

# Biometrics Authentication: Formalization and Instantiation

Keng-Yu Chen

November 7, 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  $\{1, 2, \dots, m\}$ .
- $\mathbb{Z}_q$  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  $f(n) = \text{negl}$ , and we may also use  $\text{negl}$  to represent an arbitrary negligible function.
- $\text{poly}$  is the class of polynomial functions. We may also use  $\text{poly}$  to represent an arbitrary polynomial function.
- We write sampling a value  $r$  from a distribution  $\mathcal{D}$  as  $r \leftarrow^s \mathcal{D}$ . If  $S$  is a finite set, then  $r \leftarrow^s S$  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  $\text{FE}$  for a field  $\mathbb{F}$  and input length  $k$  is composed of PPT algorithms  $\text{FE.Setup}$ ,  $\text{FE.KeyGen}$ ,  $\text{FE.Enc}$ , and  $\text{FE.Dec}$ :

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

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

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

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

**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 FE.Setup, FE.KeyGen, FE.Enc, and FE.Dec:

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

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

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

**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 FE.Setup, FE.KeyGen, FE.Enc, and FE.Dec:

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

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

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

## 2 Formalization

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

- $\text{Setup}(1^\lambda) \rightarrow \text{esk}, \text{psk}, \text{csk}$ : It outputs the enrollment secret key  $\text{esk}$ , probe secret key  $\text{psk}$ , and compare secret key  $\text{csk}$ .
- $\text{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}() \rightarrow \mathbf{x}$ : 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.
- $\text{Enroll}(\text{esk}, \mathbf{x}) \rightarrow \mathbf{c}_{\mathbf{x}}$ : It outputs the enrollment message  $\mathbf{c}_{\mathbf{x}}$  from  $\mathbf{x}$ .
- $\text{encodeProbe}^{\mathcal{O}_{\mathcal{B}}}() \rightarrow \mathbf{y}$ : 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.
- $\text{Probe}(\text{psk}, \mathbf{y}) \rightarrow \mathbf{c}_{\mathbf{y}}$ : It outputs the probe message  $\mathbf{c}_{\mathbf{y}}$  from  $\mathbf{y}$ .
- $\text{Compare}(\text{csk}, \mathbf{c}_{\mathbf{x}}, \mathbf{c}_{\mathbf{y}}) \rightarrow s$ : It compares the enrollment message  $\mathbf{c}_{\mathbf{x}}$  and probe message  $\mathbf{c}_{\mathbf{y}}$  and outputs a score  $s$ .
- $\text{Verify}(s) \rightarrow r \in \{0, 1\}$ : 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  $\text{esk}$  and  $\text{psk}$ . 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**.

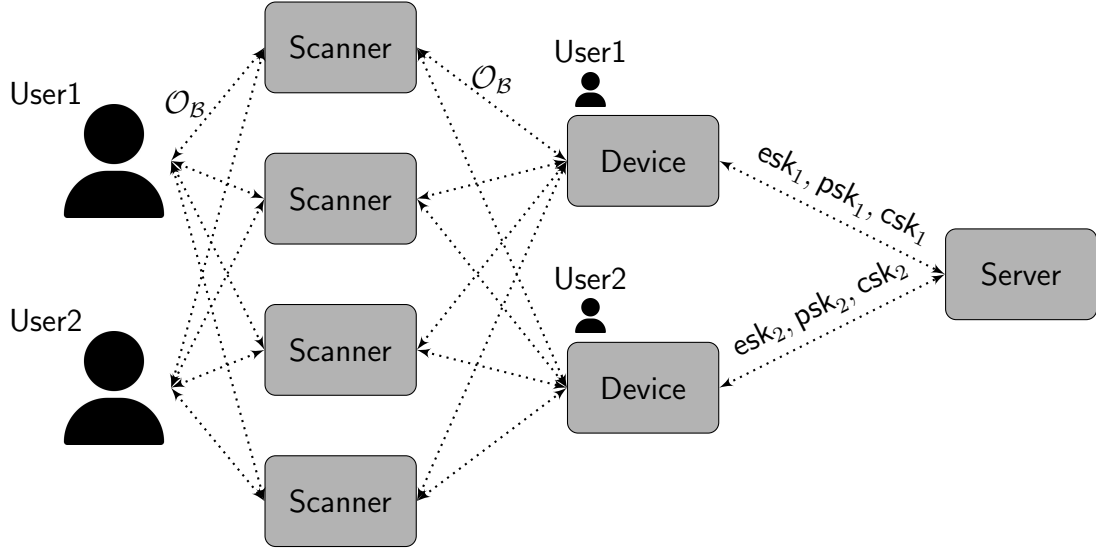


Figure 1: An Overview of the Device-of-User Usage Model

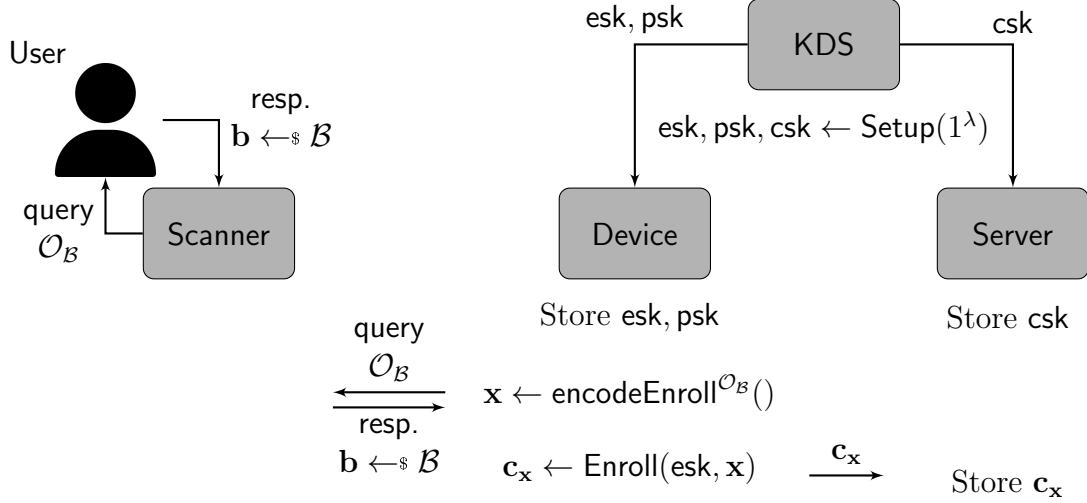


Figure 2: Device-of-User Usage Model on Enrollment

- **Server:** The server responsible for authenticating the user. It stores the comparison key  $\mathbf{c}_{sk}$  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

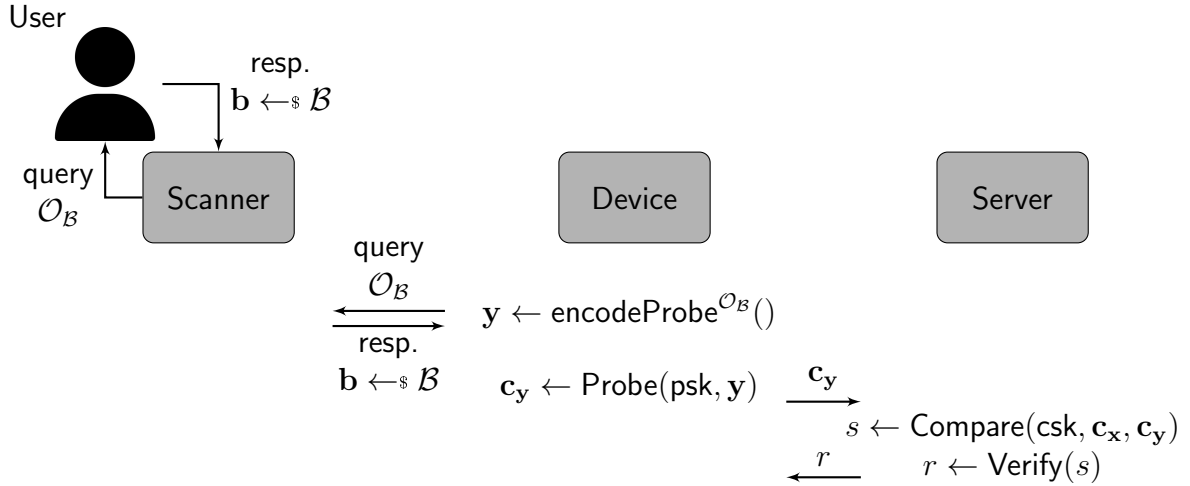


Figure 3: Device-of-User Usage Model on Authentication

keys to the domain, and comparison keys to the server. In practice, a domain can be a department in an organization, and this model 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  $\text{esk}$ , one probe key  $\text{psk}$  and one comparison key  $\text{csk}$ . 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  $\mathcal{O}_B$ .
- **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  $\text{psk}$  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  $\text{csk}$  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.

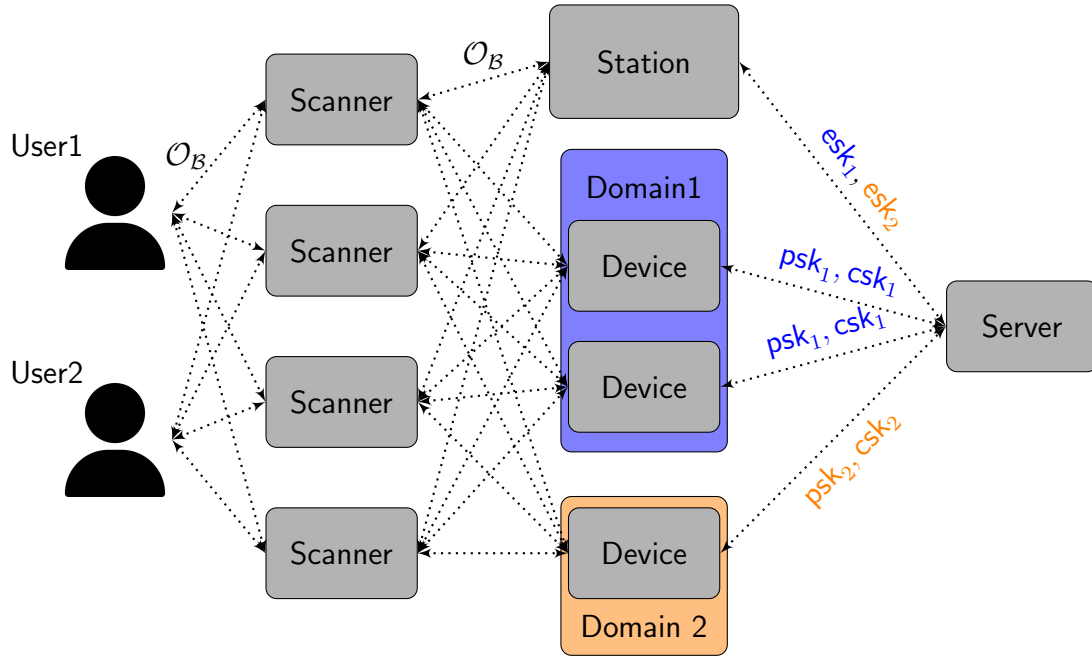


Figure 4: An overview of the Device-of-Domain Usage Model

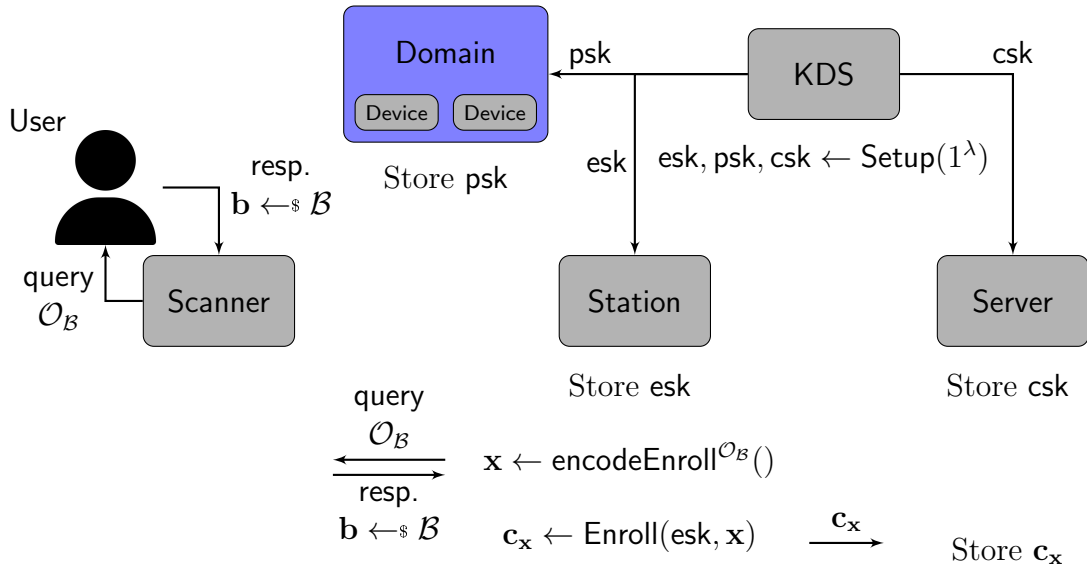


Figure 5: Device-of-Domain Usage Model on Enrollment

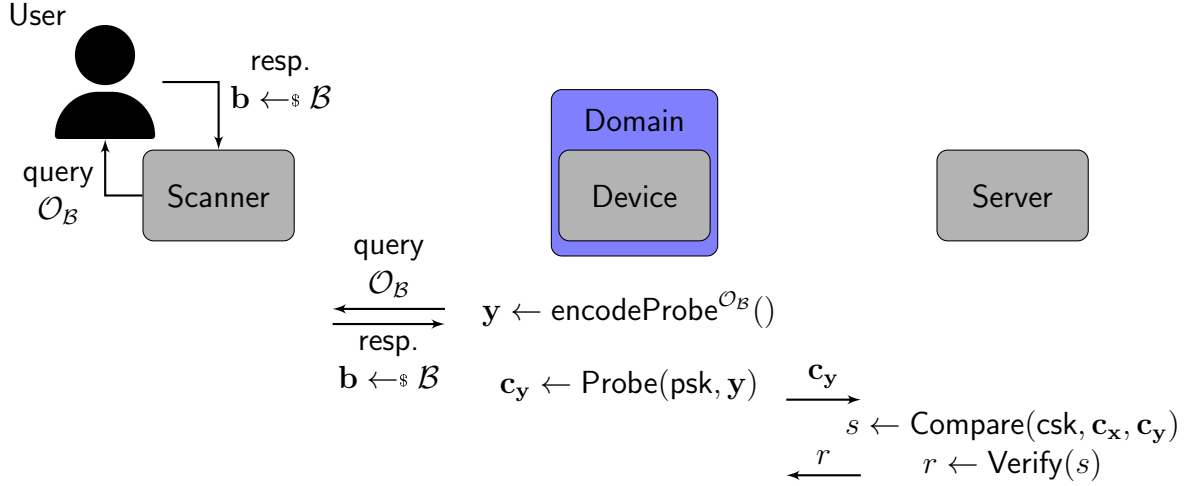


Figure 6: Device-of-Domain Usage Model on Authentication

### 2.3 Instantiation with an fh-IPFE Scheme

Let  $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$  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.

- **Setup**( $1^\lambda$ ): It calls  $\text{FE.Setup}(1^\lambda) \rightarrow \text{msk}, \text{pp}$  and outputs  $\text{esk} \leftarrow (\text{msk}, \text{pp})$ ,  $\text{psk} \leftarrow (\text{msk}, \text{pp})$  and  $\text{csk} \leftarrow \text{pp}$ .
- **encodeEnroll** $^{\mathcal{O}_B}()$ : For a template vector  $\mathbf{b} = (b_1, b_2, \dots, b_k)$  sampled from  $\mathcal{O}_B$ , the function encodes it as  $\mathbf{x} = (x_1, x_2, \dots, x_{k+2}) = (b_1, b_2, \dots, b_k, 1, \|\mathbf{b}\|^2)$ .
- **Enroll**( $\text{esk}, \mathbf{x}$ ): It calls  $\text{FE.KeyGen}(\text{msk}, \text{pp}, \mathbf{x}) \rightarrow f_{\mathbf{x}}$  and outputs  $\mathbf{c}_{\mathbf{x}} \leftarrow f_{\mathbf{x}}$ .
- **encodeProbe** $^{\mathcal{O}_B}()$ : For a template vector  $\mathbf{b}' = (b'_1, b'_2, \dots, b'_k)$  sampled from  $\mathcal{O}_B$ , the function encodes it as  $\mathbf{y} = (y_1, y_2, \dots, y_{k+2}) = (-2b'_1, -2b'_2, \dots, -2b'_k, \|\mathbf{b}'\|^2, 1)$ .
- **Probe**( $\text{psk}, \mathbf{y}$ ): It calls  $\text{FE.Enc}(\text{msk}, \text{pp}, \mathbf{y}) \rightarrow \mathbf{c}_{\mathbf{y}}$  and outputs  $\mathbf{c}_{\mathbf{y}}$ .
- **Compare**( $\text{csk}, \mathbf{c}_{\mathbf{x}}, \mathbf{c}_{\mathbf{y}}$ ): It calls  $\text{FE.Dec}(\text{pp}, \mathbf{c}_{\mathbf{x}}, \mathbf{c}_{\mathbf{y}}) \rightarrow s$  and outputs the value  $s$ .
- **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

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

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  $\mathbf{csk}$  is public, and the enrollment secret key  $\mathbf{esk}$  and probe secret key  $\mathbf{psk}$  are the same. Anyone with access to the enrollment message  $\mathbf{c}_x$  and either one of  $\mathbf{esk}$ ,  $\mathbf{psk}$ , or a probe oracle  $\text{Probe}(\mathbf{psk}, \cdot)$  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  $\mathbf{x}\mathbf{y}^{*T} < \tau$  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 **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  $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$  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.

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

$$s = \text{FE.Dec}(\text{dk}_I, \mathbf{c}_x, \mathbf{c}_y) = \mathbf{x}\mathbf{I}_{k+2}\mathbf{y}^T = \mathbf{x}\mathbf{y}^T = \|\mathbf{b} - \mathbf{b}'\|^2.$$

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  $\mathbf{csk}$  is now secret, and the enrollment secret key  $\mathbf{esk}$  and probe secret key  $\mathbf{psk}$  are distinct. Without  $\mathbf{csk}$ , 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  $\text{Probe}(\text{psk}, \cdot)$ . If **Server** is malicious, then it can use  $\text{csk}$  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  $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$  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.

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

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  $\text{esk}$ ,  $\text{psk}$ ,  $\text{csk}$ . In the scheme in this section, however, **Setup** is run only once for all the domains, and each domain shares the same  $\text{csk}$  and the same  $\text{esk}$ ,  $\text{psk}$  except different labels.

- $\text{encodeEnroll}^{\mathcal{O}_B}()$ ,  $\text{encodeProbe}^{\mathcal{O}_B}()$ : The same as the scheme in 2.4.
- **Enroll**( $\text{esk}, \mathbf{x}$ ): It calls  $\text{FE.Enc}(\ell, \text{ek}_1, \mathbf{x}) \rightarrow \mathbf{c}_x$  and outputs  $\mathbf{c}_x$ .
- **Probe**( $\text{psk}, \mathbf{y}$ ): It calls  $\text{FE.Enc}(\ell, \text{ek}_2, \mathbf{y}) \rightarrow \mathbf{c}_y$  and outputs  $\mathbf{c}_y$ .
- **Compare**( $\text{csk}, \mathbf{c}_x, \mathbf{c}_y$ ): It calls  $\text{FE.Dec}(\text{dk}_I, \mathbf{c}_x, \mathbf{c}_y) \rightarrow s$  and outputs the value  $s$ .
- **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

$$s = \text{FE.Dec}(\text{dk}_I, \mathbf{c}_x, \mathbf{c}_y) = \mathbf{x} \mathbf{I}_{k+2} \mathbf{y}^T = \|\mathbf{b} - \mathbf{b}'\|^2.$$

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  $\text{esk}$  and probe secret key  $\text{psk}$  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  $\text{esk}$  or  $\text{psk}$ ,  $\mathbf{c}_x$  still does not leak information about  $\mathbf{x}$ . This is different from the scheme in Section 2.4, where we only need security against an adversary who has a probe oracle  $\text{Probe}(\text{psk}, \cdot)$ .

If **Server** and **Domain** are both malicious, then the adversary can use  $\text{csk}$  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  $\mathcal{O}_{\text{samp}}(\cdot)$  to all adversaries in this section.

- $\mathcal{O}_{\text{samp}}(\cdot)$ : 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  $\mathbf{b} \leftarrow \mathcal{B}_i$ .

In Table 1, we list a summary of the adversary in each game.

Games	Goals	Knowledge and Oracles	
		Device-of-User	Device-of-Domain
UF-MSC (Section 3.1)	Forge a probe	$\mathbf{c}_x, \mathcal{O}_{\mathcal{B}}, \mathcal{O}_{\text{auth}}^q$	-
UF-MDV (Section 3.2)	Forge a probe	$\text{esk}, \text{psk}, \mathbf{c}_x, \mathcal{O}_{\text{auth}}^q$	$\text{psk}, \mathbf{c}_x, \mathcal{O}_{\text{auth}}^q$
UF-MDM (Section 3.3)	Forge a probe	-	$\mathbf{c}_x, \mathcal{O}_{\text{Probe}}, \mathcal{O}'_{\text{Probe}}, \mathcal{O}_{\text{auth}}^q$
UF-MST (Section 3.4)	Forge a probe	-	$\text{esk}, \mathbf{c}_x, \mathcal{O}_{\text{Probe}}, \mathcal{O}'_{\text{Enroll}}, \mathcal{O}_{\text{auth}}^q$
IND-MSV (Section 3.5)	Identify User	$\text{csk}, \mathbf{c}_x, \{\mathbf{c}_y\}_{i=1}^t, \mathcal{O}_{\mathcal{B}^{(0)}}, \mathcal{O}_{\mathcal{B}^{(1)}}$	$\text{csk}, \mathbf{c}_x, \{\mathbf{c}_y\}_{i=1}^t, \mathcal{O}_{\mathcal{B}^{(0)}}, \mathcal{O}_{\mathcal{B}^{(1)}}, \mathcal{O}_{\text{Probe}}^{\text{samp}}$

Table 1: Summary of Adversaries in Security Games

#### 3.1 Unforgeability against Malicious Scanner (UF-MSC)

In the game of Unforgeability against Malicious Scanner, we model the ability of a malicious **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  $\mathcal{O}_{\text{auth}}^q$ , and it tries to find a valid probe message  $\tilde{\mathbf{z}}$ . The whole game UF-MSC is defined in Algorithm 1.

The given oracle is defined as follows:

- $\mathcal{O}_{\mathcal{B}}$ : It outputs a biometric sample  $\mathbf{b} \leftarrow \mathcal{B}$ .
- $\mathcal{O}_{\text{auth}}^q(\text{csk}, \mathbf{c}_x, \cdot)$ : 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  $\text{Verify}(\text{Compare}(\text{csk}, \mathbf{c}_x, \mathbf{z}))$ .

**Algorithm 1** UF-MSC<sub>II,ℬ</sub>( $\mathcal{A}$ )

---

```

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

```

---

**Algorithm 2** UF-MSC'<sub>II,ℬ</sub>( $\mathcal{A}'$ )

---

```

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

```

---

To consider potential false positives of biometrics match, we consider the plain UF-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 UF-MSC game of a scheme  $\Pi$  associated with a family of distributions  $\mathbb{B}$  as

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

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

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

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

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

Note that if  $\text{csk}$  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  $\text{Compare}(\text{csk}, \mathbf{c}_{\mathbf{x}}, \cdot)$  over  $q$  times. Also note that in the Device-of-Domain model, the probe oracle  $\text{Probe}(\text{psk}, \cdot)$  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  $\text{esk}, \text{psk}$  (only  $\text{psk}$  in Device-of-Domain model), enrollment message  $\mathbf{c}_{\mathbf{x}}$ , and oracle  $\mathcal{O}_{\text{auth}}^q$  and tries to find a valid probe message  $\tilde{\mathbf{z}}$ . The whole game UF-MDV is defined in Algorithm 3. Similarly to Section 3.1, we also consider the plain UF-MDV' game in Algorithm 4.

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

**Algorithm 3** UF-MDV<sub>II,ℬ</sub>( $\mathcal{A}$ )

---

```

1:  $\mathcal{B} \leftarrow \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$ 
2:  $\text{esk}, \text{psk}, \text{csk} \leftarrow \text{Setup}(1^\lambda)$ 
3:  $\mathbf{x} \leftarrow \text{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$ 
4:  $\mathbf{c}_{\mathbf{x}} \leftarrow \text{Enroll}(\text{esk}, \mathbf{x})$ 
5: In Device-of-User model:
6:    $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\text{auth}}^q}(\text{esk}, \text{psk}, \mathbf{c}_{\mathbf{x}})$ 
7: In Device-of-Domain model:
8:    $\tilde{\mathbf{z}} \leftarrow \mathcal{A}^{\mathcal{O}_{\text{auth}}^q}(\text{psk}, \mathbf{c}_{\mathbf{x}})$ 
9:  $s \leftarrow \text{Compare}(\text{csk}, \mathbf{c}_{\mathbf{x}}, \tilde{\mathbf{z}})$ 
10: return Verify( $s$ )

```

---

**Algorithm 4** UF-MDV'<sub>II,ℬ</sub>( $\mathcal{A}'$ )

---

```

1:  $\mathcal{B} \leftarrow \mathbb{B}, \mathbb{B} \leftarrow \mathbb{B} \setminus \mathcal{B}$ 
2:  $\text{esk}, \text{psk}, \text{csk} \leftarrow \text{Setup}(1^\lambda)$ 
3:  $\mathbf{x} \leftarrow \text{encodeEnroll}^{\mathcal{O}_{\mathcal{B}}}()$ 
4:  $\mathbf{c}_{\mathbf{x}} \leftarrow \text{Enroll}(\text{esk}, \mathbf{x})$ 
5: In Device-of-User model:
6:    $\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}_{\text{auth}}^q}()$ 
7: In Device-of-Domain model:
8:    $\tilde{\mathbf{z}} \leftarrow \mathcal{A}'^{\mathcal{O}_{\text{Probe}}, \mathcal{O}_{\text{auth}}^q}()$ 
9:  $s \leftarrow \text{Compare}(\text{csk}, \mathbf{c}_{\mathbf{x}}, \tilde{\mathbf{z}})$ 
10: return Verify( $s$ )

```

---

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

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

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

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

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

### 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  $\mathbf{c}_{\mathbf{x}}$  of the honest Domain and oracles  $\mathcal{O}_{\text{Probe}}$ ,  $\mathcal{O}'_{\text{Probe}}$ , and  $\mathcal{O}_{\text{auth}}^q$ , and it tries to find a valid probe message  $\tilde{\mathbf{z}}$ . The whole game UF-MDM is defined in Algorithm 5. Similarly to Section 3.1, we also consider the plain UF-MDM' game in Algorithm 6.

**Algorithm 5** UF-MDM<sub>II,ℬ</sub>( $\mathcal{A}$ )

---

```

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

```

---

**Algorithm 6** UF-MDM'<sub>II,ℬ</sub>( $\mathcal{A}'$ )

---

```

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

```

---

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

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

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

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

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

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

### 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  $\text{esk}$ , enrollment message  $\mathbf{c}_x$ , and oracles  $\mathcal{O}_{\text{Probe}}, \mathcal{O}'_{\text{Enroll}}$ , and  $\mathcal{O}_{\text{auth}}^q$ , 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 UF-MST $_{\Pi, \mathbb{B}}(\mathcal{A})$

---

```

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

```

---



---

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

---

```

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

```

---

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

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

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

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

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

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

### 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  $\mathbf{csk}$ , 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.

---

**Algorithm 9** IND-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:  $\mathbf{esk}, \mathbf{psk}, \mathbf{csk} \leftarrow \text{Setup}(1^\lambda)$ 
5:  $\mathbf{x} \leftarrow \text{encodeEnroll}^{\mathcal{O}_{\mathcal{B}^{(b)}}}()$ 
6:  $\mathbf{c}_x \leftarrow \text{Enroll}(\mathbf{esk}, \mathbf{x})$ 
7: for  $i = 1$  to  $t$  do
8:    $\mathbf{y}^{(i)} \leftarrow \text{encodeProbe}^{\mathcal{O}_{\mathcal{B}^{(b)}}}()$ 
9:    $\mathbf{c}_y^{(i)} \leftarrow \text{Probe}(\mathbf{psk}, \mathbf{y}^{(i)})$ 
10: end for
11: In Device-of-User Model:
12:    $\tilde{b} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{B}^{(0)}}, \mathcal{O}_{\mathcal{B}^{(1)}}}(\mathbf{csk}, \mathbf{c}_x, \{\mathbf{c}_y^{(i)}\}_{i=1}^t)$ 
13: In Device-of-Domain Model:
14:    $\tilde{b} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{B}^{(0)}}, \mathcal{O}_{\mathcal{B}^{(1)}}}(\mathbf{csk}, \mathbf{c}_x, \{\mathbf{c}_y^{(i)}\}_{i=1}^t)$ 
15: return  $1_{\tilde{b}=b}$ 

```

---

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

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

We provide  $\mathcal{O}_{\text{Probe}}^{\text{samp}}(\cdot)$  instead of  $\mathcal{O}_{\text{Probe}}(\mathbf{psk}, \cdot)$ . 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 IND-MSV game of a scheme  $\Pi$  associated with a family of distributions  $\mathbb{B}$  as

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

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

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

## 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](https://doi.org/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](https://doi.org/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>.