# Biometrics-Authenticated Key Exchange for Secure Messaging

Mei Wang[†]
Wuhan University
Wuhan, China
wangmeiz@whu.edu.cn

Kun He[*†]
Wuhan University
Wuhan, China
hekun@whu.edu.cn

Jing Chen[*†]
Wuhan University
Wuhan, China
chenjing@whu.edu.cn

Zengpeng Li[‡]
Shandong University
Qingdao, China
zengpengliz@gmail.com

Wei Zhao
Science and Technology on
Communication Security Laboratory
Chengdu, China
zhaowei9801@163.com

Ruiying Du[†]
Wuhan University
Wuhan, China
duraying@whu.edu.cn

## ABSTRACT

Secure messaging heavily relies on a session key negotiated by an Authenticated Key Exchange (AKE) protocol. However, existing AKE protocols only verify the existence of a random secret key (corresponding to a certificated public key) stored in the terminal, rather than a legal user who uses the messaging application. In this paper, we propose a Biometrics-Authenticated Key Exchange (BAKE) framework, in which a secret key is derived from a user's biometric characteristics that are not necessary to be stored. To protect the privacy of users' biometric characteristics and realize one-round key exchange, we present an Asymmetric Fuzzy Encapsulation Mechanism (AFEM) to encapsulate messages with a public key derived from a biometric secret key, such that only a similar secret key can decapsulate them. To manifest the practicality, we present two AFEM constructions for two types of biometric secret keys and instantiate them with irises and fingerprints, respectively. We perform security analysis of BAKE and show its performance through extensive experiments.

## CCS CONCEPTS

• **Security and privacy** → **Biometrics**; **Security protocols**.

## KEYWORDS

Authenticated key exchange; biometrics; secure messaging; fuzzy extractor; verifiable secret sharing

[*]Corresponding authors: hekun@whu.edu.cn and chenjing@whu.edu.cn
[†]Key Laboratory of Aerospace Information Security and Trust Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University
[‡]School of Cyber Science and Technology, Shandong Univeristy (Qingdao Campus) and Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University (Qingdao Campus)


*CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea.*

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484746

**ACM Reference Format:**
Mei Wang, Kun He, Jing Chen, Zengpeng Li, Wei Zhao, and Ruiying Du. 2021. Biometrics-Authenticated Key Exchange for Secure Messaging. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3460120.3484746

## 1 INTRODUCTION

Social messaging applications have become a mainstream means of daily communication due to their convenience. As of January 2021, three messaging applications, including WhatsApp, Facebook Messenger, and WeChat, have even more than 1.2 billion monthly active users [46]. Most messaging applications follow a store-and-forward paradigm, where a service provider passes messages between two communicating participants [47, 48]. To keep adversaries, including the service provider, from access to messages during storing and forwarding, the messages need to be encrypted so that only the communicating participants can read them. This feature is known as end-to-end encryption [14, 43], in which participants carry out *Authenticated Key Exchange* (AKE) to authenticate each other and negotiate on a session key [19, 34, 54, 56], and then use this key to secure messaging.

Although some messaging applications have deployed end-to-end encryption (*e.g.*, Signal [44], WhatsApp [51], Facebook Messenger [22], and Wire [27]), they all rely on traditional public-key technology. Roughly speaking, each participant generates a pair of public and secret keys and publishes the public key to the other participant. Then, the two communicating participants can execute a (possibly asynchronous) AKE protocol based on their keys. Unfortunately, those AKE protocols are not suitable for secure messaging in practice. First, those AKE protocols actually verify the possession of the secret key rather than the participant herself/himself. Since the secret key is usually stored in a terminal, an adversary can launch lunchtime attacks to impersonate that participant without any knowledge about the secret key [20]. Second, the secret key may be stolen if an adversary has access to the terminal [8]. More importantly, messaging applications cannot immediately determine whether a secret key is cloned [15]. Third, when a participant loses or replaces the terminal, it is difficult to update the public-secret key pair timely since every new public key needs to be authenticated through an out-of-band fashion before enabling it [12].

In this paper, we seek to design a *Biometrics-Authenticated Key Exchange* (BAKE) framework, in which a participant generates a secret key and a corresponding public key based on her/his biometric characteristics. A straightforward advantage of this framework is that the session key is negotiated for authenticated users, not authenticated random public keys. Since the secret key can be generated based on the biometric characteristics when needed, the secret key (and biometric characteristics) is never stored in the terminal, and there is no need to update the secret key and the corresponding public key when replacing terminals. The main disadvantage is that biometric characteristics are permanent, which means that the secret key cannot be updated after it is leaked. Fortunately, stealing biometric characteristics is not that simple, since many biometric characteristics (e.g., iris [28] and ear canal dimension [25]) require dedicated equipment to capture within a very short distance. Even for stealing fingerprints, an adversary needs to have access to what the victim has touched. Moreover, messaging applications can defeat biometric cloning and replay attacks with well-studied liveness detection techniques [2, 40, 53, 55, 57].

The major challenge of our design is to protect participants' biometric characteristics as required under regulations on data protection, such as General Data Protection Regulation (GDPR), while tolerating noises in biometric characteristics. We note that a highly related work, called Fuzzy Asymmetric Password-Authenticated Key Exchange (fuzzy aPAKE), was proposed by Erwig et al. [21]. However, their solutions have two limitations. First, they require the communicating participants to run interactive cryptographic primitives (*e.g.*, the oblivious transfer) many times, which introduces heavy communication overhead and is not suitable for asynchronous scenarios where the participants are not online at the same time. Second, they require that the biometric representation should be a rotation-invariant bit string, which means that a similar string can be extracted even if the captured biometric image is rotated. Some biometric representations (e.g., FingerCode [31], the most common fingerprint representation) do not satisfy the rotation-invariant property.

To tackle the asynchronous issue, we present a mechanism that encapsulates messages with the biometric public key of a participant, so that only the participant with similar biometric characteristics can obtain them, called *Asymmetric Fuzzy Encapsulation Mechanism* (AFEM). With AFEM, we propose a BAKE framework, whose authenticated key exchange phase is a one-round protocol. To solve the rotation-invariant issue, we propose two AFEM constructions for biometric vector and biometric vector set, respectively. The key insight is that many biometric characteristics are composed of discrete points, in which we can extract a rotation-invariant biometric vector set through the relative relationship of those points.

We conclude our main contributions as follows.

- We introduce a new notion of an asymmetric cryptographic protocol called biometrics-authenticated key exchange, in which secret keys are derived from biometric characteristics.
- Considering the asynchronous issue in secure messaging and different types of biometric secret keys, we propose an asymmetric fuzzy encapsulation mechanism along with two constructions for biometric vector and biometric vector set, respectively.

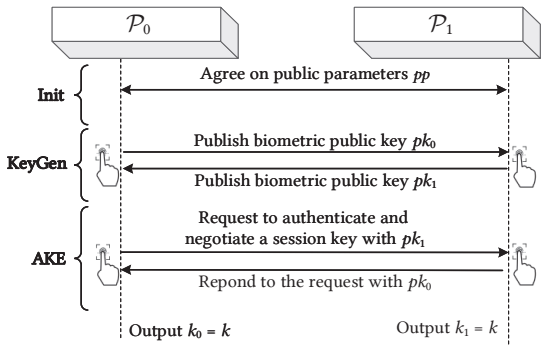



**Figure 1: System model of biometrics-authenticated key exchange for secure messaging.**

- We instantiate two biometrics-authenticated key exchange protocols for realistic biometric characteristics: irises and fingerprints. Specifically, we employ the most common IrisCode [16] for irises and design a rotation-invariant presentation for fingerprints.
- We conduct experiments on our two instantiations. The running time of our protocols is less than 0.2s on a realistic iris dataset, and is less than 0.5s on a realistic fingerprint dataset, which is at least 2000 times faster than fuzzy aPAKE [21]. The communication cost of our two instantiations is about 12.2KB and 2.7KB, which is at least 50 times lower than fuzzy aPAKE.

## 2 PROBLEM STATEMENT

This section briefly describes the system model, the threat model, and the design goals of biometrics-authenticated key exchange.

### 2.1 System Model

This work aims to provide a two-party bidirectional Biometrics-Authenticated Key Exchange (BAKE) for end-to-end secure messaging. As shown in Figure 1, there are two participants in our system: the sender $\mathcal{P}_0$ who initiates a session request and the receiver $\mathcal{P}_1$ who responds to this request. We sometimes use the terms "user" or "terminal" instead of "participant", where a user refers to a human who intends to participate in secure messaging and a terminal refers to a device that is possessed by a user, *e.g.*, a smartphone. A BAKE protocol is an asymmetric cryptographic primitive, in which a secret key is derived from the biometric characteristics of a user, such as an iris or a fingerprint, and a public key is derived from the corresponding secret key.

Specifically, a BAKE protocol consists of three phases. In the *initialization* (Init) phase, two participants agree on a set of public parameters $pp$ to initialize the whole system. In the *key generation* (KeyGen) phase, each participant $\mathcal{P}_i$ ($i \in \{0, 1\}$) generates a public key $pk_i$ based on her/his biometric characteristics and sends the public key to the other participant $\mathcal{P}_{1-i}$. In the *authenticated key exchange* (AKE) phase, the sender $\mathcal{P}_0$ makes a request to the receiver $\mathcal{P}_1$ to authenticate each other and negotiate on a session key $k$ that can be used to establish a secure channel. The participant needs her/his biometric characteristics and the public key of the other participant as input in this phase. Note that asynchronous scenarios require the AKE phase should be a one-round protocol.

## 2.2 Threat Model

As in other AKE protocols [19, 21], we only consider active adversaries in the network. Note that an honest-but-curious participant may try to learn the other participant's biometric characteristics from the BAKE protocol. However, this kind of attacks can be implemented by an adversary that only eavesdrops on the network. More specifically, we require that the participant's biometric characteristics and the session key are protected under the following threat model.

- **Insecure Channel.** We assume that adversaries have complete control of the channel between two participants in the AKE phase. That means an adversary can eavesdrop on, tamper with, and throw away any message in that phase.
- **Safeguarded Terminal.** We assume that a terminal processes a user's biometric characteristics honestly. Specifically, the terminal does not store the captured biometric characteristics or reveal the biometric characteristics to adversaries.

## 2.3 Design Goals

Our BAKE protocol should have the following properties.

- **Mutual Authentication.** Both participants authenticate each other using biometrics before actual messaging.
- **Secure Key Establishment.** A consistent session key is agreed upon between the sender and receiver and is only accessible to these two participants.
- **Biometric Privacy.** An adversary, including the communicating participant, cannot obtain the biometric characteristics of a participant from the protocol.
- **High Performance.** Both the computation and communication overhead of BAKE protocols should be low, which is critical for constrained environments, *e.g.*, mobile networks.

Note that fuzzy aPAKE [21] also achieves the first three goals and additionally provides the Universally Composable (UC) security, but fuzzy aPAKE dissatisfies the last goal since it involves heavy communication and computation overhead. In addition, fuzzy aPAKE only supports fuzzy vectors (*e.g.*, password) and is not suitable for asynchronous scenarios.

# 3 ASYMMETRIC FUZZY ENCAPSULATION MECHANISM

The core idea of our BAKE constructions is to derive a session key from random strings that are only accessible to the participant with correct biometric characteristics. To this end, we propose a cryptographic primitive called *Asymmetric Fuzzy Encapsulation Mechanism* (AFEM), which encapsulates a message with a public key that is corresponding to a target secret key. Only the participant who possesses a secret key close to the target secret key can obtain the random string from the encapsulated message.

We first introduce the syntax of AFEM and define the security notion for AFEM. Then, we present two constructions for two types of biometric secret keys.

## 3.1 Syntax

Let $\mathcal{SK}$ be the set of all possible secret keys, $\mathcal{S}$ be the set of all possible plain messages, and $dis(\cdot,\cdot)$ be a function that calculates the closeness of two inputs. We assume that the set of all possible public parameters, the set of all possible public keys, and the set of all possible encapsulated messages are implicitly defined in the algorithms.

*Definition 3.1 (AFEM).* An asymmetric fuzzy encapsulation mechanism AFEM is a tuple of four Probabilistic Polynomial Time (PPT) algorithms (Setup, PubGen, Enc, Dec) that satisfies the following syntax with the correctness property.

- $\mathsf{Setup}(1^\lambda, \tau) \to par$: This setup algorithm takes as input a security parameter $\lambda \in \mathbb{N}$ and a threshold $\tau \in \mathbb{N}$. It generates a set of public parameters $par$, which is an implicit input to the following algorithms.
- $\mathsf{PubGen}(sk) \to pk$: This public key generation algorithm takes as input a secret key $sk \in \mathcal{SK}$. It outputs a public key $pk$. (Notably, this kind of secret key is derived from biometric characteristics.)
- $\mathsf{Enc}(pk, s) \to c$: This encapsulation algorithm takes as input a public key $pk$ and a plain message $s \in \mathcal{S}$. It outputs an encapsulated message $c$.
- $\mathsf{Dec}(sk', c) \to s/\perp$: This deterministic decapsulation algorithm takes as input a secret key $sk' \in \mathcal{SK}$ and an encapsulated message $c$. It returns a plain message $s$ if $dis(sk, sk') < \tau$ or a failure symbol $\perp$ otherwise.

*Correctness.* For any $\lambda \in \mathbb{N}$, any $\tau \in \mathbb{N}$, any $par$ generated by Setup, any secret key $sk, sk' \in \mathcal{SK}$, and any plain message $s \in \mathcal{S}$, $\mathsf{Dec}(sk', \mathsf{Enc}(\mathsf{PubGen}(sk), s)) = s$ if $dis(sk, sk') < \tau$.

*Security.* The semantic security is defined by an attack game between a challenger $C$ and an adversary $\mathcal{A}$. Particularly, for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment $\mathsf{Exp}_{\mathcal{A}}(1^\lambda)$ is negligible.

(1) On input a security parameter $1^\lambda$, $\mathcal{A}$ outputs an appropriate threshold $\tau$ such that it satisfies the privacy and robustness properties for the secret key, and sends $\tau$ to $C$.
(2) $C$ executes $\mathsf{Setup}(1^\lambda, \tau) \to par$, $\mathsf{PubGen}(par, sk) \to pk$, and sends $pk$ to $\mathcal{A}$.
(3) $\mathcal{A}$ is given input $1^\lambda$, $par$, and oracle access to $\mathsf{Enc}(\cdot)$.
(4) $C$ generates a new secret key $sk'$ such that $dis(sk, sk') < \tau$. Further, $C$ sends the encapsulated message $c \leftarrow \mathsf{AFEM.Enc}(pk, b)$ for $b \leftarrow \{0,1\}$ to $\mathcal{A}$.
(5) $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The advantage of the adversary is denoted as $|\Pr[b = b'] - 1/2|$.

## 3.2 Construction for Biometric Vector

We propose the first AFEM construction for secret keys in the form of biometric vectors, which means that the biometric characteristics of a participant can be converted into a string. Assuming the secret key is $sk = \mathbf{u} \in \mathbb{F}_q^m$, where $q$ is a prime and $\mathbb{F}_q$ is a finite field, and the closeness is defined by Hamming distance, we give the technical description and construction as follows.

*3.2.1 Generating Public Key.* The key idea is to exploit the Learning With Errors (LWE) problem to securely encode a traditional secret key into a vector with the help of a biometric vector. Specifically, a random vector $\mathbf{x} \in \mathbb{F}_q^l$ is encoded by $\mathbf{e} = \mathbf{A}\mathbf{x} + \mathbf{u}$, where $\mathbf{A} \in \mathbb{F}_q^{m \times l}$ is a random matrix and $\mathbf{u}$ is the biometric vector. Then, $\mathbf{x}$ is mapped to $\mathbb{Z}_q$ through a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$, which allows us

**Algorithm 1:** Decoding algorithm $\text{Decode}_\tau$

Input $\mathbf{A}$, $\mathbf{b}$
Randomly select rows without replacement $j_1, \ldots, j_{2l} \leftarrow [1, m]$
Restrict $\mathbf{A}$, $\mathbf{b}$ to rows $j_1, \ldots, j_{2l}$ and denote as $\mathbf{A}^{j_1,\ldots,j_{2l}}$, $\mathbf{b}^{j_1,\ldots,j_{2l}}$
**if** *there exist l linearly independent rows of* $\mathbf{A}^{j_1,\ldots,j_{2l}}$ **then**
  Let $\mathbf{A}' = \mathbf{A}^{j_1,\ldots,j_{2l}}$, $\mathbf{b}' = \mathbf{b}^{j_1,\ldots,j_{2l}}$
  Compute $\mathbf{x}' = \mathbf{A}'^{-1}\mathbf{b}'$
**else**
  Abort
**if** $\mathbf{b} - \mathbf{A}\mathbf{x}'$ *has more than* $\tau$ *nonzero coordinates* **then**
  Go to step 2
Output $\mathbf{x}'$

to adopt an ElGamal-like encryption. Therefore, the public key is $(\mathbf{A}, \mathbf{e}, y = g^{H(\mathbf{x})})$, where $g$ is a generator of a group.

*3.2.2 Encapsulating Message.* To encapsulate a message $s$, we encrypt it with an ElGamal-like encryption. Specifically, for the public key $y$, a random value $r \in \mathbb{Z}_q$ is selected. Then, the encrypted message is $(g^r, y^r \oplus s)$.

*3.2.3 Decapsulating with Implicit Authentication.* To decapsulate the encapsulated message, we need to recover the traditional secret key. Due to the difficulty of the LWE problem, the key can be recovered only if the two biometric vectors are similar, which realizes implicit authentication. Specifically, we employ a decoding algorithm $\text{Decode}_\tau$ as shown in Algorithm 1, which can decode a random linear code with at most $\tau$ errors [24].

*3.2.4 Putting it All Together.* Let $sk, sk' \in \mathbb{F}_q^m$ and $s \in \mathbb{G}$. The AFEM construction for the biometric vector is as follows.

- $\text{Setup}(1^\lambda, \tau)$. Output $par = (\lambda, \mathbb{G}, q, g, \tau, m, l, H)$ where $\mathbb{G}$ is a cyclic group of prime order $q$, $g$ is a generator of $\mathbb{G}$, $m \geq 3l \in \mathbb{N}$, and $H : \{0,1\}^* \to \mathbb{Z}_q$ is a hash function.
- $\text{PubGen}(sk)$. Parse $sk$ as $\mathbf{u}$. Choose a random matrix $\mathbf{A} \in \mathbb{F}_q^{m \times l}$ and a random vector $\mathbf{x} \in \mathbb{F}_q^l$. Compute $\mathbf{e} = \mathbf{A}\mathbf{x} + \mathbf{u}$ and $y = g^{H(\mathbf{x})}$ and output $pk = (\mathbf{A}, \mathbf{e}, y)$.
- $\text{Enc}(pk, s)$. Parse $pk$ as $(\mathbf{A}, \mathbf{e}, y)$. Choose a random $r \in \mathbb{Z}_q$. Compute $c_0 = g^r$ and $c_1 = y^r \oplus s$ and output $c = (\mathbf{A}, \mathbf{e}, c_0, c_1)$.
- $\text{Dec}(sk', c)$. Parse $sk'$ as $\mathbf{u}'$ and $c$ as $(\mathbf{A}, \mathbf{e}, c_0, c_1)$. Compute $\mathbf{x}' = \text{Decode}_\tau(\mathbf{A}, \mathbf{e} - \mathbf{u}')$. Output $\perp$ if the decoding algorithm is aborted and otherwise $s = c_1 \oplus c_0^{H(\mathbf{x}')}$.

Since the correctness is clear, below, we sketch the security.

Theorem 3.2. *The* AFEM *scheme is semantic secure guaranteed by the decisional-LWE assumption and DDH assumption under the random oracle model.*

Our goal is to show that any PPT adversary $\mathcal{A}$ can only break the semantic security with negligible advantage. Next, we show the security via the following hybrids.

Hy.0. This is the real game with any PPT adversary $\mathcal{A}$ and a challenger $C$. The game outputs 1 if $b' = b$ and 0 otherwise. We define the advantage $\text{Adv}_{\mathcal{A}}^{\text{Hy}_0}(\lambda) = Pr[b = b']$.

Hy.1. Hy.1 is identical to Hy.0 except that we use $\mathbf{e}'$ from the uniform distribution to replace the second element $\mathbf{e} = \mathbf{A}\mathbf{x} + \mathbf{u}$ in $pk$. Under the decisional-LWE assumption, the uniform $\mathbf{e}'$ is computationally indistinguishable from $\mathbf{e} = \mathbf{A}\mathbf{x} + \mathbf{u}$ with computational distance $negl(\lambda)$. Thus, we have $\text{Adv}_{\mathcal{A}}^{\text{Hy}_0}(\lambda) \leq \text{Adv}_{\mathcal{A}}^{\text{Hy}_1}(\lambda) + negl(\lambda)$.

Hy.2. Hy.2 is identical to Hy.1 except that we use $h$ from the uniform to replace $H(\mathbf{x})$, and the random oracle guarantees no PPT adversary can distinguish $H(\mathbf{x})$ from the uniform. Further, the discrete-logarithm assumption guarantees no one has the ability to obtain $\mathbf{x}$ given $y$ and $g$, which implies that no PPT adversary can tell the diffidence between $y = g^{H(\mathbf{x})}$ and $y' = g^h$. Thus, we have $\text{Adv}_{\mathcal{A}}^{\text{Hy}_1}(\lambda) \leq \text{Adv}_{\mathcal{A}}^{\text{Hy}_2}(\lambda) + negl(\lambda)$.

Hy.3. Hy.3 is identical to Hy.2 except that we use $z$ from the uniform to calculate $c_1' = s \oplus g^z$. Under the decisional Diffie-Hellman (DDH) assumption, $c_1'$ is computationally indistinguishable from $c_1$ with computational distance $negl(\lambda)$, thus, we have $\text{Adv}_{\mathcal{A}}^{\text{Hy}_2}(\lambda) \leq \text{Adv}_{\mathcal{A}}^{\text{Hy}_3}(\lambda) + negl(\lambda)$. In Hy.3, all the elements of both the public key and the ciphertext are uniformly random and independent of the message. Hence, $\text{Adv}_{\mathcal{A}}^{\text{Hy}_3}(\lambda) = 1/2$.

Finally, we have $\text{Adv}_{\mathcal{A}}^{\text{Hy}_0}(\lambda) \leq 1/2 + negl(\lambda)$. Therefore, the adversary $\mathcal{A}$ can break the semantic security with negligible advantage. This completes the sketched proof.

## 3.3 Construction for Biometric Vector Set

We propose the second AFEM construction for secret keys in the form of biometric vector sets, which means that the biometric characteristics of a participant can be converted into a set of strings. Assuming the secret key is $sk = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ where $\mathbf{u}_j \in \{0,1\}^*$ for $j \in [1, n]$ and the closeness is defined by set difference, we give the technical description and construction as follows.

*3.3.1 Generating Public Key.* Different from biometric vector, biometric vector set usually consists of many relatively short vectors (but each vector still has enough entropy) that are not suitable for LWE-based constructions. Fortunately, we can obtain many of the same biometric vectors in two captures due to the relatively short length. Then, these biometric vectors can be treated as traditional secret keys as in Section 3.2. Specifically, for $\mathbf{u}_j \in sk$ ($j \in [1, n]$), the corresponding partial public key is $y_j = g^{H(\mathbf{u}_j)}$, where $g$ is a generator of a group and $H : \{0,1\}^* \to \mathbb{Z}_q$ is a hash function. Considering the relatively short length, we can further require $H$ to be a computationally expensive hash function.

*3.3.2 Encapsulating Message.* To encapsulate a message $s$, we also employ an ElGamal-like encryption as in Section 3.2. Instead of encrypting $s$ with each partial public key, we first divide $s$ into $n$ shares with a $(t, n)$ Verifiable Secret Sharing (VSS) scheme [23] and then encrypt each share with one partial public key. To further protect the encrypted shares, we transform them into a set of points and obtain a polynomial through interpolation. Specifically, we employ the VSS scheme proposed by Paul Feldman whose share generation algorithm is $\text{VSS.ShareGen}(s, t, n) \to (\{s_1, \ldots, s_n\}, \{com_0, \ldots, com_{t-1}\})$. A random value $r \in \mathbb{Z}_q$ is selected for encrypting each share as $(g^r, \beta_j = y_j^r \oplus s_j)$ ($j \in [1, n]$). Then, another random value $r' \in \mathbb{Z}_q$ is selected and each $\beta_j$ is transformed into a point $(\alpha_j = y_j^{r'}, \beta_j)$. Finally, we interpolate the unique polynomial $poly$ of degree $n-1$ over the points $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\}$ that consists of $n$ coefficients.

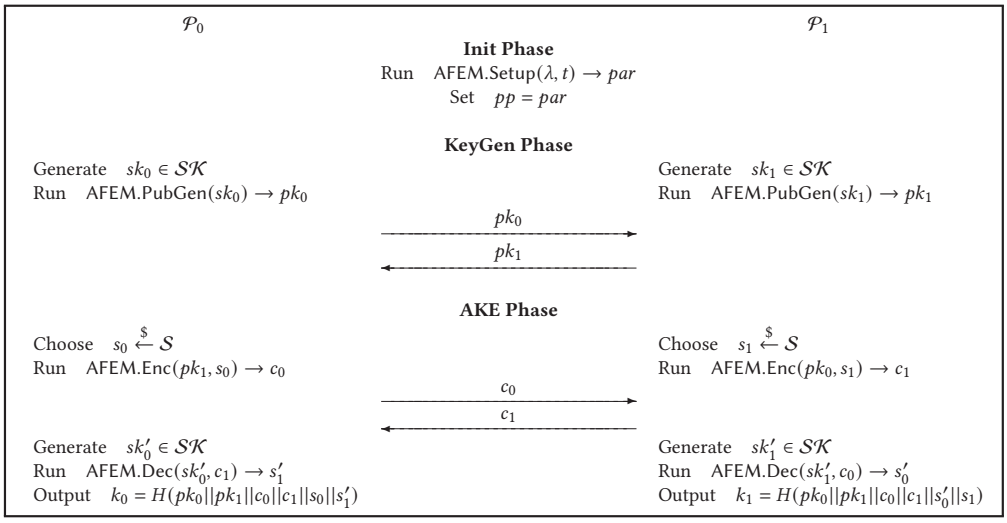



**Figure 2: Detail of our BAKE framework.**

*3.3.3 Decapsulating with Implicit Authentication.* Generally, a $(t, n)$ VSS scheme has the property that any $t$ or more than $t$ shares can recover $s$ while any less than $t$ shares reveal no information about $s$, and check whether the shares can be used to reconstruct the message. Therefore, the authentication is guaranteed implicitly. Specifically, $n$ encrypted shares are obtained from the polynomial $poly$ and are decrypted as in Section 3.2. Then, each share can be verified by running the verification algorithm VSS.Verify$(\{com_0, \ldots, com_{t-1}\}, s_j)$, where the output 1 means that $s_j$ is valid. If the number of valid shares is not less than the threshold $t$, all valid shares could be recovered as the original message $s$ by running the share reconstruction algorithm VSS.ShareRecon.

*3.3.4 Putting it All Together.* Let $sk = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$, $sk' = \{\mathbf{u}'_1, \ldots, \mathbf{u}'_n\}$, and $s \in \mathbb{G}$. The AFEM construction for the biometric vector set is as follows.

- Setup$(1^\lambda, \tau)$. Output $par = (\lambda, \mathbb{G}, q, g, \tau, n, H)$ where $\mathbb{G}$ is a cyclic group of prime order $q$, $g$ is a generator of $\mathbb{G}$, and $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a hash function.
- PubGen$(sk)$. Parse $sk$ as $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$. Compute $y_j = g^{H(\mathbf{u}_j)}$ for $j \in [1, n]$ and output $pk = \{y_1, \ldots, y_n\}$.
- Enc$(pk, s)$. Parse $pk$ as $\{y_1, \ldots, y_n\}$. Run VSS.ShareGen$(s, n - \tau, n) \rightarrow (\{s_1, \ldots, s_n\}, \{com_0, \ldots, com_{n-\tau-1}\})$. Interpolate a polynomial $poly$ on the point set $\{(y_1^{r'}, y_1^r \oplus s_1), \ldots, (y_n^{r'}, y_n^r \oplus s_n)\}$ where $r', r \xleftarrow{\$} \mathbb{Z}_q$, and output $c = (poly, g^{r'}, g^r, \{com_0, \ldots, com_{n-\tau-1}\})$.
- Dec$(sk', c)$. Parse $sk'$ as $\{\mathbf{u}'_1, \ldots, \mathbf{u}'_n\}$ and $c$ as $(poly, g^{r'}, g^r, \{com_0, \ldots, com_{n-\tau-1}\})$. Compute $s'_j = poly\left((g^{r'})^{H(\mathbf{u}'_j)}\right) \oplus (g^r)^{H(\mathbf{u}'_j)}$ and run VSS.Verify$(\{com_0, \ldots, com_{n-\tau-1}\}, s'_j)$ to check the validity for $j \in [1, n]$. Output $\perp$ if the number of valid shares is less than $n - \tau$ and otherwise the output of VSS.ShareRecon on the $n - \tau$ valid shares.

The correctness is clear and the semantic security is guaranteed by the DDH assumption under the random oracle model. We omit the details since the analysis is similar to that in Section 3.2.

# 4 BIOMETRICS-AUTHENTICATED KEY EXCHANGE FOR SECURE MESSAGING

We design a Biometrics-Authenticated Key Exchange (BAKE) framework based on AFEM and explain how this framework can be applied to secure messaging. We also instantiate BAKE with two common biometric characteristics, including irises and fingerprints, to intuitively show the practicability of our BAKE framework.

## 4.1 BAKE Framework

Our BAKE framework involves two participants communicating on an insecure channel and consists of three phases: the initialization (Init) phase, the key generation (KeyGen) phase, and the authenticated key exchange (AKE) phase. The core idea of our framework is to derive a session key for secure messaging from random strings generated by two participants. To securely transmit a random string generated by one participant to the other one, we employ an AFEM scheme to encapsulate the string. Authentication is implicitly executed when a participant tries to decapsulate the received encapsulated string. The detail of our BAKE framework is shown in Figure 2.

**Init Phase.** This phase provides all public parameters required in other phases. Specifically, $\mathcal{P}_0$ and $\mathcal{P}_1$ have to agree on essential parameters, that is, the security parameter $\lambda$ and the threshold $\tau$. Then, the setup algorithm AFEM.Setup is invoked to produce the public parameters $par$ of AFEM. Finally, the public parameters of BAKE are set to $par$ and are accessible to both participants.

In real-world applications (*e.g.*, secure messaging), a service provider could produce the public parameters $pp$ and publish them on a bulletin board or encode them into software, so that every participant can access them.

**KeyGen Phase.** In this phase, each participant produces a public key based on her/his biometric characteristics and sends it to the other one. Specifically, $\mathcal{P}_i$ ($i \in \{0,1\}$) generates a secret key $sk_i$ based on the biometric characteristics, which is instantiated in the following two subsections. Then, $\mathcal{P}_i$ obtains her/his public key $pk_i$

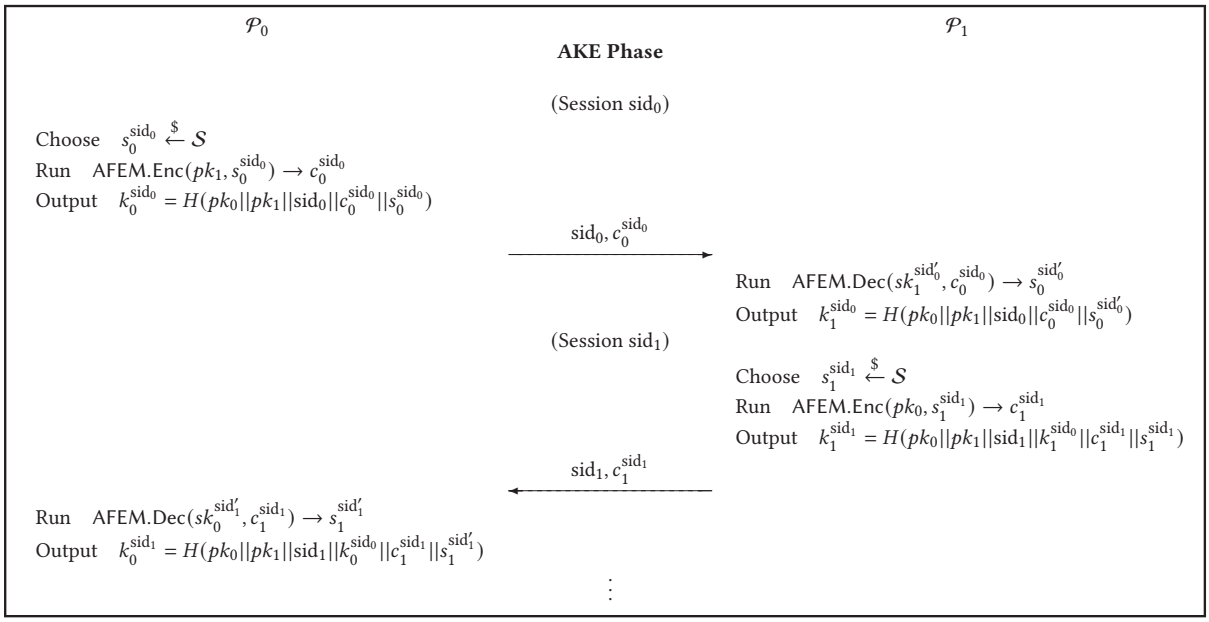



**Figure 3: Asynchronous mode of our BAKE framework.**

by running the public key generation algorithm AFEM.PubGen. Finally, $\mathcal{P}_i$ sends the public key $pk_i$ to $\mathcal{P}_{1-i}$ through an authenticated channel, which means that adversaries cannot modify that public key. Note that the requirement of an authenticated channel is essential for all authenticated key exchange protocols [6].

In real-world applications, the authenticated channel can be implemented by Public-Key Infrastructure (PKI) technology [52], in which an authority generates a certificate to bind an identity and a public key. Messaging applications also suggest an out-of-band fashion to authenticate public keys, such as comparing public key fingerprints and scanning a Quick Response (QR) code [44].

**AKE Phase.** This phase enables both participants to authenticate each other and negotiate a session key. Specifically, $\mathcal{P}_i$ ($i \in \{0, 1\}$) first chooses a random message $s_i$. Then, $s_i$ is encapsulated into $c_i$ by running the encapsulation algorithm AFEM.Enc and sent to $\mathcal{P}_{1-i}$. After receiving $c_{1-i}$, $\mathcal{P}_i$ generates a secret key $sk_i'$ based on the biometric characteristics and decapsulates $c_{1-i}$ using the decapsulation algorithm AFEM.Dec to obtain $s_{1-i}'$. Finally, $\mathcal{P}_i$ computes a session key $k_i$ by a hash function $H$.

In real-world applications, communicating participants are usually not online at the same time and a participant may want to leave a message to an offline participant through a service provider. To deal with this asynchronous scenario, we design another AKE phase that allows a unidirectional session key at the beginning as shown in Figure 3. Roughly speaking, in each session, a session key is derived from the session ID, a random string chosen by a participant, and the session key of the last session if it exists. To obtain the random string encapsulated by the other participant, a participant needs to generate a fresh secret key based on her/his biometric characteristics in each session for implicit authentication.

More specifically, for the first session whose ID is $\text{sid}_0$, the sender $\mathcal{P}_0$ first chooses a random message $s_0^{\text{sid}_0}$ to generate the session key $k_0^{\text{sid}_0}$, and then encapsulates $s_0^{\text{sid}_0}$ into $c_0^{\text{sid}_0}$. When the receiver $\mathcal{P}_1$ gets online, she/he generates a secret key based on the biometric characteristics to decapsulate $c_0^{\text{sid}_0}$ and obtain the session key $k_1^{\text{sid}_0}$. Then, the sender $\mathcal{P}_1$ launches the second session $\text{sid}_1$ to send a message to the receiver $\mathcal{P}_0$, where the new session key $k_1^{\text{sid}_1}$ is generated by decapsulating $c_1^{\text{sid}_1}$. The subsequent session keys could be negotiated in the same manner, which provides the key rotation property for our BAKE framework.

## 4.2 Secret Key from Iris

IrisCode [16] is the most widely used iris recognition method due to its computational advantages, *e.g.*, high matching speed and accuracy. Over 60 million people are using IrisCode to perform iris recognition and many other biometric algorithms are extended from IrisCode [16]. Generally, iris recognition is simply achieved by computing the Hamming distance between two IrisCodes. In BAKE, IrisCode is suitable for instantiation by our first AFEM construction that is illustrated in Section 3.2.

*4.2.1 Iris Vector from IrisCode.* An iris image is transformed into random texture after localization, segmentation, and normalization, which is then encoded into a 2048-bit stream [17]. To construct a valid secret key for the AFEM construction in Section 3.2, a 2048-bit IrisCode should be transformed into a vector. We naturally employ a simple solution to shard a 2048-bit IrisCode into $m$ elements that compose an iris vector. As shown in Figure 4, we take the first $2048/m$ bits as the first element, and take the second $2048/m$ bits as the second element, and so on. Finally, we obtain an iris vector $\mathbf{v} = \{v_1, \ldots, v_m\}$.

*4.2.2 Tolerating Iris Noise with Lattice.* Due to capture deviation, the decoding algorithm in Section 3.2 may fail even if the two iris

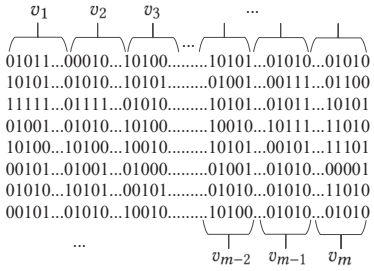



**Figure 4: An illustration of constructing an iris vector.**

vectors are from the same user. To tackle this issue, we expect to find an algorithm $f(\cdot)$ to obtain the same output vectors $\mathbf{u} = \mathbf{u}'$ from two slightly different iris vectors $\mathbf{v}$ and $\mathbf{v}'$, where $\mathbf{u} = f(\mathbf{v})$, $\mathbf{u}' = f(\mathbf{v}')$. We enforce this goal by solving the $\gamma$-CVP problem [1] on a well-chosen lattice $\mathcal{L}$ with Babai's algorithm [3]. Roughly speaking, Babai's algorithm maps all vectors close to a certain point in a lattice to that point, where a lattice is a discrete set of points in a vector space. Therefore, we employ this algorithm to eliminate noises in our design.

*Definition 4.1 (Lattice [29]).* Let $\mathbf{u}_1, \ldots, \mathbf{u}_m \in \mathbb{R}^\delta$ be a set of linearly independent vectors. The lattice $\mathcal{L}$ generated by $\mathbf{u}_1, \ldots, \mathbf{u}_m$ is the set of linear combinations of $\mathbf{u}_1, \ldots, \mathbf{u}_m$,

$$\mathcal{L} = \{a_1\mathbf{u}_1 + a_2\mathbf{u}_2 + \cdots + a_m\mathbf{u}_m : a_1, a_2, \ldots, a_m \in \mathbb{Z}\}.$$

Any set of independent vectors that generates $\mathcal{L}$ is a basis. The number $m$ of vectors in a basis is known as the dimension for $\mathcal{L}$.

*Definition 4.2 ($\gamma$-Closest Vector Problem [1]).* For any approximation factor $\gamma(m) \geq 1$, given a target vector $\mathbf{v} \in \mathbb{R}^\delta$ and a basis for a lattice $\mathcal{L} \subset \mathbb{R}^\delta$, find $\mathbf{u} \in \mathcal{L}$ satisfying

$$\|\mathbf{u} - \mathbf{v}\| \leq \gamma(m) \cdot dis(\mathbf{v}, \mathcal{L}),$$

where $dis(\mathbf{v}, \mathcal{L}) = \min_{\mathbf{u} \in \mathcal{L}} \|\mathbf{u} - \mathbf{v}\|$.

For efficiency, we first choose an orthonormal basis for $\mathcal{L}$. Then for any two slightly different target vectors $\mathbf{v}$ and $\mathbf{v}'$, we invoke the Babai's algorithm to achieve

$$\text{Babai}(\mathcal{L}, \mathbf{v}) = \mathbf{u} = \mathbf{u}' = \text{Babai}(\mathcal{L}, \mathbf{v}').$$

## 4.3 Secret Key from Fingerprint

Most researches take FingerCode to present the fingerprint, which is a 640-dimensional vector of integers [11, 58]. However, FingerCode is rotation-variant such that rotating a fingerprint image usually causes distinct FingerCodes as described in [58], which is not suitable for our first AFEM construction in Section 3.2. In addition, a 640-dimensional vector is not lightweight enough for real-world applications.

Another fingerprint recognition method is based on minutiae-based fingerprint presentation, which consists of a set of minutiae points [35]. Specifically, a human fingerprint is a unique pattern of ridges and valleys on the surface of an individual finger. Minutiae points are defined as the positions of local discontinuities where the ridge splits or ends, and are typically represented as: 1) an

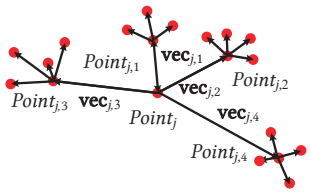



**Figure 5: An illustration of constructing a fingerprint vector.**

$X$-coordinate, 2) a $Y$-coordinate, 3) an orientation corresponding to the angle between the minutiae ridge and the horizontal line measured in degrees. To extract high-accuracy minutiae points with varied-quality fingerprint images, the segmentation algorithm first separates the foreground from the noisy background. Then, the original ridge flow pattern is kept with an image enhancement algorithm without introducing false information. Finally, minutiae points are located accurately with binarized minutiae extraction.

Since minutiae-based fingerprint presentation produces a set of minutiae points, it is suitable for instantiation by our second AFEM construction in Section 3.3. Moreover, we note that a good quality human fingerprint generally contains about $40 \sim 100$ minutiae points and a partial human fingerprint contains much fewer minutiae points ($20 \sim 30$ approximately). The performance of our BAKE protocol is significantly enhanced with the smaller fingerprint vector set $n$ compared to FingerCode.

*4.3.1 Fingerprint Vector Set from Minutiae Points.* To facilitate the practicability of our BAKE protocol, we propose a minutiae-based algorithm to produce a fingerprint vector set of size $n$, where $n$ is the number of minutiae points that a human fingerprint contains. Specifically, for $n$ minutiae points $\{Point_1, \ldots, Point_n\}$ in the $X$-$Y$ coordinate space, a fingerprint vector set is constructed as depicted in Algorithm 2. In this algorithm, every minutiae point is initialized as the central point for exactly one time. Then, the straight-line nearest $\mu$ points are chosen to form a structure as shown in Figure 5.

We take $\mu = 4$ as an example. Let $Point_j$ ($j \in [1, n]$) be the core point and $Point_{j,\rho}$ ($\rho \in [1, 4]$) be the top $\mu$ straight-line nearest points to $Point_j$. We define $\mathbf{vec}_{j,\rho}$ ($\rho \in [1, 4]$) as the vector from $Point_j$ to $Point_{j,\rho}$. Let $d_{j,\rho}$ denote the length of the vector $\mathbf{vec}_{j,\rho}$ and $\phi_{j,\omega}$ ($\omega \in [1, 6]$) denote the angles. Then, we can represent $\mathbf{v}_{j,0} = (d_{j,1}, d_{j,2}, d_{j,3}, d_{j,4}, \phi_{j,1}, \phi_{j,2}, \phi_{j,3}, \phi_{j,4}, \phi_{j,5}, \phi_{j,6})$. Next, for each point $Point_{j,\rho}$ ($\rho \in [1, 4]$), find the nearest $\mu$ points $Point_{j,\rho,\sigma}$ ($\sigma \in [1, 4]$). We define $\mathbf{vec}_{j,\rho,\sigma}$ ($\sigma \in [1, 4]$) as the vector from $Point_{j,\rho}$ to $Point_{j,\rho,\sigma}$. Let $d_{j,\rho,\sigma}$ denote the length of the vector $\mathbf{vec}_{j,\rho,\sigma}$ and $\phi_{j,\rho,\omega}$ ($\omega \in [1, 6]$) denote the angles. Similarly, we can represent the vector $\mathbf{v}_{j,\rho} = (d_{j,\rho,1}, d_{j,\rho,2}, d_{j,\rho,3}, d_{j,\rho,4}, \phi_{j,\rho,1}, \phi_{j,\rho,2}, \phi_{j,\rho,3}, \phi_{j,\rho,4}, \phi_{j,\rho,5}, \phi_{j,\rho,6})$ ($\rho, \sigma \in [1, 4]$). Finally, a fingerprint vector is represented as $\mathbf{v}_j = (\mathbf{v}_{j,0}, \mathbf{v}_{j,1}, \mathbf{v}_{j,2}, \mathbf{v}_{j,3}, \mathbf{v}_{j,4})$ that is concatenated with 50 values.

Since the above fingerprint processing method is based on the relative position of minutiae points, rotating images does not affect the fingerprint representation.

*4.3.2 Tolerating Fingerprint Noise with Lattice.* Similar to the iris vector, a fingerprint vector also should be processed to mitigate noise. To this end, we also employ a well-chosen lattice. We omit the repeatability description and refer to Section 4.2.2 for details.

**Algorithm 2:** Minutiae-based Fingerprint Vector Set Constructing Algorithm

**for** $Point_j \in \{Point_1, \ldots, Point_n\}$ **do**
  Set $Point_j$ as the center and find the nearest $\mu$ points
  Construct $\mu$ vectors $\{\mathbf{vec}_{j,\rho}\}_\mu$
  Compute vector lengths $\{d_{j,\rho}\}_\mu$ for $\{\mathbf{vec}_{j,\rho}\}_\mu$
  Compute inter-vector angles $\{\phi_{j,\omega}\}_{\frac{\mu(\mu-1)}{2}}$ in $\{\mathbf{vec}_{j,\rho}\}_\mu$
  Represent vector $\mathbf{v}_{j,0}$ with $\{d_{j,\rho}\}_\mu$ and $\{\phi_{j,\omega}\}_{\frac{\mu\cdot(\mu-1)}{2}}$
  **for** $\rho \in [1, \mu]$ **do**
    Set $Point_{j,\rho}$ as the core and find the nearest $\mu$ points
    Construct $\mu$ vectors $\{\mathbf{vec}_{j,\rho,\sigma}\}_\mu$
    Compute vector lengths $\{d_{j,\rho,\sigma}\}_\mu$ for $\{\mathbf{vec}_{j,\rho,\sigma}\}_\mu$
    Compute inter-vector angles $\{\phi_{j,\rho,\omega}\}_{\frac{\mu\cdot(\mu-1)}{2}}$ in $\{\mathbf{vec}_{j,\rho,\sigma}\}_\mu$
    Represent vector $\mathbf{v}_{j,\rho}$ with $\{d_{j,\rho,\sigma}\}_\mu$ and $\{\phi_{j,\rho,\omega}\}_{\frac{\mu\cdot(\mu-1)}{2}}$
  Represent $\{\mathbf{vec}_{j,\rho}\}_\mu$ as vector $\mathbf{v}_j = (\mathbf{v}_{j,0}, \mathbf{v}_{j,1}, \mathbf{v}_{j,2}, \mathbf{v}_{j,3}, \mathbf{v}_{j,4})$

## 5 SECURITY

We detail security analysis with Find-then-Guess (FtG) paradigm in the Bellare-Pointcheval-Rogaway (BPR) model [7], where the adversary $\mathcal{A}$ is given access to oracles through the following oracle queries. Particularly, users who hold biometrics (from a specific biometric dictionary) are modeled as PPT algorithms that respond to queries. $\mathcal{P}_0^i$ (vs. $\mathcal{P}_1^j$) denotes instance $i$ (vs. $j$) of user $\mathcal{P}_0$ (vs. $\mathcal{P}_1$) who executes the protocol multiple times with different partners (we use vs. to simply describe the similar case of the other user).

*Execute.* This oracle models a passive $\mathcal{A}$ (*e.g.*, an eavesdropper) who receives all the transcripts of an honest execution between an instance of a sender $\mathcal{P}_0^i$ and an instance of a receiver $\mathcal{P}_1^j$.

*Send.* This oracle models an active adversary $\mathcal{A}$ who can intercept a message, modify it, create a new one, or simply forward it to the intended party. For example, $\mathcal{A}$ could launch impersonation attacks via *Send* queries. Particularly, we separate three kinds of *Send* oracles in BAKE, $Send_0(\mathcal{P}_0^i, \mathcal{P}_1^j)$ implies that $\mathcal{P}_0$ initiates an execution with $\mathcal{P}_1$, $Send_1(\mathcal{P}_0^i, m)$ and $Send_2(\mathcal{P}_1^j, m)$ implies a message $m$ is sent by $\mathcal{A}$ to the instance $\mathcal{P}_0^i$ and $\mathcal{P}_1^j$, and outputs the first and second message that the instance of the user who generates upon receipt of the messages.

*Reveal.* This oracle models the misuse of session keys by a user, and $\mathcal{A}$ gets the session key held by the user.

*Test.* This oracle models $\mathcal{A}$ is given either a session key or a random value (depending on a choice bit $b$) and must distinguish them. If no session key for instance $\mathcal{P}_0^i$ (vs. $\mathcal{P}_1^j$) is defined , then return the undefined symbol $\perp$. Otherwise, return a guess bit $b' \in \{0, 1\}$ for the choice bit $b$.

*Corrupt.* This oracle models the client outputs the biometric secret key, which does not reveal the internal state, but reveals the secret key and can be made at any point during the protocol.

**Advantage of the adversary**. *Freshness* oracle is defined to restrict the queries to a target session. The security experiment is performed as a game between a challenger $C$ and an adversary $\mathcal{A}$ based on BAKE with a security parameter $\lambda$. Concretely, $C$ first initializes the whole system and realizes all oracles, then gives all public information to $\mathcal{A}$, $\mathcal{A}$ then interacts with $C$ via a series of queries on *Execute*, *Send*, *Reveal*, and *Corrupt*. In the meanwhile, $\mathcal{A}$ issues a *Test* query, and keeps querying other oracles afterward as before. Finally, $\mathcal{A}$ terminates this experiment and outputs a guess bit $b'$ for the choice bit $b$ in *Test*. For BAKE, an instance of the sender represents an *online* attack if both the following cases are true at the time of the *Test* query, (1) at some point, $\mathcal{A}$ queried $Send_1(\mathcal{P}_0^i, *)$ (vs. $Send_2(\mathcal{P}_1^j, *)$); (2) at some point, $\mathcal{A}$ queried $Test(\mathcal{P}_0^i)$ (vs. $Test(\mathcal{P}_1^j)$). The number of *online* attacks represents a bound on the number of biometric secret keys that $\mathcal{A}$ could have tested in an online fashion. A PPT $\mathcal{A}$ may succeed with probability 1 by trying all biometric secret keys if the size of the biometric dictionary is small. Therefore, $\mathcal{A}$ is only said to have succeeded if $\mathcal{A}$ asks a single *Test* query, outputs a guess bit $b'$ such that $b' = b$. The advantage of $\mathcal{A}$ in attacking BAKE is formally defined by

$$\mathrm{Adv}_{\mathcal{A}}(\lambda) = |2 \cdot \Pr[b = b'] - 1|. \quad (1)$$

*Definition 5.1.* BAKE is said to be secure if for every biometric dictionary $D$ with size $\|D\|$ and for all PPT $\mathcal{A}$ making at most $Q_s$ online dictionary attacks (*i.e.*, the number of *Send* queries), it holds that $\mathrm{Adv}_{\mathcal{A}}(\lambda) \le Q_s/\|D\| + negl(\lambda)$.

Remark 1. *Fuzzy aPAKE [21] is using an ideal cipher as a block cipher that takes as input a plaintext or a ciphertext. In our solution, we instantiate the block cipher using our proposed AFEM instead.*

Theorem 5.2. *BAKE is secure with the advantage* $\mathrm{Adv}_{\mathcal{A}}(\lambda) \le Q_s/\|D\| + negl(\lambda)$ *if AFEM is semantic secure and assuming all collision-resistant hash functions are random oracles* $\mathcal{H}$*, where* $Q_s$ *is the number of the online attacks (i.e., the number of Send queries) and the bit-length of the output of* $\mathcal{H}$ *is* $\ell$*.*

*Proof of Theorem 5.2.* We use $\mathcal{A}$ to construct a simulator $\mathcal{S}$ that controls all oracles to which $\mathcal{A}$ has the ability to access. $\mathcal{S}$ executes the Init phase including selecting biometric characteristics for each user, and answers $\mathcal{A}$'s queries as defined in the *Execute*, *Test*, and *Send* oracles. Thus, $\mathcal{A}$ succeeds in breaking the semantic security of AFEM if it can guess the bit $b$ that $\mathcal{S}$ uses during the *Test*-query. The proof uses a sequence of hybrids, starting from the real case and ending at the ideal (or simulation) case where the advantage of $\mathcal{A}$ is 0. Let $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_i}(\lambda)$ denote the advantage of $\mathcal{A}$ in the hybrid Hy.*i*. To prove the desired bound on $\mathrm{Adv}_{\mathcal{A}}(\lambda) = \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_0}(\lambda)$, we bound the effect of each change in the hybrid on the advantage of $\mathcal{A}$, and then illustrate that $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_7}(\lambda) \le Q_s/\|D\|$.

Hy.0. A real hybrid follows BAKE specification. Hy.0 corresponds to the real attack that the honest users have their key-pair $(pk_0, sk_0)$ (vs. $(pk_1, sk_1)$). *Execute* and *Send* are answered exactly as the honest users with their keys; *Reveal* to an instance of the participant $\mathcal{P}_0^i$ (vs. $\mathcal{P}_1^j$) is answered by issuing the session key (*i.e.*, $k_0 \leftarrow H(pk_0, pk_1, c_0, c_1, s_0', s_1)$ vs. $k_1 = H(pk_0, pk_1, c_0, c_1, s_0, s_1')$) that is generated by $\mathcal{P}_0^i$ (vs. $\mathcal{P}_1^j$) during the execution of the protocol (or $\perp$ if no session key is set); and *Test* to a fresh instance $\mathcal{P}_0^i$ (vs. $\mathcal{P}_1^j$) is answered after flipping a coin $b$, by either the output of $Reveal(\mathcal{P}_0^i)$ (vs. $Reveal(\mathcal{P}_1^j)$) or $sk \xleftarrow{\$} \{0,1\}^*$. By definition, we have $\mathrm{Adv}_{\mathcal{A}}(\lambda) = \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_0}(\lambda) = |2 \cdot \Pr[b = b'] - 1|$.

Hy.1. Hy.1 is identical to Hy.0 on simulating all the instances for *Send*, *Execute*, *Test* and *Corrupt* queries, except that we simulate the random oracle $\mathcal{H}$ on new queries $(pk_0, pk_1, c_0, c_1, s_0', s_1)$ and $(pk_0, pk_1, c_0, c_1, s_0, s_1')$, and we obtain two associated random outputs as session keys, either $k_0 \leftarrow H(pk_0, pk_1, c_0, c_1, s_0', s_1)$ or $k_1 \leftarrow H(pk_0, pk_1, c_0, c_1, s_0, s_1')$ in all of the sessions. For keeping consistent, the corresponding valid records $\big((pk_0, pk_1, c_0, c_1, s_0', s_1), k_0\big)$ and $\big((pk_0, pk_1, c_0, c_1, s_0, s_1'), k_1\big)$ are stored in the list $\Lambda_H$ that is used to give the same answer if the same query is asked twice. This hybrid excludes the collision, the protocol halts and $\mathcal{A}$ fails if any instance chooses any input of the random oracle that has been used. This is a perfect simulation of the random oracle $\mathcal{H}$, and we have

Claim 1. $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_0}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_1}(\lambda)|$ *is negligible.*

Proof. This claim is guaranteed by the collision resistance and one-wayness of the hash function. All executions will be halted if a collision occurs in the transcript $\big((pk_0, c_0), (pk_1, c_1)\big)$ since the inputs $pk_0, c_0, pk_1, c_1$ are simulated and chosen uniformly at random. Thus, the collision in the transcript is still regarded as negligible for convenience, and the probability is at most $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_0}(\lambda) \leq \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_1}(\lambda) + \frac{(Q_s+Q_e)^2}{2^\ell}$ via the birthday paradox, where $Q_s$ and $Q_e$ are the numbers of *Send* and *Execute* queries, and $\ell$ is the bit-length of the output of $\mathcal{H}$. □

Hy.2. *Execute* queries between compatible users, before corruption. In this hybrid, we first deal with the passive attacks between two compatible users because *Execute* models the behavior of the passive adversary. Indeed, there is a simple query to the *Execute* oracle (in either Hy.3 or Hy.2), and the transcript $(pk_0, c_0, pk_1, c_1)$ is returned to $\mathcal{A}$. In order to respond to $Execute(\mathcal{P}_0^i, \mathcal{P}_1^j)$, the public key $pk_0 \leftarrow \mathsf{AFEM.PubGen}(sk_0)$ for the sender $\mathcal{P}_0$ (vs. $pk_1 \leftarrow \mathsf{AFEM.PubGen}(\mathsf{sk}_1)$ for the receiver $\mathcal{P}_1$) with the associated correct secret key $sk_0$ derived from the biometric characteristics (i.e., $\mathcal{SK}$) for the sender $\mathcal{P}_0$ (vs. $sk_1$ for the receiver $\mathcal{P}_1$) is replaced by encrypting a random sampled $\overline{sk}_0$ (vs. $\overline{sk}_1$) from $\mathcal{SK}$. Additionally, the message $s_0$ for $\mathcal{P}_0$ (vs. the message $s_1$ for $\mathcal{P}_1$) is encrypted under a dummy public key $\overline{pk}_i \leftarrow \mathsf{AFEM.PubGen}(\overline{sk}_i)$ (for $i = 0, 1$) instead of $pk_i \leftarrow \mathsf{AFEM.PubGen}(sk_i)$. Thus, we obtain $\bar{c}_0 \leftarrow \mathsf{AFEM.Enc}(\overline{pk}_1, s_0)$ for $\mathcal{P}_0$ (vs. $\bar{c}_1 \leftarrow \mathsf{AFEM.Enc}(\overline{pk}_0, s_1)$ for $\mathcal{P}_1$). Next, random session keys $\bar{k}_0 \leftarrow \{0,1\}^\ell$ for $\mathcal{P}_0$ and $\bar{k}_1 \leftarrow \{0,1\}^\ell$ for $\mathcal{P}_1$ are drawn uniformly, and the list $\Lambda_H^{\mathrm{Hy}.2}$ stores the records $\big((\overline{pk}_0, \overline{pk}_1, \bar{c}_0, \bar{c}_1, s_0, s_1'), \bar{k}_0\big)$ and $\big((\overline{pk}_0, \overline{pk}_1, \bar{c}_0, \bar{c}_1, s_1, s_0'), \bar{k}_1\big)$. Thus, we consider $\mathcal{A}$ wins this hybrid if $\Lambda_H^{\mathrm{Hy}.2} \cap \Lambda_H \neq \emptyset$. To summarize what has been mentioned above, we utilize Hy.2 to exclude passive attacks between compatible users by *Execute*, where Hy.2 is indistinguishable from Hy.1 unless the two aforementioned records have been queried to $\mathcal{H}$. Thus, we have

Claim 2. $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_1}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_2}(\lambda)|$ *is negligible.*

Proof. This indistinguishability proof is guaranteed by AFEM that has been proved in Section 3.2.4, and under the decisional-LWE assumption and random oracle, the adversary $\mathcal{B}$ cannot distinguish the public key $\overline{pk}_i$ for $i = 0, 1$ generated in Hy.2 from the public key $pk_i$ for $i = 0, 1$ generated in previous hybrid. Additionally, the message $s_0$ for $\mathcal{P}_0$ (vs. the message $s_1$ for $\mathcal{P}_1$) is encrypted under a dummy public key $\overline{pk}_i$ instead of the real $pk_i \leftarrow \mathsf{AFEM.PubGen}(sk_i)$, thus, $\bar{c}_0 \leftarrow \mathsf{AFEM.Enc}(\overline{pk}_1, s_0)$ for $\mathcal{P}_0$ (vs. $\bar{c}_1 \leftarrow \mathsf{AFEM.Enc}(\overline{pk}_0, s_1)$ for $\mathcal{P}_1$). This is indistinguishability proof under the semantic security property of the AFEM scheme, and the adversary $\mathcal{B}$ cannot distinguish the encapsulated messages generated in Hy.2 and Hy.1. □

Hy.3. In this hybrid, we continue to deal with passive attacks between two incompatible users. Hy.3 is identical to Hy.2 except that we modify the answered way of *Execute* on how to compute the session key. The transcript is computed in the same way as Hy.2 but with independent two session keys. We use the secret sampled $\bar{k}_0$ and $\bar{k}_1$ from a uniform to replace the random oracle $\mathcal{H}$ for computing the session keys (i.e., $k_0$ and $k_1$) in all sessions generated via *Execute*. Below, the record $(pk_0, pk_1, c_0, c_1, s_0, s_1')$ is truncated to $(s_0, s_1')$ and for the sake of illustration, where $s_1^*$ is obtained from $\mathsf{AFEM.Dec}(sk_0', c_1')$ if $c_1' \leftarrow \mathsf{AFEM.Enc}(\overline{pk}_0, s_1)$ can be decrypted correctly, otherwise, $s_1^*$ is sampled from a uniform distribution. Then, the session key $k_0 = H(s_0 \| s_1^*)$ for $\mathcal{P}_0$ is replaced by a random $\bar{k}_0 \leftarrow \{0,1\}^\ell$. Likewise, we use $\bar{k}_1$ for the receiver instead of $k_1 = H(s_0^* \| s_1)$ at this stage, where $s_0^* \leftarrow \mathsf{AFEM.Dec}(sk_1', c_0')$ if $c_0' \leftarrow \mathsf{AFEM.Enc}(\overline{pk}_1, s_0)$ can be decrypted correctly, otherwise, $s_0^*$ is sampled from a uniform distribution. Thus, we have

Claim 3. $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_2}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_3}(\lambda)|$ *is negligible.*

Proof. In the aforementioned hybrid, $\mathcal{A}$'s probability of guessing $b$ in *Test* oracle is exactly 1/2 if *Test* query is made to a fresh instance that was activated via *Execute*. In the following Hybrids, we deal with active attacks via *Send*, where $\mathcal{A}$ has possibly generated the inputs of a *Send* query. To this end, *Send* is modified so that its output is chosen uniformly at random and independently of the biometric secret key. Notably, $Send_0(\mathcal{P}_0^i, \mathcal{P}_1^j)$ is the start-query for a user to initiate an execution of BAEK, followed by $Send_1(\mathcal{P}_1^j, pk_0)$ (vs. $Send_1(\mathcal{P}_0^i, pk_1)$) and $Send_2(\mathcal{P}_1^j, c_0)$ (vs. $Send_2(\mathcal{P}_0^i, c_1)$). □

Hy.4. We deal with active attacks using $Send_2(\mathcal{P}_1^j, c_0)$ between compatible users before corruption. $Send_2(\mathcal{P}_1^j, c_0)$ represents that $\mathcal{A}$ sends $c_0$ to $\mathcal{P}_0$. We modify the behavior of $\mathcal{P}_0^i$ to $Send_2(\mathcal{P}_0^i, c_1)$. At this stage, $c_1$ for $\mathcal{P}_0^i$ is adversarially generated without knowing the secret $sk_0$. Upon receiving $Send_2(\mathcal{P}_0^i, c_1)$, $\mathcal{S}$ checks whether $c_1$ for the instance $\mathcal{P}_0^i$ is either *valid* or *invalid*. If $c_1$ is valid, then outputs $c_0$ and the session key $k_0$ is assigned with a special value. If $c_1$ is invalid, then outputs $c_0$ and $\mathcal{S}$ samples $\overline{s'}_1$ and $k_0^*$ from a uniform and stores the record $(pk_0, pk_1, c_0, \bar{c}_1, s_0, \overline{s'}_1, k_0^*)$. The inconsistency can be detected if $\mathcal{H}$ has been asked with same $\bar{c}_1$ and $\overline{s'}_1$. More generally, $\mathcal{A}$ wins if a collision occurs. $Send_2(\mathcal{P}_0^i, c_1)$ queries are proceeded in a similar way, and we omit it here. Thus,

Claim 4. $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_3}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_4}(\lambda)|$ *is negligible.*

Proof. If $c_0$ is generated by $\mathcal{A}$ who used the public key $pk_1$ with an associated secret key $sk_1$, and the regenerated secret key is close to the previous one, *e.g.*, $dis(sk_1, sk_1') < \tau$, then the decapsulation oracle will output a correct value. Thus, $\mathcal{S}$ can declare $\mathcal{A}$ succeeds and terminates the hybrid. Otherwise, $\mathcal{S}$ selects a session key from

a uniform distribution. More generally, if $c_0$ is generated by $\mathcal{A}$ before corruption, then $\mathcal{S}$ queries the decapsulation oracle and obtains $\overline{s'}_0 \leftarrow \mathsf{AFEM.Dec}(sk'_1, \bar{c}_0)$. Next, if $s'_0 = \overline{s'}_0$, then $\mathcal{S}$ declares $\mathcal{A}$ succeeds. This case can only increase the advantage of $\mathcal{A}$. If $s'_0 \neq \overline{s'}_0$, then $\mathcal{S}$ sets a random sampled key from a uniform instead of $k_0$. This case introduces a negligible difference, guaranteed by the pseudorandomness of random oracle. □

Hy.5. We modify $Send_2$ between compatible users again to exclude corruptions. To answer $Send_2(\mathcal{P}_0^i, c_1)$ (vs. $Send_2(\mathcal{P}_1^j, c_0)$) where $c_1$ (vs. $c_0$) was used, $Send_2(\mathcal{P}_0^i, c_1)$ (vs. $Send_2(\mathcal{P}_1^j, c_0)$) outputs $c_0$ (vs. $c_1$). To get a session key $k_0$ (vs. $k_1$), $\mathcal{S}$ queries $\mathcal{H}$ with the appropriate $(pk_0, pk_1, c_0, c_1, s_0, s'_1)$, if there is a record, $\mathcal{A}$ wins. Otherwise, the session keys are set as $k_0^*, k_1^* \leftarrow \{0,1\}^\ell$.

Claim 5. $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_4}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_5}(\lambda)|$ *is negligible.*

Proof. This proof depends on the security of AFEM that is proved in Theorem 3.2. $\mathcal{A}$ issues the *Send* query, and $\mathcal{S}$'s behavior is defined in Definition 3.1 and proceeds as follows:

(1) $\mathcal{S}$ is given $pk_0$ and $c_0$ (vs. $pk_1$ and $c_1$).
(2) $\mathcal{S}$ generates $s$, collects and generates $sk'_0$ (vs. $sk'_1$).
(3) $\mathcal{S}$ responds to *Execute* queries by generating $(pk_0, c_0, pk_1, c_1)$, where $pk$ ($pk_0$ and $pk_1$) is computed by invoking $\mathsf{AFEM.PubGen}(\cdot)$ on input a dummy $sk$, and $c$ ($c_0$ and $c_1$) is computed by invoking $\mathsf{AFEM.Enc}(\cdot)$ on input 0 under the dummy $\overline{pk}$. The matching session keys are chosen uniformly at random.
(4) $\mathcal{S}$ responds to the $i$-th $Send_0$ queries by submitting an $sk$ to the PubGen oracle, and receives a public key $pk$, then gives it to $\mathcal{A}$.
(5) $\mathcal{S}$ responds to $Send_1(pk)$ and $Send_2(c)$ by checking whether $(pk, c)$ is previously used or adversarially generated. If it is the former, then the session key is calculated as the protocol description. Otherwise, the session key is chosen uniformly if $dis(sk_0, sk'_0) > \tau$, and $\mathcal{S}$ declares that $\mathcal{A}$ succeeds and terminates the hybrid if $dis(sk_0, sk'_0) < \tau$.
(6) At the end of this hybrid, $\mathcal{S}$ outputs 1 if and only if $\mathcal{A}$ succeeds.

Let $b$ be a choice bit defined in Equation.1. If $b = 0$, then the view of $\mathcal{A}$ in the above execution with $\mathcal{S}$ is identical to the view of $\mathcal{A}$ in Hy.4. If $b = 1$, the view of $\mathcal{A}$ in the above execution with $\mathcal{S}$ is identical to the view of $\mathcal{A}$ in Hy.5. Thus, this claim holds. □

Hy.6. We modify $Send_1(\mathcal{P}_1^j, pk_0)$ between incompatible users before a *Corrupt* query, where the public key $pk_0$ is generated by taking as input an incorrect secret key $sk_0$, and the output of $Send_1(\mathcal{P}_1^j, pk_0)$ is $\overline{pk}_1 \leftarrow \{0,1\}^*$, we set the session keys $k_0^*, k_1^* \leftarrow \{0,1\}^\ell$. $Send_1(\mathcal{P}_1^j, pk_0)$ is proceeded in a similar manner. Thus,

Claim 6. $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_5}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_6}(\lambda)|$ *is negligible.*

Proof. This proof is implied by Claim 2, the difference is that the dummy $\overline{pk}_1$ here is sampled from a uniform distribution, but in Claim 2 it is computed by inputting a random secret key. Thus, we omit the redundant analysis here. □

Hy.7. The secret keys/public keys are not known at the beginning, but just at the corruption time or to check whether $\mathcal{A}$ wins when $\mathcal{A}$ guesses the correct secret key at the very end only.

- $Execute(\mathcal{P}_0^i, \mathcal{P}_1^j)$: $\mathcal{S}$ randomly selects $\bar{c}_0$ (vs. $\bar{c}_1$) from a uniform distribution. If they are compatible, they are given the same random session key $k_0 = k_1$. Otherwise, they are given two independent random keys $k_0, k_1 \leftarrow \{0,1\}^\ell$.
- $Send_0(\mathcal{P}_0^i, \mathcal{P}_1^j)$ implies that the instance of $\mathcal{P}_0^i$ initiates an execution of BAKE, and $\mathcal{S}$ selects a random $sk_i$ from $\mathcal{SK}$ and calculates the public key $pk_i \leftarrow \mathsf{AFEM.PubGen}(sk_i)$.
- $Send_1(\mathcal{P}_1^j, pk_0)$ (vs. $Send_1(\mathcal{P}_0^i, pk_1)$) simulates the behavior of $\mathcal{P}_1^j$ (vs. $\mathcal{P}_0^i$) before a corruption. $\mathcal{S}$ selects the secret key $sk_1$ (vs. $sk_0$), and randomly selects a public key $pk_1 \leftarrow \{0,1\}^*$ (vs. $pk_0 \leftarrow \{0,1\}^*$). Otherwise, $\mathcal{S}$ selects $sk_1$ (vs. $sk_0$) and sets $pk_1 \leftarrow \mathsf{AFEM.PubGen}(sk_1)$ (vs. $pk_0 \leftarrow \mathsf{AFEM.PubGen}(sk_0)$).
- $Send_2(\mathcal{P}_0^i, c_1)$ (vs. $Send_2(\mathcal{P}_1^j, c_0)$) simulates the behavior of $\mathcal{P}_0^i$ (vs. $\mathcal{P}_1^j$) in the following cases.
  - Before a corruption, $\mathcal{S}$ outputs a randomly sampled session key $k_0$ (vs. $k_1$).
  - After a corruption, but $c_1$ (vs. $c_0$) has been generated before corruption. Particularly, $\mathcal{S}$ selects the message $s_1$ (vs. $s_0$) randomly, and simulates the ciphertext by sampling $\bar{c}_0$ (vs. $\bar{c}_1$) randomly. After that, $\mathcal{S}$ cannot decrypt $\bar{c}_0$ (vs. $\bar{c}_1$) to a correct value under the re-generated secret key $sk'_0$ (vs. $sk'_1$), and $\mathcal{S}$ samples a randomly chosen plaintext. Finally, $\mathcal{S}$ asks for the session key $k_0$ (vs. $k_1$) from $\mathcal{H}$ on the pair $(s_0, s'_1)$ (vs. $(s'_0, s_1)$).
  - After a corruption, $\mathcal{S}$ selects a message $s_1$ (vs. $s_0$) randomly and invokes $c_0 \leftarrow \mathsf{AFEM.Enc}(pk_1, s_0)$ (vs. $c_1 \leftarrow \mathsf{AFEM.Enc}(pk_0, s_1)$). After that, $\mathcal{S}$ decapsulates $\bar{c}_0$ (vs. $\bar{c}_1$) to a correct value under the re-generated secret key $sk'_0$ (vs. $sk'_1$). Finally, $\mathcal{S}$ asks for the session key $k_0$ (vs. $k_1$) from $\mathcal{H}$ on the pair $(s_0, s'_1)$ (vs. $(s'_0, s_1)$).
- $Corrupt(\mathcal{P}_0^i)$ (vs. $Corrupt(\mathcal{P}_1^j)$) implies that if this is the first corruption query involving $\mathcal{P}_0^i$ (vs. $\mathcal{P}_1^j$), one could first obtain a secret key $sk_0$ (vs. $sk_1$), then define the public key $pk_0$ (vs. $pk_1$) via the algorithm $\mathsf{AFEM.PubGen}(sk_0)$ (vs. $\mathsf{AFEM.PubGen}(sk_1)$).
- $Test(b)$ is answered using the defined session key according to the choice bit $b$.

At the very end, or at the time of corruption, the biometric characteristics are selected at random, and corresponding public keys are calculated via the secret keys. As a consequence, we have

Claim 7. $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_6}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_7}(\lambda)|$ *is negligible.*

Proof. This claim is guaranteed by the security of AFEM. □

In the final hybrid, $\mathcal{A}$'s view is independent of the real biometric secret keys chosen by $\mathcal{S}$ until the following cases happen. 1). $\mathcal{A}$ queries $Reveal(\mathcal{P}_0^i)$ or $Test(\mathcal{P}_0^i)$ (vs. $Reveal(\mathcal{P}_1^j)$ or $Test(\mathcal{P}_1^j)$) after $Send_1(\mathcal{P}_0^i, pk_1)$ (vs. $Send_1(\mathcal{P}_1^j, pk_0)$) for a malicious and valid $pk_1$ (vs. $pk_0$); 2) $\mathcal{A}$ queries $Reveal(\mathcal{P}_0^i)$ or $Test(\mathcal{P}_0^i)$ (vs. $Reveal(\mathcal{P}_1^j)$ or $Test(\mathcal{P}_1^j)$) after $Send_2(\mathcal{P}_0^i, c_1)$ (vs. $Send_2(\mathcal{P}_1^j, c_0)$) for a malicious and valid $c_1$ (vs. $c_0$). Thus, it holds that $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_7}(\lambda) \leq Q_s/\|D\|$, the way the session keys are defined is exactly the same as in the random or the real cases (chosen at random before corruption). The probability $b = b'$ is exactly $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Hy}_0}(\lambda) \leq Q_s/\|D\| + negl(\lambda)$. This concludes the proof of Theorem 5.2 with Claim 1 to 7.

**Table 1: Asymptotic Comparison, where $\widetilde{n}$ denotes the bit-length of a biometric string, $n$ denotes the size of a biometric vector set, and $\zeta$ is a constant. A round means an interaction between two participants, *i.e.*, one participant sends a message to the other, who then sends another message back. For the researches [19] and [21], the number of rounds is evaluated based on the building blocks (*e.g.*, aPAKE is regarded as one round), *i.e.*, the actual number of rounds depends on the instantiations, which may be bigger than the number presented in this table.**

| Scheme | | Technique | Round | Multiplication | Exponentiation | Hash | Secret Sharing | Symmetric Encryption |
|---|---|---|---|---|---|---|---|---|
| fPAKE-1 [19] | sender | Garbled Circuit | 5 | – | $3\widetilde{n}+\zeta$ | $4\widetilde{n}+\zeta$ | – | – |
| | receiver | | | – | $3\widetilde{n}+\zeta$ | $4\widetilde{n}+\zeta$ | – | – |
| fPAKE-2 [19] | sender | PAKE + Secret Sharing | 2 | – | $2\widetilde{n}+\zeta$ | $\widetilde{n}$ | $\zeta$ | – |
| | receiver | | | – | $2\widetilde{n}+\zeta$ | $\widetilde{n}$ | $\zeta$ | – |
| fuzzy aPAKE-1 [21] | sender | Secret Sharing + Oblivious Transfer | 2 | $2\widetilde{n}+\zeta$ | $4\widetilde{n}+\zeta$ | $3\widetilde{n}+\zeta$ | 1 | $2\widetilde{n}+\zeta$ |
| | receiver | | | $\widetilde{n}+\zeta$ | $4\widetilde{n}+\zeta$ | $2\widetilde{n}+\zeta$ | – | $2\widetilde{n}+\zeta$ |
| fuzzy aPAKE-2 [21] | sender | aPAKE | 2 | $4\widetilde{n}+\zeta$ | $5\widetilde{n}+\zeta$ | $4\widetilde{n}+\zeta$ | – | $2\widetilde{n}$ |
| | receiver | | | $2\widetilde{n}+\zeta$ | $5\widetilde{n}+\zeta$ | $2\widetilde{n}+\zeta$ | – | $\widetilde{n}$ |
| BAKE-1 | sender | Random Linear Codes | 1 | – | $\zeta$ | – | – | – |
| | receiver | | | $\zeta\widetilde{n}^2$ | $\zeta$ | $\zeta$ | – | – |
| BAKE-2 | sender | Secret Sharing + Polynomial Interpolation | 1 | $n^2$ | $2n+\zeta$ | $\zeta$ | $\zeta$ | – |
| | receiver | | | $\zeta n^3$ | $2n+\zeta$ | $\zeta$ | $\zeta$ | – |

**Table 2: Running Time (ms) on IrisCode and FVC2004.**

| | IrisCode | | | | FVC2004 | | | |
|---|---|---|---|---|---|---|---|---|
| | Case 1 | Case 2 | Case 3 | Case 4 | DB1 | DB2 | DB3 | DB4 |
| $m/n$ | 16 | 32 | 64 | 128 | 95 | 91 | 138 | 150 |
| PubGen | 54 | 55 | 56 | 58 | 29 | 28 | 43 | 45 |
| Enc | 101 | 110 | 111 | 110 | 151 | 148 | 221 | 232 |
| Dec | 71 | 79 | 85 | 116 | 315 | 293 | 598 | 631 |

**Table 3: Communication Cost (KB) on IrisCode and FVC2004.**

| | IrisCode | | | | FVC2004 | | | |
|---|---|---|---|---|---|---|---|---|
| | Case 1 | Case 2 | Case 3 | Case 4 | DB1 | DB2 | DB3 | DB4 |
| $m/n$ | 16 | 32 | 64 | 128 | 95 | 91 | 138 | 150 |
| $pk$ | 0.813 | 3.094 | 12.156 | 48.281 | 2.969 | 2.844 | 4.313 | 4.688 |
| $c$ | 0.844 | 3.125 | 12.188 | 48.313 | 2.672 | 2.566 | 3.867 | 4.219 |

For asynchronous BAKE, the security proof is identical to synchronous BAKE except for the Session $\text{sid}_0$, which is reduced to the security of AFEM.

# 6 EVALUATION

We show the asymptotic comparison with the state-of-the-art solutions and the experimental results on our implementation.

## 6.1 Asymptotic Comparison

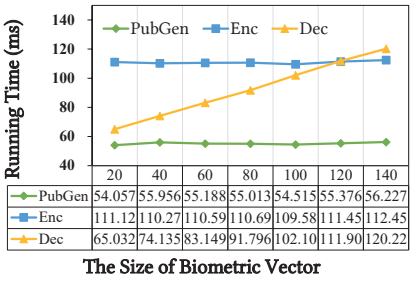
The asymptotic comparison with the state-of-the-art solutions is shown in Table 1, where our BAKE protocol for biometric vector is denoted as BAKE-1 and the one for biometric vector set is denoted as BAKE-2. Note that fPAKE [19] is a symmetric primitive, which gives biometric characteristics away to the receiver and thus dissatisfies the design goal of biometric privacy, while fuzzy aPAKE [21] is an asymmetric primitive that has similar goals to BAKE.

BAKE-1, fPAKE (instantiated as in [19]), and fuzzy aPAKE (instantiated as suggested in [21]) are designed for an $\widetilde{n}$-bit string, while BAKE-2 is designed for a set of cardinal $n$. The computation complexities of fPAKE-2 and fuzzy aPAKE-2 heavily rely on the underlying PAKE and aPAKE solutions. Therefore, in all the solutions for an $\widetilde{n}$-bit string, BAKE-1 is the most efficient one in terms of the computation complexity. Moreover, among these solutions, only BAKE-1 and BAKE-2 are one-round protocols that are suitable for the asynchronous secure messaging setting.

## 6.2 Experiments

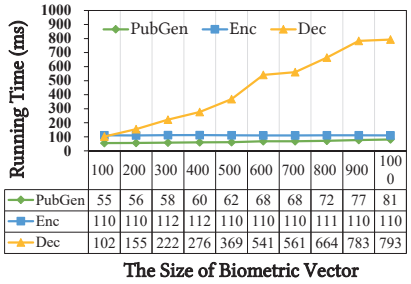
*6.2.1 Implementation.* To measure the performance of our BAKE protocols, we implemented a prototype in Python using a laptop computer, with the Intel Core i5-8300H CPU @ 2.30 GHz and 8 GB RAM. The group in both solutions is implemented with the elliptic curve Curve25519. BAKE-1 is implemented with the random linear code provided in Fuller et al. [24] and BAKE-2 is instantiated with Feldman's secret sharing [23]. To ensure the biometric keys from the same user are considered close, and the ones from different users are considered distant, the parameters (*e.g.*, $\tau$) were chosen by conducting experiments to obtain appropriate accuracy.

*6.2.2 Results on IrisCode and FVC2004.* We first investigate the performance of our BAKE protocols on two realistic datasets. For iris, we transform an IrisCode into 4 cases: a 16-dimensional vector, a 32-dimensional vector, a 64-dimensional vector, and a 128-dimensional vector. For fingerprint, we use four databases from the Third International Fingerprint Verification Competition (FVC2004) [36], in which DB1 and DB2 involve a similar size of fingerprint vector set, while the distorted DB3 and synthetic DB4 are extracted more noisy points, leading to big size $n$. The fingerprint images are preprocessed with the OpenCV library and the minutiae points are




| | 20 | 40 | 60 | 80 | 100 | 120 | 140 |
|---|---|---|---|---|---|---|---|
| PubGen | 54.057 | 55.956 | 55.188 | 55.013 | 54.515 | 55.376 | 56.227 |
| Enc | 111.12 | 110.27 | 110.59 | 110.69 | 109.58 | 111.45 | 112.45 |
| Dec | 65.032 | 74.135 | 83.149 | 91.796 | 102.10 | 111.90 | 120.22 |

**Figure 6: Running time of the AFEM construction for biometric vector, as the size of biometric vector increases from 20 to 140.**




| | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| PubGen | 55 | 56 | 58 | 60 | 62 | 68 | 68 | 72 | 77 | 81 |
| Enc | 110 | 110 | 112 | 112 | 110 | 110 | 110 | 111 | 110 | 110 |
| Dec | 102 | 155 | 222 | 276 | 369 | 541 | 561 | 664 | 783 | 793 |

**Figure 7: Running time of the AFEM construction for biometric vector, as the size of biometric vector increases from 100 to 1000.**

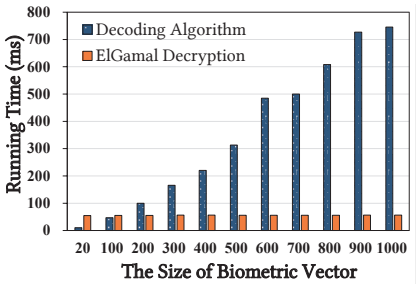
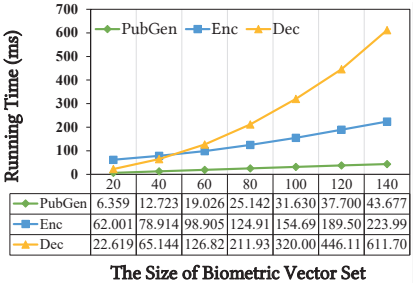
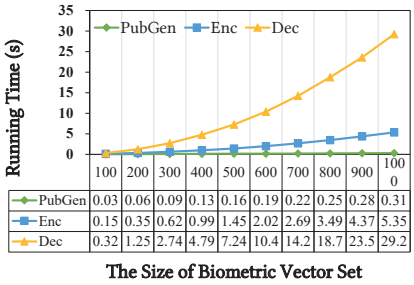



**Figure 8: Running time of different operations in Dec for biometric vector, as the size of biometric vector increases from 20 to 1000.**




| | 20 | 40 | 60 | 80 | 100 | 120 | 140 |
|---|---|---|---|---|---|---|---|
| PubGen | 6.359 | 12.723 | 19.026 | 25.142 | 31.630 | 37.700 | 43.677 |
| Enc | 62.001 | 78.914 | 98.905 | 124.91 | 154.69 | 189.50 | 223.99 |
| Dec | 22.619 | 65.144 | 126.82 | 211.93 | 320.00 | 446.11 | 611.70 |

**Figure 9: Running time of the AFEM construction for biometric vector set, as the size of biometric vector set increases from 20 to 140.**




| | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| PubGen | 0.03 | 0.06 | 0.09 | 0.13 | 0.16 | 0.19 | 0.22 | 0.25 | 0.28 | 0.31 |
| Enc | 0.15 | 0.35 | 0.62 | 0.99 | 1.45 | 2.02 | 2.69 | 3.49 | 4.37 | 5.35 |
| Dec | 0.32 | 1.25 | 2.74 | 4.79 | 7.24 | 10.4 | 14.2 | 18.7 | 23.5 | 29.2 |

**Figure 10: Running time of the AFEM construction for biometric vector set, as the size of biometric vector set increases from 100 to 1000.**

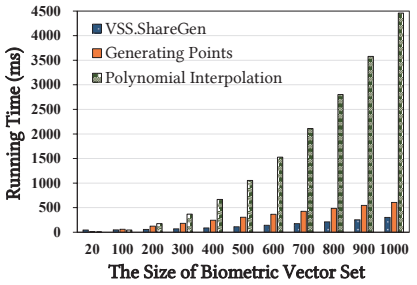



**Figure 11: Running time of different operations in Enc for biometric vector set, as the size of biometric vector set increases from 20 to 1000.**

extracted as coordinate values. For Algorithm 2, we choose $\mu = 4$, *i.e.*, each fingerprint vector is composed of 50 4-bit values.

The computation cost of each algorithm in our BAKE protocols is shown in Table 2. Our protocol on IrisCode is more efficient than that on FVC2004, since there is only one ElGamal-like operation in the AFEM construction for biometric vector while there are $n$ ElGamal-like operations in the AFEM construction for biometric vector set. However, both protocols are suitable for practical applications from the view of computation overhead.

The communication cost consists of transmitting the public key $pk$ in the KeyGen phase and transmitting the encapsulated message $c$ in the AKE phase, as shown in Table 3. Again, we can conclude that the two protocols are efficient in practice, even for the resource-limited network, in terms of the communication overhead.

*6.2.3 Further Results.* We then investigate the computation cost of our BAKE protocols with the size of biometric secret keys.

The time consumption of algorithms in BAKE-1 is illustrated in Figure 6 and Figure 7. The running time of PubGen is the smallest and increases slowly and the curve of Enc is smooth as the size of the biometric vector $m$ grows since Enc only involves an ElGamal-like encryption operation. For Dec, the time consumption grows substantially as $m$ increases and exceeds 0.5 seconds when $m = 600$, which is similar to the general biometric authentication [53]. Figure 8 further depicts the detailed time consumption of Dec, in which the decoding algorithm is dominated when $m \geq 200$.

The time consumption of algorithms in BAKE-2 is shown in Figure 9 and Figure 10, which implies that the running time of these three algorithms increases as the size of biometric vector set $n$ grows. The time consumption of Enc and Dec exceeds 1 second when $n = 500$ and $n = 200$, respectively, which is more efficient than existing tow-factor authentication methods (at least 13 seconds on average) [41]. We also experimented with different operations in Enc and Dec, as shown in Figure 11 and Figure 12. As $n$ increased, the most time-consuming operation in Enc is the polynomial interpolation. In Dec, secret reconstruction and polynomial evaluation are time-consuming operations when $n$ is big.

## 6.3 Comparison

We compare the computation and communication costs of BAKE-1 and BAKE-2 with the recent fuzzy aPAKE constructions [21], denoted as fuzzy aPAKE-1 and fuzzy aPAKE-2. From a practical point of view, we set $m = 64$ for the irises in BAKE-1 and employ the average size of fingerprint vector set in DB1, *i.e.*, $n = 95$, in BAKE-2. Fuzzy aPAKE-1 and fuzzy aPAKE-2 employ an iris as the "password". Since fuzzy aPAKE-1 and fuzzy aPAKE-2 are designed based on oblivious transfer protocols and standard asymmetric PAKE, respectively, different instantiations cause distinct performance. To make the comparison more convincing, as recommended in [21],

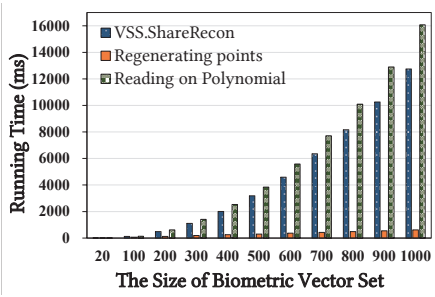



**Figure 12: Running time of different operations in** Dec **for biometric vector set, as the size of biometric vector set increases from 20 to 1000.**

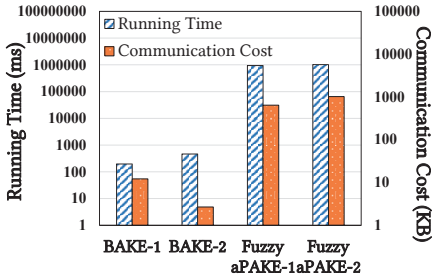



**Figure 13: Comparison of running time and communication cost.**

we instantiate fuzzy aPAKE-1 with the oblivious transfer protocol from [4] and instantiate fuzzy aPAKE-2 with OPAQUE [32].

The results are shown in Figure 13. The running time of fuzzy aPAKE-1 is more than 4000 times cost of BAKE-1, and more than 2000 times cost of BAKE-2. The running time of fuzzy aPAKE-2 is more than 5000 times cost of BAKE-1, and more than 2000 times cost of BAKE-2. The communication cost of fuzzy aPAKE-1 is more than 50 times cost of BAKE-1, and more than 230 times cost of BAKE-2. The communication cost of fuzzy aPAKE-2 is more than 80 times cost of BAKE-1, and more than 380 times cost of BAKE-2. Therefore, BAKE has an overwhelming advantage over fuzzy aPAKE in terms of both computation and communication costs.

## 7 RELATED WORK

According to whether the input secret is precise, we divide AKE into two categories: *precise AKE* and *fuzzy AKE*.

**Precise AKE.** In this category, the input secret cannot contain any typo or noise. The most common precise AKE includes password-based solutions [5, 32, 49] and PKI-based solutions [13, 45]. For resource-constrained devices, HB-type authentication protocols [26, 30, 33] were proposed, whose security mainly relies on the Learning Parity with Noise (LPN) problem. In practice, many messaging applications (*e.g.*, WhatsApp [51]) enable out-of-band authentication, assuming that users have access to an external channel, such as Short Authenticated Strings (SAS) [37, 42, 50]. Unfortunately, these solutions cannot apply to biometric AKE since the captured biometrics (*i.e.*, the input secret) contains unpredictable noises. Moreover, many solutions (*e.g.*, HB-type protocols and SAS-based solutions) require the two participants to share a secret, which violates data protection regulations when applied to biometrics.

**Fuzzy AKE.** In this category, the input secret may contain typos or noises when fed into cryptographic algorithms. Existing solutions can be classified into two types: *symmetric* and *asymmetric*.

In the symmetric solutions, the two communicating participants possess the same secrets to authenticate each other and negotiate a session key. Fuzzy extractor is a typical solution introduced by Dodis et al. [18]. In their solution, two similar biometric inputs can be used to extract the same randomness from public information while hiding these biometric inputs. Afterward, various fuzzy extractor variants emerged [9–11], and multi-factor AKE protocols were constructed based on the fuzzy extractor [38, 39]. However, these solutions need to store the (secret) randomness on the other participant, which may be exploited by an adversary. Another representative symmetric solution is presented by Dupont et al. [19], called fuzzy Password-Authenticated Key Exchange (fPAKE), which solved the problem of key exchange from noisy low-entropy password strings. However, their solutions need the other participant to store the password, which also violates data protection regulations when applied to biometrics.

As required in General Data Protection Regulation (GDPR), biometric characteristics should be protected against disclosure. For this goal, the study on asymmetric solutions, in which a participant has no access to the biometric characteristics of other participants, attracts the attention of researchers. To the best of our knowledge, the only suitable asymmetric solutions are the fuzzy asymmetric PAKE protocols proposed by Erwig et al. [21]. They considered asymmetric PAKE and fuzzy PAKE simultaneously, and gave two constructions based on secret sharing and standard asymmetric PAKE, respectively. However, this primitive is only proposed for fuzzy vectors (*e.g.*, password). Moreover, their constructions involve frequent interactions and are not suitable for asynchronous messaging scenarios. In contrast, BAKE is designed for both fuzzy vectors and fuzzy sets, and our constructions involve only one-round interaction and support both synchronous and asynchronous scenarios.

## 8 CONCLUSION

To facilitate secure messaging in social life, we propose a BAKE framework that supports both synchronous and asynchronous scenarios and does no need to store any secret, including biometric characteristics, in a terminal. We present a cryptographic primitive called AFEM to enable only the participant with similar biometric characteristics to obtain a message that is encapsulated with the corresponding public key. We also give two constructions for AFEM and initiates them with the iris and fingerprint in our BAKE framework. The security analysis demonstrates that BAKE is secure and the experimental results show the practicality of BAKE.

## ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under grants No. U1836202, 61772383, 62172303, 61802214, 62076187.

## REFERENCES


[1] Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. 2015. Solving the Closest Vector Problem in 2ˆn Time - The Discrete Gaussian Strikes Again!. In *Proc. of FOCS*. IEEE Computer Society.

[2] Muhammad Ejaz Ahmed, Il-Youp Kwak, Jun Ho Huh, Iljoo Kim, Taekkyung Oh, and Hyoungshick Kim. 2020. Void: A Fast and Light Voice Liveness Detection System. In *Proc. of USENIX Security Symposium*. USENIX Association.

[3] László Babai. 1986. On Lovász' Lattice Reduction and the Nearest Lattice Point Problem. *Comb.* 6, 1 (1986), 1–13.

[4] Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. 2017. A Framework for Efficient Adaptively Secure Composable Oblivious Transfer in the ROM. *IACR Cryptol. ePrint Arch.* (2017). http://eprint.iacr.org/2017/993.

[5] José Becerra, Dimiter Ostrev, and Marjan Skrobot. 2018. Forward Secrecy of SPAKE2. In *Proc. of IEEE ProvSec*.

[6] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. 1998. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract). In *Proc. of TCC*. ACM.

[7] Mihir Bellare, David Pointcheval, and Phillip Rogaway. 2000. Authenticated Key Exchange Secure against Dictionary Attacks. In *Proc. of EUROCRYPT*. Springer.

[8] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei P. Skorobogatov, and Ross J. Anderson. 2014. Chip and Skim: Cloning EMV Cards with the Pre-play Attack. In *Proc. of S & P*. IEEE Computer Society.

[9] Xavier Boyen. 2004. Reusable Cryptographic Fuzzy Extractors. In *Proc. of CCS*. ACM.

[10] Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam D. Smith. 2005. Secure Remote Authentication Using Biometric Data. In *Proc. of EUROCRYPT*. Springer.

[11] Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam D. Smith. 2021. Reusable Fuzzy Extractors for Low-Entropy Distributions. *J. Cryptol.* 34, 1 (2021), 2.

[12] Melissa Chase, Apoorvaa Deshpande, Esha Ghosh, and Harjasleen Malvai. 2019. SEEMless: Secure End-to-End Encrypted Messaging with less</> Trust. In *Proc. of CCS*. ACM.

[13] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. 2020. A Formal Security Analysis of the Signal Messaging Protocol. *J. Cryptol.* 33 (2020), 1914–1983.

[14] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2018. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. In *Proc. of CCS*. ACM.

[15] Cas Cremers, Jaiden Fairoze, Benjamin Kiesl, and Aurora Naska. 2020. Clone Detection in Secure Messaging: Improving Post-Compromise Security in Practice. In *Proc. of CCS*. ACM.

[16] John Daugman. 1993. High Confidence Visual Recognition of Persons by a Test of Statistical Independence. *IEEE Trans. Pattern Anal. Mach. Intell.* 15, 11 (1993), 1148–1161.

[17] John Daugman. 2016. Information Theory and the IrisCode. *IEEE Trans. Inf. Forensics Secur.* 11, 2 (2016), 400–409.

[18] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. 2004. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Proc. of EUROCRYPT*. Springer.

[19] Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakoubov. 2018. Fuzzy Password-Authenticated Key Exchange. In *Proc. of EUROCRYPT*. Springer.

[20] Simon Eberz, Kasper Bonne Rasmussen, Vincent Lenders, and Ivan Martinovic. 2015. Preventing Lunchtime Attacks: Fighting Insider Threats With Eye Movement Biometrics. In *Proc. of NDSS*. The Internet Society.

[21] Andreas Erwig, Julia Hesse, Maximilian Orlt, and Siavash Riahi. 2020. Fuzzy Asymmetric Password-Authenticated Key Exchange. In *Proc. of ASIACRYPT*. Springer.

[22] Facebook. 2017. Messenger Secret Conversatinos, Technical Whitepaper. https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf. (2017).

[23] Paul Feldman. 1987. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *Proc. of FOCS*. IEEE Computer Society.

[24] Benjamin Fuller, Xianrui Meng, and Leonid Reyzin. 2013. Computational Fuzzy Extractors. In *Proc. of ASIACRYPT*. Springer.

[25] Yang Gao, Wei Wang, Vir V. Phoha, Wei Sun, and Zhanpeng Jin. 2019. EarEcho: Using Ear Canal Echo for Wearable Authentication. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3 (2019), 81:1–81:24.

[26] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. 2008. HB$^{\#}$: Increasing the Security and Efficiency of HB$^{+}$. In *Proc. of EUROCRYPT*. Springer.

[27] Wire Swiss GmbH. 2018. Wire Security Whitepaper. https://wire-docs.wire.com/download/Wire+Security+Whitepaper.pdf. (2018).

[28] Yiliang Han. 2021. Design of An Active Infrared Iris Recognition Device. In *Proc. of IPEC*. IEEE Computer Society.

[29] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 2008. *An Introduction to Mathematical Cryptography*. Springer.

[30] Nicholas J. Hopper and Manuel Blum. 2001. Secure Human Identification Protocols. In *Proc. of ASIACRYPT*. Springer.

[31] Anil K. Jain, Salil Prabhakar, Lin Hong, and Sharath Pankanti. 1999. FingerCode: A Filterbank for Fingerprint Representation and Matching. In *Proc. of CVPR*. IEEE Computer Society.

[32] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. 2018. OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-computation Attacks. In *Proc. of EUROCRYPT*. Springer.

[33] Ari Juels and Stephen A. Weis. 2005. Authenticating Pervasive Devices with Human Protocols. In *Proc. of CRYPTO*. Springer.

[34] Xiangyu Liu, Shengli Liu, Dawu Gu, and Jian Weng. 2020. Two-Pass Authenticated Key Exchange with Explicit Authentication and Tight Security. In *Proc. of ASIACRYPT*. Springer.

[35] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. 2009. *Handbook of Fingerprint Recognition, Second Edition*. Springer.

[36] Biometric System Lab-University of Bologna. 2004. Fingerprint Verification Competition 2004. http://bias.csr.unibo.it/fvc2004/. (2004).

[37] Sylvain Pasini and Serge Vaudenay. 2006. SAS-Based Authenticated Key Agreement. In *Proc. of PKC*. Springer.

[38] David Pointcheval and Sébastien Zimmer. 2008. Multi-factor Authenticated Key Exchange. In *Proc. of ACNS*. Springer.

[39] Mingping Qi, Jianhua Chen, and Yitao Chen. 2018. A Secure Biometrics-based Authentication Key Exchange Protocol for Multi-server TMIS using ECC. *Comput. Methods Programs Biomed.* 164 (2018), 101–109.

[40] Aditya Singh Rathore, Weijin Zhu, Afee Daiyan, Chenhan Xu, Kun Wang, Feng Lin, Kui Ren, and Wenyao Xu. 2020. SonicPrint: a Generally Adoptable and Secure Fingerprint Biometrics in Smart Devices. In *Proc. of MobiSys*. ACM.

[41] Ken Reese, Trevor Smith, Jonathan Dutson, Jonathan Armknecht, Jacob Cameron, and Kent E. Seamons. 2019. A Usability Study of Five Two-Factor Authentication Methods. In *Proc. of SOUPS*. USENIX Association.

[42] Lior Rotem and Gil Segev. 2018. Out-of-Band Authentication in Group Messaging: Computational, Statistical, Optimal. In *Proc. of CRYPTO*. Springer.

[43] Jörg Schwenk, Marcus Brinkmann, Damian Poddebniak, Jens Müller, Juraj Somorovsky, and Sebastian Schinzel. 2020. Mitigation of Attacks on Email End-to-End Encryption. In *Proc. of CCS*. ACM.

[44] Signal. 2021. Signal Technical Information. https://signal.org/docs/. (2021).

[45] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. 2020. Post-Quantum Authentication in TLS 1.3: A Performance Study. In *Proc. of NDSS*. The Internet Society.

[46] Statista. 2021. Most popular global mobile messenger apps as of January 2021, based on number of monthly active users. https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/. (2021).

[47] Nirvan Tyagi, Paul Grubbs, Julia Len, Ian Miers, and Thomas Ristenpart. 2019. Asymmetric Message Franking: Content Moderation for Metadata-Private End-to-End Encryption. In *Proc. of CRYPTO*. Springer.

[48] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. 2015. SoK: Secure Messaging. In *Proc. of S &P*. IEEE Computer Society.

[49] Mathy Vanhoef and Eyal Ronen. 2020. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *Proc. of IEEE S & P*.

[50] Serge Vaudenay. 2005. Secure Communications over Insecure Channels Based on Short Authenticated Strings. In *Proc. of CRYPTO*. Springer.

[51] WhatsApp. 2016. WhatsApp Encryption Overview. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf. (2016).

[52] Wikipedia. 2021. Public Key Infrasstructure. https://en.wikipedia.org/wiki/Public_key_infrastructure. (2021).

[53] Cong Wu, Kun He, Jing Chen, Ziming Zhao, and Ruiying Du. 2020. Liveness is Not Enough: Enhancing Fingerprint Authentication with Behavioral Biometrics to Defeat Puppet Attacks. In *Proc. of USENIX Security Symposium*. USENIX Association.

[54] Xiu Xu, Haiyang Xue, Kunpeng Wang, Man Ho Au, and Song Tian. 2019. Strongly Secure Authenticated Key Exchange from Supersingular Isogenies. In *Proc. of ASIACRYPT*. Springer.

[55] Chen Yan, Yan Long, Xiaoyu Ji, and Wenyuan Xu. 2019. The Catcher in the Field: A Fieldprint based Spoofing Detection for Text-Independent Speaker Verification. In *Proc. of CCS*. ACM.

[56] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. 2015. Authenticated Key Exchange from Ideal Lattices. In *Proc. of EUROCRYPT*. Springer.

[57] Bing Zhou, Jay Lohokare, Ruipeng Gao, and Fan Ye. 2018. EchoPrint: Two-factor Authentication using Acoustics and Vision on Smartphones. In *Proc. of MobiCom*. ACM.

[58] Kai Zhou and Jian Ren. 2018. PassBio: Privacy-Preserving User-Centric Biometric Authentication. *IEEE Trans. Inf. Forensics Secur.* 13, 12 (2018), 3050–3063.