}$ is called RUF secure for a real number $\gamma$ if for any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the $\mathsf{RUF}$ game in Algorithm 5 is

$$\mathbf{Adv}^{\mathsf{RUF},\gamma}_{\mathsf{FE},\mathcal{A}} := \Pr[\mathsf{RUF}^{\gamma}_{\mathsf{FE}}(\mathcal{A}) \to 1] = \mathsf{negl}.$$

## 4.1 fh-IND Security and UF Security

Let $\Pi$ be the authentication scheme instantiated by an fh-IPFE scheme $\mathsf{FE}$ as in Section 2.3.

**Theorem 1.** *Let* $\mathsf{option} = \{\mathbf{c}_{\mathbf{x}}, \mathcal{O}_{\mathcal{B}}, \mathcal{O}_{\mathsf{auth}}, \mathcal{O}_{\mathsf{Enroll}}\}$*. For any distribution family* $\mathbb{B}$*, if* $\mathsf{FE}$ *is fh-IND secure and RUF secure for a* $\gamma \geq \tau^2$*, then* $\Pi$ *is* $\mathsf{option}$*-unforgeable.*

*Proof.* Given an adversary $\mathcal{A}$ in the $\mathsf{UF}_{\mathsf{option}}$ game, consider the reduction adversary $\mathcal{R}$ in Algorithm 6 which plays the $\mathsf{fh\text{-}IND}$ game. $\mathcal{R}$ runs $\mathcal{A}$ and simulates each oracle in the following way.

- $\mathcal{O}_{\mathsf{samp}}(i)$: This is sampled by $\mathcal{O}_{\mathsf{samp}}(i)$ of $\mathcal{R}$.
- $\mathcal{O}_{\mathcal{B}}$: This is simulated by the distribution $\mathcal{B}$ generated in Line 1.

- $\mathcal{O}_{\mathsf{auth}}(\mathsf{csk}, \tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}})$: This is simulated by calling $\mathsf{Verify}(\mathsf{FE.Dec}(\mathsf{pp}, \tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}}))$.
- $\mathcal{O}_{\mathsf{Enroll}}(\mathsf{esk}, \mathbf{x}')$: This is simulated by $\mathcal{O}_{\mathsf{KeyGen}}(\mathbf{x}', \mathbf{x}')$ given in the fh-IND game.

Note that since $\mathcal{R}$ never calls $\mathcal{O}_{\mathsf{Enc}}$, it is an admissible adversary.

**Algorithm 6** $\mathcal{R}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Enc}}}(\mathsf{pp})$

```
1: $\mathcal{B} \leftarrow_\$ \mathbb{B}, \quad \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
2: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_\mathcal{B}}()$
3: $\mathbf{r} \leftarrow_\$ \mathbb{F}^{k+2}$
4: $\mathbf{c} \leftarrow \mathcal{O}_{\mathsf{KeyGen}}(\mathbf{x}, \mathbf{r})$
5: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{samp}}, \mathcal{O}_\mathcal{B}, \mathcal{O}_{\mathsf{auth}}, \mathcal{O}_{\mathsf{Enroll}}}(\mathbf{c})$
6: $s \leftarrow \mathsf{FE.Dec}(\mathsf{pp}, \mathbf{c}, \tilde{\mathbf{z}})$
7: if $\mathsf{Verify}(s) = 1$ then
8:     return $\tilde{b} = 0$
9: else
10:     return $\tilde{b} \leftarrow_\$ \{0, 1\}$
11: end if
```

If the challenge bit $b = 0$, then $\mathcal{R}$ perfectly simulates a $\mathsf{UF}_{\mathsf{option}}$ game for $\mathcal{A}$. Therefore, the probability that $\mathsf{Verify}(s) = 1$ in Line 7 is $\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1]$.

For the case when the challenge bit $b = 1$, consider an adversary $\mathcal{A}'$ in Algorithm 7 in the RUF game. $\mathcal{A}'$ runs $\mathcal{A}$ and simulates each oracle in the following way.

- $\mathcal{O}_{\mathsf{samp}}(i)$: This is sampled by $\mathcal{O}_{\mathsf{samp}}(i)$ of $\mathcal{A}'$.
- $\mathcal{O}_\mathcal{B}$: This is simulated by the oracle $\mathcal{O}_{\mathsf{samp}}(i^*)$, where $i^*$ is an index that is never queried by $\mathcal{A}$ in $\mathcal{O}_{\mathsf{samp}}$.
- $\mathcal{O}_{\mathsf{auth}}(\mathsf{csk}, \tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}})$: This is simulated by calling $\mathsf{Verify}(\mathsf{FE.Dec}(\mathsf{pp}, \tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}}))$.
- $\mathcal{O}_{\mathsf{Enroll}}(\mathsf{esk}, \mathbf{x}')$: This is simulated by $\mathcal{O}'_{\mathsf{KeyGen}}(\mathbf{x}')$ given in the RUF game.

**Algorithm 7** $\mathcal{A}'^{\mathcal{O}'_{\mathsf{KeyGen}}, \mathcal{O}'_{\mathsf{Enc}}}(\mathsf{pp}, \mathbf{c})$

```
1: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{samp}}, \mathcal{O}_{\mathsf{samp}}(i^*), \mathcal{O}_{\mathsf{auth}}, \mathcal{O}'_{\mathsf{KeyGen}}}(\mathbf{c})$
2: return $\tilde{\mathbf{z}}$
```

Now, if the challenge bit $b = 1$, then $\mathcal{R}$ perfectly simulates $\mathcal{A}'$ in the RUF game. The probability that $\mathsf{Verify}(s) = 1$, which is equivalent to $s \leq \tau^2$, in Line 7 is $\Pr[\mathsf{RUF}_{\mathsf{FE}}^{\tau^2}(\mathcal{A}) \to 1]$

In conclusion, since $\gamma \geq \tau^2$,

$$\begin{aligned}\Pr[\mathsf{fh\text{-}IND}(\mathcal{R}) \to 1] = \Pr[b=0] \cdot \Big(&\Pr[\mathsf{Verify}(s)=1 \mid b=0] + \frac{1}{2}\Pr[\mathsf{Verify}(s)=0 \mid b=0]\Big) \\ &+ \Pr[b=1] \cdot \frac{1}{2}\Pr[\mathsf{Verify}(s)=0 \mid b=1] \\ &= \frac{1}{2} + \frac{1}{4}\left(\Pr[\mathsf{Verify}(s)=1 \mid b=0] - \Pr[\mathsf{Verify}(s)=1 \mid b=1]\right) \\ &= \frac{1}{2} + \frac{1}{4}\Big(\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1] - \Pr[\mathsf{RUF}^{\tau^2}_{\mathsf{FE}}(\mathcal{A}) \to 1]\Big) \\ &\geq \frac{1}{2} + \frac{1}{4}\left(\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1] - \Pr[\mathsf{RUF}^{\gamma}_{\mathsf{FE}}(\mathcal{A}) \to 1]\right)\end{aligned}$$

Since both $\mathbf{Adv}^{\mathsf{fh\text{-}IND}}_{\mathsf{FE},\mathcal{R}} = \left|\Pr[\mathsf{fh\text{-}IND}(\mathcal{R}) \to 1] - \frac{1}{2}\right|$ and $\mathbf{Adv}^{\mathsf{RUF},\gamma}_{\mathsf{FE},\mathcal{A}} = \Pr[\mathsf{RUF}^{\gamma}_{\mathsf{FE}}(\mathcal{A}) \to 1]$ are negligilbe,

$$\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1] \leq 4 \cdot \mathbf{Adv}^{\mathsf{fh\text{-}IND}}_{\mathsf{FE},\mathcal{R}} + \mathbf{Adv}^{\mathsf{RUF},\gamma}_{\mathsf{FE},\mathcal{A}} = \mathsf{negl}.$$

□

For option that includes $\mathcal{O}_{\mathsf{Probe}}$, we first note that for any $d \in \mathbb{Z}_q$ and any nonzero vector $\mathbf{r} \in \mathbb{Z}_q^{k+2}$, there exists a vector $\mathbf{y}$ such that $\mathbf{r}\mathbf{y}^T = d$.

**Theorem 2.** *Let option* $= \{\mathbf{c_x}, \mathcal{O}_{\mathcal{B}}, \mathcal{O}_{auth}, \mathcal{O}_{Probe}\}$. *For any distribution family* $\mathbb{B}$, *if FE is fh-IND secure and RUF secure for a* $\gamma \geq \tau^2$, *then* $\Pi$ *is option-unforgeable.*

*Proof.* Given an adversary $\mathcal{A}$ in the $\mathsf{UF}_{\mathsf{option}}$ game, we will show that

$$\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1] = \mathsf{negl}.$$

Hence, we can ignore the false positive rate $\sup_{\mathrm{PPT}\ \mathcal{A}'} \Pr[\mathsf{UF}'_{\Pi,\mathbb{B}}(\mathcal{A}') \to 1]$.

Consider the reduction adversary $\mathcal{R}$ in Algorithm 8 which plays the fh-IND game. $\mathcal{R}$ runs $\mathcal{A}$ and simulates each oracle in the following way.

- $\mathcal{O}_{\mathsf{samp}}(i)$: This is sampled by $\mathcal{O}_{\mathsf{samp}}(i)$ of $\mathcal{R}$.
- $\mathcal{O}_{\mathcal{B}}$: This is simulated by the distribution $\mathcal{B}$ generated in Line 1.
- $\mathcal{O}^q_{\mathsf{auth}}(\mathsf{csk}, \tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}})$: This is simulated by calling $\mathsf{Verify}(\mathsf{FE.Dec}(\mathsf{pp}, \tilde{\mathbf{c_x}}, \tilde{\mathbf{c_y}}))$.
- $\mathcal{O}_{\mathsf{Probe}}(\mathsf{psk}, \mathbf{y}')$: It first computes $d \leftarrow \mathbf{x}\mathbf{y}'^T$ and finds a vector $\mathbf{y}''$ such that $\mathbf{r}\mathbf{y}''^T = d$. Next, it calls $\mathcal{O}_{\mathsf{Enc}}(\mathbf{y}', \mathbf{y}'')$, which is given by the fh-IND game, and returns the result.

Note that $(\mathbf{x}, \mathbf{r})$ is the only query of $\mathcal{R}$ to $\mathcal{O}_{\mathsf{KeyGen}}$, and for any query $(\mathbf{y}', \mathbf{y}'')$ to $\mathcal{O}_{\mathsf{Enc}}$, it satisfies $\mathbf{x}\mathbf{y}'^T = \mathbf{r}\mathbf{y}''^T$. Hence, $\mathcal{R}$ is an admissible adversary.

If the challenge bit $b = 0$, then $\mathcal{R}$ perfectly simulates a $\mathsf{UF}_{\mathsf{option}}$ game for $\mathcal{A}$. Therefore, the probability that $\mathsf{Verify}(s) = 1$ in Line 9 is $\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1]$.

For the case when the challenge bit $b = 1$, consider an adversary $\mathcal{A}'$ in Algorithm 9 in the RUF game. $\mathcal{A}'$ runs $\mathcal{A}$ and simulates each oracle in the following way.

**Algorithm 8** $\mathcal{R}^{\mathcal{O}_{\mathsf{KeyGen}},\mathcal{O}_{\mathsf{Enc}}}(\mathsf{pp})$

```
1: $\mathcal{B} \leftarrow_{\$} \mathbb{B}, \quad \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$
2: $\mathbf{x} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$
3: $\mathbf{r} \leftarrow_{\$} \mathbb{F}^{k+2}$
4: $\mathbf{c} \leftarrow \mathcal{O}_{\mathsf{KeyGen}}(\mathbf{x}, \mathbf{r})$
5: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{samp}},\mathcal{O}_{\mathcal{B}},\mathcal{O}^q_{\mathsf{auth}},\mathcal{O}_{\mathsf{Probe}}}(\mathbf{c})$
6: if $\tilde{\mathbf{z}}$ is equal to any output of $\mathcal{O}_{\mathsf{Probe}}$ then
       return $\perp$
7: end if
8: $s \leftarrow \mathsf{FE.Dec}(\mathsf{pp}, \mathbf{c}, \tilde{\mathbf{z}})$
9: if $\mathsf{Verify}(s) = 1$ then
10:     return $\tilde{b} = 0$
11: else
12:     return $\tilde{b} \leftarrow_{\$} \{0, 1\}$
13: end if
```

- $\mathcal{O}_{\mathsf{samp}}(i)$: This is sampled by $\mathcal{O}_{\mathsf{samp}}(i)$ of $\mathcal{A}'$.
- $\mathcal{O}_{\mathcal{B}}$: This is simulated by the oracle $\mathcal{O}_{\mathsf{samp}}(i^*)$, where $i^*$ is an index that is never queried by $\mathcal{A}$ in $\mathcal{O}_{\mathsf{samp}}$.
- $\mathcal{O}^q_{\mathsf{auth}}(\mathsf{csk}, \tilde{c_{\mathbf{x}}}, \tilde{c_{\mathbf{y}}})$: This is simulated by calling $\mathsf{Verify}(\mathsf{FE.Dec}(\mathsf{pp}, \tilde{c_{\mathbf{x}}}, \tilde{c_{\mathbf{y}}}))$.
- $\mathcal{O}_{\mathsf{Probe}}(\mathsf{psk}, \mathbf{y}')$: It first computes $d \leftarrow \mathbf{x}^{(*)}\mathbf{y}'^T$ and finds a vector $\mathbf{y}''$ such that $\mathbf{r}\mathbf{y}''^T = d$. Next, it calls $\mathcal{O}'_{\mathsf{Enc}}(\mathbf{y}'')$ , which is given by the RUF game, and returns the result.

**Algorithm 9** $\mathcal{A}'^{\mathcal{O}'_{\mathsf{KeyGen}},\mathcal{O}'_{\mathsf{Enc}}}(\mathsf{pp}, \mathbf{c})$

```
1: $\mathbf{x}^{(*)} \leftarrow \mathsf{encodeEnroll}^{\mathcal{O}_{\mathsf{samp}}(i^*)}()$
2: Sample $k+2$ linearly independent vectors $\{\mathbf{e}^{(i)}\}_{i=1}^{k+2}$.
3: for $i = 1$ to $k+2$ do
4:     $\mathbf{c}^{(i)} \leftarrow \mathcal{O}'_{\mathsf{Enc}}(\mathbf{e}^{(i)})$.
5:     $d_i \leftarrow \mathsf{FE.Dec}(\mathsf{pp}, \mathbf{c}, \mathbf{c}^{(i)})$.
6: end for
7: Find the vector $\mathbf{r}$ by solving the linear system $\{\mathbf{r}\mathbf{e}^{(i)T} = d_i\}_{i=1}^{k+2}$.
8: if $\mathbf{r} = \mathbf{0}$ then
9:     return $\perp$
10: end if
11: $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{samp}},\mathcal{O}_{\mathsf{samp}}(i^*),\mathcal{O}^q_{\mathsf{auth}},\mathcal{O}_{\mathsf{Probe}}}(\mathbf{c})$
12: return $\tilde{\mathbf{z}}$
```

To make $\mathcal{R}$ simulate $\mathcal{A}'$ in the RUF game, we still need to ensure two conditions.

- $\mathbf{r} \neq \mathbf{0}$. Otherwise, $\mathcal{A}'$ cannot simulate $\mathcal{O}_{\mathsf{Probe}}$.

- $\tilde{\mathbf{z}} \neq \mathbf{c}^{(i)}$ for all $i$. The answers of $\mathcal{O}_{\mathsf{Probe}}$ have already been checked in $\mathcal{R}$.

Let $\mathcal{A}'$ play a tweaked $\mathsf{RUF}_{\mathsf{FE}}^{\tau^2}$ game which does not check that $\tilde{\mathbf{z}}$ is not equal to $\mathbf{c}^{(i)}$ for all $i$. That is, the game only checks whether $\tilde{\mathbf{z}}$ is not equal to any output of $\mathcal{O}'_{\mathsf{Enc}}$ called by $\mathcal{O}_{\mathsf{Probe}}$ of $\mathcal{A}$. Let the returned value of this game be $V$. We have Equation 1 and 2. The former one is a relation between $\mathcal{R}$ playing $\mathsf{fh\text{-}IND}$ game when the challenge bit $b = 1$ and $V$, and the other one is a relation between $\mathcal{A}'$ playing a regular $\mathsf{RUF}_{\mathsf{FE}}^{\tau^2}$ game and the tweaked one.

$$\Pr[\mathsf{Verify}(s) = 1 \mid b = 1 \wedge \mathbf{r} \neq \mathbf{0}] = \Pr[V = 1] \tag{1}$$

$$\Pr[\mathsf{RUF}_{\mathsf{FE}}^{\tau^2}(\mathcal{A}') \to 1] = \Pr\left[V = 1 \mid \bigwedge_{i=1}^{k+2} \tilde{\mathbf{z}} \neq \mathbf{c}^{(i)}\right] \tag{2}$$

For Equation 1, consider that

$$\begin{aligned}
\Pr[\mathsf{Verify}(s) = 1 \mid b = 1] &= \Pr[\mathsf{Verify}(s) = 1 \mid b = 1 \wedge \mathbf{r} \neq \mathbf{0}] \cdot \Pr[\mathbf{r} \neq \mathbf{0}] \\
&\quad + \Pr[\mathsf{Verify}(s) = 1 \mid b = 1 \wedge \mathbf{r} = \mathbf{0}] \cdot \Pr[\mathbf{r} = \mathbf{0}] \\
&\leq \Pr[V = 1] + \Pr[\mathbf{r} = 0] \\
&= \Pr[V = 1] + \frac{1}{q^{k+2}}
\end{aligned}$$

For Equation 2, consider that

$$\begin{aligned}
\Pr[\mathsf{RUF}_{\mathsf{FE}}^{\tau^2}(\mathcal{A}') \to 1] &= \Pr\left[V = 1 \mid \bigwedge_{i=1}^{k+2} \tilde{\mathbf{z}} \neq \mathbf{c}^{(i)}\right] \\
&\geq \Pr[V = 1] - \Pr\left[\neg\left(\bigwedge_{i=1}^{k+2} \tilde{\mathbf{z}} \neq \mathbf{c}^{(i)}\right)\right] \\
&= \Pr[V = 1] - \Pr\left[\bigvee_{i=1}^{k+2} \tilde{\mathbf{z}} = \mathbf{c}^{(i)}\right] \\
&\geq \Pr[V = 1] - \sum_{i=1}^{k+2} \Pr[\tilde{\mathbf{z}} = \mathbf{c}^{(i)}].
\end{aligned}$$

Note that each $\mathbf{c}^{(i)} = \mathsf{FE.Enc}(\mathsf{msk}, \mathsf{pp}, \mathbf{e}^{(i)})$ for some uniform nonzero vector $\mathbf{e}^{(i)}$. Also note that distinct vectors in $\mathbb{Z}_q^{k+2}$ will have different encryptions due to the correctness of $\mathsf{FE}$. Therefore, $\Pr[\tilde{\mathbf{z}} = \mathbf{c}^{(i)}] \leq \frac{1}{q^{k+2}-1}$ and

$$\Pr[\mathsf{RUF}_{\mathsf{FE}}^{\tau^2}(\mathcal{A}') \to 1] \geq \Pr[V = 1] - \frac{k+2}{q^{k+2}-1}.$$

Combining both results from Equation 1 and Equation 2, we derive

$$\Pr[\mathsf{Verify}(s) = 1 \mid b = 1] \leq \Pr[V = 1] + \frac{1}{q^{k+2}} \leq \Pr[\mathsf{RUF}_{\mathsf{FE}}^{\tau^2}(\mathcal{A}') \to 1] + \frac{k+2}{q^{k+2}-1} + \frac{1}{q^{k+2}}.$$

Finally, similar to the proof of Theorem 1, we derive

$$\begin{aligned}\Pr[\mathsf{fh\text{-}IND}(\mathcal{R}) \to 1] &= \frac{1}{2} + \frac{1}{4}\left(\Pr[\mathsf{Verify}(s) = 1 \mid b = 0] - \Pr[\mathsf{Verify}(s) = 1 \mid b = 1]\right) \\ &\geq \frac{1}{2} + \frac{1}{4}\left(\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1] - \Pr[\mathsf{RUF}^{\tau^2}_{\mathsf{FE}}(\mathcal{A}') \to 1] - \frac{k+2}{q^{k+2}-1} - \frac{1}{q^{k+2}}\right) \\ &\geq \frac{1}{2} + \frac{1}{4}\left(\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1] - \Pr[\mathsf{RUF}^{\gamma}_{\mathsf{FE}}(\mathcal{A}') \to 1] - \frac{k+2}{q^{k+2}-1} - \frac{1}{q^{k+2}}\right)\end{aligned}$$

Since both $\mathbf{Adv}^{\mathsf{fh\text{-}IND}}_{\mathsf{FE},\mathcal{R}} = \left|\Pr[\mathsf{fh\text{-}IND}(\mathcal{R}) \to 1] - \frac{1}{2}\right|$ and $\mathbf{Adv}^{\mathsf{RUF},\gamma}_{\mathsf{FE},\mathcal{A}} = \Pr[\mathsf{RUF}^{\gamma}_{\mathsf{FE}}(\mathcal{A}) \to 1]$ are negligible,

$$\Pr[\mathsf{UF}_{\mathsf{option}}(\mathcal{A}) \to 1] \leq 4 \cdot \mathbf{Adv}^{\mathsf{fh\text{-}IND}}_{\mathsf{FE},\mathcal{R}} + \mathbf{Adv}^{\mathsf{RUF},\gamma}_{\mathsf{FE},\mathcal{A}} + \frac{k+2}{q^{k+2}-1} + \frac{1}{q^{k+2}} = \mathsf{negl}.$$

□

## 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.