Hindawi
Security and Communication Networks
Volume 2018, Article ID 6107912, 14 pages
https://doi.org/10.1155/2018/6107912

Research Article

Construction of a New Biometric-Based Key Derivation Function and Its Application

Minhye Seo,[1] **Jong Hwan Park**,[2] **Youngsam Kim**,[3] **Sangrae Cho**,[3] **Dong Hoon Lee**,[1] **and Jung Yeon Hwang**[3]

[1]*Graduate School of Information Security, Korea University, Seoul, Republic of Korea*
[2]*Department of Computer Science, Sangmyung University, Seoul, Republic of Korea*
[3]*Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea*

Correspondence should be addressed to Jung Yeon Hwang; videmot@etri.re.kr

Received 9 April 2018; Accepted 1 November 2018; Published 2 December 2018

Academic Editor: Leandros Maglaras

Biometric data is user-identifiable and therefore methods to use biometrics for authentication have been widely researched. Biometric cryptosystems allow for a user to derive a cryptographic key from noisy biometric data and perform a cryptographic task for authentication or encryption. The fuzzy extractor is known as a prominent biometric cryptosystem. However, the fuzzy extractor has a drawback in that a user is required to store user-specific *helper data* or receive it online from the server with additional trusted channel, to derive a correct key. In this paper, we present a new biometric-based key derivation function (BB-KDF) to address the issues. In our BB-KDF, users are able to derive cryptographic keys solely from their own biometric data: users do not need any other user-specific helper information. We introduce a security model for the BB-KDF. We then construct the BB-KDF and prove its security in our security model. We then propose an authentication protocol based on the BB-KDF. Finally, we give experimental results to analyze the performance of the BB-KDF. We show that our proposed BB-KDF is computationally efficient and can be deployed on many different kinds of devices.

1. Introduction

Biometric data is unique to the individual, and, therefore, there has been a lot of research on using biometrics for authentication systems. There are two main types of biometrics: physical (e.g., a fingerprint, face, iris, or hand) and behavioral (e.g., a handwritten signature or keyboard dynamics such as rhythm, speed, and use of the left or right shift key). Recently, much research has been conducted to develop models to combine several biometrics for user authentication [1–4]. In comparison to previous non-biometric-based authentication systems, biometric-based systems do not require the user to remember passwords or possess security tokens. Instead, the server authenticates the user just by using his/her unique physical or behavioral features (i.e., who you *are*). Various kinds of devices that collect biometric data can now be found in the surrounding environment, making deployment of a biometric-based authentication system practicable.

In the early stages of research on biometric authentication, biometric templates were developed and used as such. In other words, in the enrollment phase, a template extracted from a user's biometric data was stored in the server, and, in the authentication phase, a newly extracted template was compared with the stored one. However, biometric data in authentication have given rise to a host of privacy issues [5]. Once a biometric template is stored in the server or database, the raw biometric data can be recovered, compromising user privacy [6, 7]. In addition, once a particular piece of biometric data has been compromised, it cannot be used again for authentication, yet since there is not a wide array of biometric data available to use for authentication, it cannot continually be replaced. With the threat of compromise and limited biometric resources for authentication, recent research has focused on protection of biometric templates.

The research can be categorized into two types: (1) cancelable biometrics and (2) biometric cryptosystems. In

cancelable biometrics, biometric data are altered via noninvertible transformations [8–10]. The transformed template is stored in a server for matching. If the transformed template is compromised, the system cancels the validity of the template and reissues a new template with different parameters. However, the template is used simply for an authentication purpose. In a biometric cryptosystem, the user derives a deterministic key from his/her own noisy biometric data (with help of additional data which was generated at an initial step). The derived key is used to operate cryptographic mechanisms for various security goals such as authentication, encryption, and message integrity [11–14].

The main challenge in biometric cryptosystems is how to deal with the noisy trait of biometric data while keeping the privacy of biometric data in a cryptographic aspect. Biometric readings may differ every time, even though they are derived from the same individual. Nevertheless, the same digital key should be able to be derived if the difference between two pieces of noisy biometric data is within a certain minimal threshold.

The fuzzy extractor was a biometric cryptosystem first introduced by Dodis et al. in 2004 [13] and has been widely researched since then. The fuzzy extractor consists of two algorithms. The first algorithm, *Gen*, takes a piece of noisy data as an input and outputs a key as well as so-called *helper data*, which is public information. The second algorithm, *Rep*, takes as inputs the helper data and another piece of noisy data and outputs a key. If the two pieces of noisy data are quite similar, then the two algorithms, *Gen* and *Rep*, will generate the same key with the help of the helper data. Although the fuzzy extractor is a useful cryptographic primitive in that it derives a cryptographically secure key from noisy biometric data, it has a drawback. A user must personally receive the user-specific helper data from the *Gen* algorithm and keep it as a security token to input into the *Rep* algorithm, or the user must receive it from the server via additionally established channel, whenever authentication is required. Since the helper data is of substantial size (approximately 33,569MB or 19,372GB when the error occurs by 15% or 20% between measurements, respectively, in [15]), when a large number of users participate in the system, a considerable network bandwidth is used, which causes heavy network traffic and overwhelms the system.

1.1. Our Contributions. In this paper, we propose a new efficient biometric-based key derivation function (BB-KDF) that will require no user-dependent randomized information (such as helper data of the fuzzy extractor) and only use the user's biometric data to generate a cryptographic key for authentication. In our BB-KDF, biometric data are assumed to be encoded to a biometric vector of real numbers as in [16, 17]. Though the BB-KDF is described for biometrics in the paper, it is applicable to other fuzzy or noisy data.

The proposed BB-KDF is conceptually simple and computationally efficient, and, therefore, we may deploy the BB-KDF in a wide range of devices for the Internet of Things (IoT). In the IoT, various devices are connected to the Internet and potentially to each other, requiring authentication to limit access to a particular user or users. Since the proposed construction has little computational overhead, the BB-KDF is deployable on many different kinds of devices, from low-performance devices (e.g., sensors) to high-performance devices (e.g., smartphones).

In the proposed BB-KDF, a cryptographic key is computed only by using a user's biometric vector and a public parameter that is generated when the system is set up and is not associated with the user or his/her biometric vector. Therefore, in the BB-KDF, no user biometric-dependent data is available to adversaries. Capturing it precisely, we define a security model for the BB-KDF and prove the security of our construction using this model.

As an application, we propose an authentication protocol using the BB-KDF. The Schnorr identification scheme is used to construct a biometric authentication scheme, except that a secret key is derived from the BB-KDF. The security of the authentication scheme is taken from that of the Schnorr identification scheme, based on the hardness of the discrete-logarithm problem.

Finally, we give experimental results to show the performance of the BB-KDF. To analyze the efficiency of the BB-KDF more fairly, we conducted experiments in various settings considering both a device specification and the length of biometric vector. We also give comparison between the BB-KDF and an existing KDF (e.g., password-based KDF) in terms of computational cost. BB-KDF is approximately 146 times faster than PBKDF1 in PKCS♯5 (setting the iteration number to 1000) on smartphones.

1.2. Organization. The remainder of this paper is organized as follows. In Section 1.3, we briefly review related work in the area. In Section 2, we introduce basic notions and definitions. In Section 3, we define the notion of BB-KDF and its security model. We describe our proposed BB-KDF construction and experimental results in Section 4, followed by its security analysis in Section 5. In Section 6, we present an authentication protocol as an application of the BB-KDF. Section 7 is dedicated to analyzing the efficiency of the BB-KDF with implementation results. Conclusions are given in Section 8.

1.3. Related Work

Key Derivation Function. The goal of a key derivation function (KDF) is to derive a pseudorandom key (i.e., cryptographically secure key), taking a source of initial keying material as an input. The main key material used has been imperfect, namely, *not* uniformly random or pseudorandom, such as physical sources [18, 19], a shared Diffie-Hellman value [20, 21], a user password [21–24], or any bit sequence from a source of more or less entropy [21]. In case of an imperfect input material, the extract-then-expand approach has been used to derive a cryptographic key [25]. A password-based key derivation function (PB-KDF) has been the most typical one, which takes a user-chosen password as an input. PB-KDFs have been constructed by using a variety of cryptographic primitives such as block ciphers [23], stream cipher [24], and cryptographic hash functions

[21, 22]. In PKCS♯5, the *de facto* standard for password-based cryptography, the methods to construct a practical PB-KDF are provided [26]. PB-KDF has been employed in various environments. For example, PBKDF2, described in PKCS♯5, was used in Android's full disk encryption [27], WPA/WPA2 encryption process [28], FileVault MAC OS X [29, 30], and Winrar [31]. In the case of applying a PB-KDF to existing cryptosystems (e.g., authentication systems), users basically have to memorize their passwords, which is quite inconvenient. In this paper, we propose the construction of a new KDF that requires no additional information to be memorized. The proposed KDF can replace existing PB-KDFs in various environments.

Biometric Cryptosystems. As a means of biometric template protection, a biometric cryptosystem represents a comprehensive biometric-based key generation and encryption system [14]. In previously existing biometric cryptosystems, biometric-dependent, but public, information (a.k.a. *helper data*) has been generated using a biometric template to correctly recover a cryptographic key. Biometric data are noisy and there have been various research projects on derivation of cryptographic keys from noisy data. Fuzzy commitment schemes have been constructed based on binary error-correcting codes [11], in which biometric data are represented as binary strings, the similarity of which is estimated by using the hamming distance metric. Fuzzy vault schemes have corrected errors in noisy data by using polynomial interpolation [12]. In such schemes, biometric data have been represented as a set of elements in a finite field. In 2004, Dodis et al. generalized the fuzzy vault and the fuzzy commitment and proposed the fuzzy extractor [13]. Since then, the information-theoretic fuzzy extractors have been widely researched [32–34], and, recently, the computational fuzzy extractor was also constructed [35]. A reusable fuzzy extractor was first constructed in 2004 by Boyen [36], and, until recently, very little further research has been done on this. Over a decade later, in 2016, a reusable fuzzy extractor was proposed with no limitation on the number of correlated readings of the source [15]. Since then, reusable fuzzy extractors based on various assumptions have been proposed [37, 38]. In the fuzzy extractor, user-specific, biometric-dependent information is generated to correct the noise in biometric data, causing a great deal of inconvenience to the user because he/she should carry it in person or receive it from the server whenever the authentication is required. On the other hand, the proposed (biometric-based) KDF does not require any user-specific information in authentication, mitigating the user's inconvenience.

Fuzzy Signature. Fuzzy signature uses the user's biometric data as a signing key. The concept of fuzzy signature was first proposed by Takahashi et al. in [39] and has since been improved by relaxing the requirements for construction or increasing efficiency [40]. However, all the fuzzy signature schemes proposed so far are not robust since the user's biometric data can be directly recovered from the (public) verification key or signature [41].

2. Preliminaries

In this section, we introduce the basic notations and definitions that will be used throughout the paper.

Basic Notations. Let $\mathbb{R}$ denote the set of all real numbers. Let $[0, 1)$ denote the set of all real numbers x satisfying $0 \le x < 1$. If $b \in \mathbb{R}^n$ for an integer $n > 0$, then b denotes a vector of length n whose components are all real numbers. Similarly, if $t \in [0, 1)^n$ for an integer $n > 0$, then each component of t is a real number in a range of $[0, 1)$. Let $[1, n]$ denote the set of all integers x satisfying $1 \le x \le n$ for a positive integer n. The symbol "$\|$" denotes concatenation. If $a \in \mathbb{R}$, then "$\lceil a \rfloor$" denotes a rounding to the nearest integer.

2.1. Entropy and Hash Functions

Min-Entropy. For a probability distribution $\mathcal{X}$, we use the notation $\Pr_{\mathcal{X}}(x)$ to denote the probability assigned by $\mathcal{X}$ to the value x. We often omit the subscript when the probability distribution is clear from the context. For a probability distribution $\mathcal{X}$ over $\{0, 1\}^k$, we define its *min-entropy* as the minimum integer m such that, for all $x \in \{0, 1\}^k$, $\Pr_{\mathcal{X}}(x) \le 2^{-m}$. We denote the min-entropy of such $\mathcal{X}$ by $\widetilde{\mathbf{H}}_\infty(\mathcal{X})$.

Conditional Entropy. The (average) min-entropy of $\mathcal{X}$ given $\mathcal{Z}$ is defined as follows:

$$\widetilde{\mathbf{H}}_\infty(\mathcal{X} \mid \mathcal{Z}) = -\log\left(\mathbf{E}_{z \leftarrow \mathcal{Z}}\left[\max_x \Pr(\mathcal{X} = x \mid \mathcal{Z} = z)\right]\right). \tag{1}$$

The *collision probability* of $\mathcal{X}$ given $\mathcal{Z}$ is given by

$$\begin{aligned}\mathbf{Col}(\mathcal{X} \mid \mathcal{Z}) &= \mathbf{E}_{z \leftarrow \mathcal{Z}}\left[\sum_x \Pr(\mathcal{X} = x \mid \mathcal{Z} = z)^2\right] \\ &= \mathbf{E}_{z \leftarrow \mathcal{Z}}\left[\mathbf{Col}(\mathcal{X}|_{\mathcal{Z}=z})\right],\end{aligned} \tag{2}$$

and the *collision entropy* of $\mathcal{X}$ given $\mathcal{Z}$ is equal to

$$\mathbf{H}_2(\mathcal{X} \mid \mathcal{Z}) = -\log\left(\mathbf{E}_{z \leftarrow \mathcal{Z}}\left[\sqrt{\mathbf{Col}(\mathcal{X}|_{\mathcal{Z}=z})}\right]\right)^2. \tag{3}$$

For any joint distribution $\mathcal{X}$ and $\mathcal{Z}$, these three notions are related as follows:

$$\mathbf{H}_2(\mathcal{X} \mid \mathcal{Z}) \ge -\log[\mathbf{Col}(\mathcal{X} \mid \mathcal{Z})] \ge \widetilde{\mathbf{H}}_\infty(\mathcal{X} \mid \mathcal{Z}). \tag{4}$$

Definition 1. Let ℓ and k be integers and $\mathcal{H}$ be a family of hash functions h with domain $\{0, 1\}^\ell$ and range $\{0, 1\}^k$. We say that the family $\mathcal{H}$ is δ-***almost universal (δ-AU)*** if for every pair of different inputs x, y from $\{0, 1\}^\ell$ it holds that $\Pr[h(x) = h(y)] \le \delta$, where the probability is taken over $h \in_R \mathcal{H}$. For a given probability distribution $\mathcal{X}$ over $\{0, 1\}^\ell$, we say that $\mathcal{H}$ is δ-AU with respect to $\mathcal{X}$ if $\Pr[h(x) = h(y)] \le \delta$, where the probability is taken over $h \in_R \mathcal{H}$ and $x, y \in_R \mathcal{X}$ conditioned to $x \ne y$.

Clearly, a family $\mathcal{H}$ is δ-AU if it is δ-AU with respect to all distributions over $\{0, 1\}^\ell$. If $\delta = 2^{-k}$ then we say that $\mathcal{H}$ is universal.

2.2. Leftover Hash Lemma. The leftover hash lemma (LHL) for the conditional min-entropy [42] is given as follows.

Lemma 2. *Let $(\mathcal{X}, \mathcal{Z})$ be a joint distribution on $\mathcal{X} \times \mathcal{Z}$. Let ℓ and k be integers, let $\mathcal{X}$ be a probability distribution over $\{0,1\}^{\ell}$, and let H be a family of hash function with domain $\{0,1\}^{\ell}$ and range $\{0,1\}^{k}$. If $\mathcal{H}$ is $(1/2^{k} + \epsilon)$-almost universal with respect to $\mathcal{X}$, U_k being uniform over $\{0,1\}^{k}$ and h being uniformly chosen over $\mathcal{H}$, then the statistical distance between $h(\mathcal{X})$ and U_k, given h and $\mathcal{Z}$, is defined as follows:*

$$\mathbf{SD}\left(h(\mathcal{X}); U_k \mid h, \mathcal{Z}\right) \leq \frac{1}{2} \cdot \sqrt{2^{-L} + \epsilon} \tag{5}$$

where $L = \widetilde{\mathbf{H}}_{\infty}(\mathcal{X} \mid \mathcal{Z}) - k$ is the entropy loss.

For universal hashing, we need $L \approx 2\log(1/\epsilon)$ to make the statistical distance smaller than ϵ. Note that we need $\epsilon \ll 1/2^{k}$ (say, $\epsilon \approx 1/2^{2k}$). There are examples of families with $\epsilon = 1/2^{k}$ (i.e., $(2/2^{k})$-AU families) that generate outputs that are easily distinguishable from uniform distribution. For example, if $\mathcal{H}$ is a family of pairwise independent hash functions with k-bit outputs, and we define a new family $\mathcal{H}'$ that is identical to $\mathcal{H}$ except that it replaces the last bit of output with 0, then the new family has a collision probability of $2/2^{k}$, yet its output (which has a fixed bit of output) is trivially distinguishable from uniform distribution. Therefore, to be secure, we need $\epsilon \ll 1/2^{k}$.

2.3. Pseudorandom Function. A pseudorandom function (PRF) [43] is a function $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ where $\mathcal{K}$ is a key space, $\mathcal{X}$ is a domain, and $\mathcal{Y}$ is a range. For a given security parameter k, we say that a PRF F is secure if, for all efficient algorithms $\mathcal{A}$,

$$\begin{aligned}&\left|\Pr\left[v \longleftarrow_R \mathcal{K} : \mathcal{A}^{F(v,\cdot)}\left(1^{k}\right) = 1\right]\right.\\&\left. - \Pr\left[f \longleftarrow_R \mathsf{Funs}(\mathcal{X}, \mathcal{Y}) : \mathcal{A}^{f(\cdot)}\left(1^{k}\right) = 1\right]\right| \\&\leq \mathrm{negl}(k),\end{aligned} \tag{6}$$

where $\mathsf{Funs}(\mathcal{X}, \mathcal{Y})$ denotes a set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

3. Definition and Security Framework

3.1. Biometric-Based Key Derivation Function (BB-KDF). A biometric-based key derivation function consists of two (PPT) algorithms. (1) The public parameter generation algorithm **Setup**(1^{k}), on input of security parameter 1^{k}, outputs a threshold vector t, a PRF F_v with its key v, and a hash function H as a public parameter PP. (2) The deterministic algorithm **KDF**(PP, b), on inputs of the public parameter PP and a biometric vector b, outputs a derived key, denoted by K, of length k. (**Setup**, **KDF**) has the following property.

(Correctness) If $\mathbf{d}(b, b') < t$ and $PP \longleftarrow$**Setup**(1^{k}), then $\Pr[\mathbf{KDF}(PP, b') = \mathbf{KDF}(PP, b)] \geq 1 - \delta$, where δ is an error related to a false rejection rate.

Here, $\mathbf{d}(b, b') < t$ implies that each component-wise difference of two vectors b and b', denoted by $|b_i - b_i'|$, is within a threshold value t_i for $i \in [1, n]$. If t is a single value, then $\mathbf{d}(b, b') < t$ implies that $|b_i - b_i'| < t$ for all $i \in [1, n]$. A distance $\mathbf{d}$ between two vectors is defined in detail below. In this paper, we construct a practical biometric-based key derivation function (see Section 4 for detailed explanations). With biometric data of w-bit size and vector of length n, each component of the biometric vector, denoted by b_i, has a bit size of (w/n).

(i) **Biometric space $\mathcal{S}$.** The biometric vector consists of n real number components. We will refer to the set of all biometric vectors as biometric space, expressed as follows:

$$\mathcal{S} \subset [0, s)^{n} \subset \mathbb{R}^{n} \quad \text{for some positive integer } s \tag{7}$$

For example, a biometric vector b can be expressed as $b = (0.004002, 0.007759, \ldots)$, which is a transformation of fingerprint images [44] through [45].

(ii) **Distance d**. If the distance between two different biometric vectors is smaller than the threshold value or, to be exact, the threshold vector, two outputs of BB-KDF with these biometric vector inputs will be the same. The distance between two biometric vectors is defined as follows.

For two vectors $b = (b_1, \ldots, b_n)$ and $b' = (b_1', \ldots, b_n')$,

$$\mathbf{d}\left(b, b'\right) = \left(d_1, \ldots, d_n\right) \ \textit{where}\ d_i = \left|b_i - b_i'\right|. \tag{8}$$

For $t = (t_1, \ldots, t_n)$, if $\mathbf{d}(b, b') < t$, this means that $d_i < t_i$ for all $i \in [1, n]$. If t is a single value, $\mathbf{d}(b, b') < t$ implies that $d_i < t$ for all $i \in [1, n]$.

Biometrics, such as fingerprint and face, is represented as a real number vector, and a number of cryptographic protocols using this format of biometrics have been proposed [16, 17]. Under the condition where a feature vector is extracted as a real number vector, our BB-KDF scheme, described above, can be applied while improving overall efficiency of the authentication system.

3.2. Security Model. In this section, we introduce two security requirements: privacy of a biometric vector and security of the derived key.

3.2.1. Privacy of a Biometric Vector. A biometric vector is unique to each user and therefore user-identifiable. Since a biometric vector is sensitive information, privacy should be guaranteed. Maintaining privacy of a biometric vector implies that no significant information about the biometric vector will become public. The only available public information in our BB-KDF scheme is its public parameter, from which an adversary can extract no meaningful information about users' biometric vector.

Leakage of information results in loss of entropy. If our key generation scheme permits only a negligible loss of entropy of a biometric vector from public side information, e.g., the public parameter, we can say that it guarantees the

privacy of a biometric vector. The privacy of a biometric vector will be deemed to be maintained when the following definition is met.

Definition 3. Let $\mathscr{B}$ be an input source and $\mathscr{Z}$ be a side information. When $\mathscr{Z}$ is public, the BB-KDF maintains the privacy of a biometric vector if ε, represented in below formula, is negligible.

$$\widetilde{\mathbf{H}}_{\infty}(\mathscr{B}) - \widetilde{\mathbf{H}}_{\infty}(\mathscr{B} \mid \mathscr{Z}) \leq \varepsilon \tag{9}$$

3.2.2. Security of the Derived Key. The key derived by our BB-KDF should be cryptographically secure. In other words, an adversary should not be able to figure out the key generated from BB-KDF or obtain any nonnegligible information from the biometric vector associated with the derived key. We can define the security level afforded by the BB-KDF by computing the statistical distance (SD) between two distributions, the distribution of the key derived by BB-KDF and uniform distribution.

Assume that BB-KDF outputs a k-bit key string. Given a string K of length k, the adversary should not be able to distinguish whether K is derived from the BB-KDF or chosen randomly from $\{0,1\}^k$. In other words, it should be nearly impossible to tell the difference between the distribution of the key derived from BB-KDF and uniform distribution over $\{0,1\}^k$. We consider the derived key to be secure when the following definition is met.

Definition 4. Let BB-KDF be a biometric-based key derivation function that outputs k-bit value, $\mathscr{B}$ be an input source of BB-KDF, and $\mathscr{Z}$ be a side information. When $\mathscr{Z}$ is public, the key derived from the BB-KDF is secure if ε, represented in below formula, is negligible.

$$\mathbf{SD}\left(\mathsf{BB\text{-}KDF}(\mathscr{B});\ U_k \mid \mathscr{Z}\right) \leq \varepsilon \tag{10}$$

where U_k is uniform distribution over $\{0,1\}^k$.

4. Construction

4.1. Design Goal. The design goal of the BB-KDF is to derive cryptographic keys only from users' biometric vector. A cryptographic primitive, called the *fuzzy extractor*, has been widely used to generate keys from biometric data. In the fuzzy extractor, additional information called *helper data* is generated to deal with noisy biometric data. The purpose of helper data, which is derived from the user's biometric data, is to generate or regenerate the same key using several user inputs of biometric data within a threshold range. In other words, a user who wants to extract a key from his/her biometric data has to access the helper data in addition to his/her own biometric data, which is burdensome to the user.

By contrast, in the BB-KDF scheme, each user is able to derive a correct key using only a public parameter and user's own biometric vector. Recall that the public parameter is not associated with the user's biometric vector and is configured when the system is deployed. Therefore, a user does not need to *store or obtain* any additional information, such as the helper data, and derives a cryptographic key only by using his/her own biometric vector.

4.2. Biometric-Based Key Derivation Function. Our construction of the BB-KDF consists of two algorithms, Setup and KDF. Let n be a positive integer to denote the length of a biometric vector.

Setup(1^k). It takes a security parameter k as input and then generate a public parameter PP through the following steps:

(1) It sets a threshold value $t_i \in [0,1)$ for each $i \in [1,n]$. Let $t = (t_1, \ldots, t_n)$ denote a threshold vector.

(2) It selects a PRF F and a key v from the key space of F uniformly at random.

(3) It selects a cryptographic hash function $H : \{0,1\}^{\ell} \longrightarrow \{0,1\}^k$.

The public parameter PP is defined by $PP = (t, F_v, H)$. Note that the key v for PRF F is also included in the public parameter.

KDF(PP, b). It takes PP and a biometric vector $b = (b_1, \ldots, b_n) \in \mathbb{R}^n$ as input and then generates a k-bit cryptographic key K through the following steps:

(1) It computes $h_i = 1/2t_i$ and $f_{h_i}(b_i) = \lceil b_i \cdot h_i \rfloor$ for each $i \in [1,n]$.

(2) It computes $\mathscr{X} = F_v(f_{h_1}(b_1) \parallel \ldots \parallel f_{h_n}(b_n))$ and outputs $K = H(\mathscr{X})$.

Given i-th components of both a biometric vector and a threshold vector; b_i and t_i, a biometric value b_i' satisfies $f_{h_i}(b_i) = \lceil b_i \cdot h_i \rfloor = \lceil b_i' \cdot h_i \rfloor = f_{h_i}(b_i')$ only if it is in the range of $[b_i - t_i, b_i + t_i)$. Note that the error-correcting data, denoted by h_i, have no correlation with a biometric vector and are set only by a threshold vector.

In our construction, we use (nonadaptive) PRF so that the input source $\mathscr{X}$ of a hash function is uniformly distributed over $\{0,1\}^{\ell}$. Let $F : \mathbb{N}^n \longrightarrow \{0,1\}^{\ell}$ be a PRF family. In the **Setup** algorithm, the description of PRF and its key v are released as public parameters. While it is unusual to publish the PRF key, we do so because we only require the fact that the distribution of the output of PRF is indistinguishable from the uniform distribution. Note that this technique has been used in previous constructions [46–49]. This is a different concept to the helper data of the *fuzzy extractor* in that the PRF key v is a system parameter and, therefore, independent of the individual. Note that the helper data is biometric-dependent information and each user has its own helper data.

4.3. Accuracy Analysis. To illustrate how our BB-KDF derives a key from similar biometric vectors, we give a simple example. Assume that the length of a biometric vector is equal to 3; i.e., $n = 3$, and a threshold vector is set to $t = (0.01, 0.02, 0.004)$. Given a biometric vector $b = (23.784, 50.125, 9.369) \in \mathbb{R}^3$, the BB-KDF outputs a key K_b as follows:

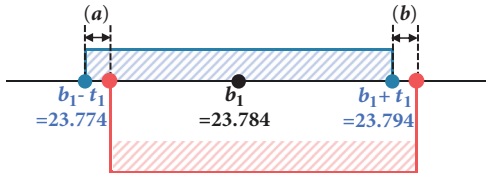


Figure 1: Example in case of b_1=23.784, t_1=0.01.

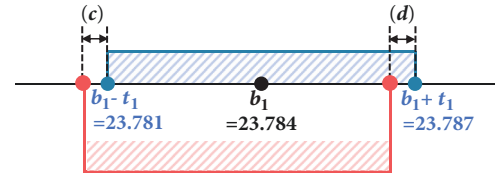


Figure 2: Example in case of b_1=23.784, t_1=0.003.

(1) $(h_1, h_2, h_3) = (1/0.02, 1/0.04, 1/0.008) = (50, 25, 125)$

(2) $f_{h_1}(b_1) = \lceil (23.784) \cdot (50) \rfloor = 1189$

$f_{h_2}(b_2) = \lceil (50.125) \cdot (25) \rfloor = 1253$

$f_{h_3}(b_3) = \lceil (9.369) \cdot (125) \rfloor = 1171$

(3) $K_b = H(F_v(1189, 1253, 1171))$

Next we consider another biometric vector $b' = (23.789, 50.117, 9.371) \in \mathbb{R}^3$ which is close to b. Note that b' and b satisfy $\mathbf{d}(b, b') < t$ because $\mathbf{d}(b, b') = (|23.784 - 23.789|, |50.125 - 50.117|, |9.369 - 9.371|) = (0.005, 0.008, 0.002) < (0.01, 0.02, 0.004) = t$. For b', the BB-KDF outputs a key $K_{b'}$ as follows.

(1) $(h_1, h_2, h_3) = (1/0.02, 1/0.04, 1/0.008) = (50, 25, 125)$

(2) $f_{h_1}(b_1') = \lceil (23.789) \cdot (50) \rfloor = 1189$

$f_{h_2}(b_2') = \lceil (50.117) \cdot (25) \rfloor = 1253$

$f_{h_3}(b_3') = \lceil (9.371) \cdot (125) \rfloor = 1171$

(3) $K_{b'} = H(F_v(1189, 1253, 1171))$

Hence we have that $K_{b'}$ is exactly equal to K_b.

By multiplying $h_i = 1/2t_i$ by b_i, for each $i \in [1, n]$, we allow only a biometric vector $b' = (b_1', \ldots, b_n')$, which satisfies the formula below, to generate the same key as K_b.

$$\forall i \in [1, n], \quad b_i' \in [b_i - t_i, b_i + t_i) \tag{11}$$

Let us take a closer look at this, using the first components of a biometric vector b and a threshold vector t, denoted by b_1 and t_1, respectively. Given the biometric value b_1, we regard $b_1' \in [b_1 - t_1, b_1 + t_1)$ as a valid piece of biometric vector, namely, one from the same user. By multiplying $h_1 = 1/2t_1$, we can assure that both b_1 and b_1' have the same output as the output of a rounding function; i.e., $\lceil b_1 \cdot h_1 \rfloor = \lceil b_1' \cdot h_1 \rfloor$, if b_1' is within a range of $[b_1 - t_1, b_1 + t_1)$; however, some errors may arise. In the case of $b_1 = 23.784$ and $t_1 = 0.01$, the range of valid biometric values is $[23.774, 23.794]$, as shown in blue on Figure 1, whereas the actual range that satisfies $\lceil b_1 \cdot h_1 \rfloor = 1189$ is equal to $[23.77475\ldots, 23.79475\ldots)$. As a matter of fact, we deal with a range that is slightly higher than what we expected, as shown in red on Figure 1.

As described in Figure 1, there are two gaps between two ranges, denoted by (a) and (b). Biometric values within the gap on the left side of Figure 1, $(a)[23.774, 23.77475\ldots)$, will not succeed in deriving a correct key, even though it is valid data provided by an authorized user. This exemplifies the case of a false rejection, where a genuine user is incorrectly rejected as an imposter. It is difficult to reduce the false rejection rate (FRR) by taking a design-level approach. However, since the margin of error is quite small, this can be managed by establishing appropriate policies for this system.

By contrast, biometric values within the gap on the right side of Figure 1, $(b)[23.794, 23.79475\ldots)$, will succeed in deriving a correct key, even though it is not a valid one from the user. This exemplifies the case of false acceptance, where an imposter is incorrectly accepted as a genuine user. We can reduce the false acceptance rate (FAR) by increasing the length of a biometric vector. The more we increase the length of a biometric vector b, the smaller the probability that all the components of b will fall within FAR-related ranges. We should configure the length of a biometric vector so the probability that an imposter will generate a correct key is negligible.

The fundamental reason for the difference between two ranges is that the rounding function used to derive a key has a discrete property, whereas a biometric vector is continuous. However, as explained above, there are strategies to deal with these gaps. Furthermore, we are able to close the gap between two ranges, thereby reducing both FRR and FAR, by decreasing threshold values. In the example described above, if we set the threshold value t_1 to 0.003, the errors (i.e., gaps on both sides) converge to almost zero.

In Figure 2, the gaps on both the left and right sides indicate the FAR and FRR, respectively. The left side of those gaps, denoted by (c), represents the range $[23.78095, \ldots, 23.781)$, and the gap on the right side, denoted by (d), represents the range $[23.78695, \ldots, 23.787)$. So, we need to ensure that (c) and (d) shown in Figure 2 are smaller than (b) and (a) in Figure 1, respectively. If we set the threshold value to 0.002 or 0.001, the errors (i.e., gaps on both sides) are eliminated altogether. Note that since biometric value b_1 is presented with three decimal places, the threshold value t_1 cannot be smaller than 0.001.

4.4. Performance Evaluation. In order to show accuracy performance of our BB-KDF, we give experimental results in terms of FAR and FRR. The experiments were conducted on two public datasets, FVC2002 (DB1, DB2) [44], where each dataset consists of 100 users with 8 samples per user.

As input for the BB-KDF evaluation, we used a real-valued fixed-length fingerprint vector which is generated by [50]. For example, an input vector is represented as $(\ldots, 0.013335, 0.032443, 0.009297, \ldots)$. The method of [50] consists of two main components, that is, minutiae descriptor extraction and kernel learning-based transformation. As a minutia descriptor it makes use of variable-sized Minutia Cylinder-Code (MCC) [45] which is considered as a

Table 1: FAR/FRR of implementation of the proposed BB-KDF.

FVC2002-DB1			FVC2002-DB2		
Threshold t	FAR	FRR	Threshold t	FAR	FRR
0.05	0.0117	0.825	0.05	0.0164	0.789
0.058	0.0311	0.737	0.053	0.0513	0.665
0.063	0.0513	0.66	0.056	0.0822	0.624
0.07	0.1008	0.578	0.06	0.1071	0.582
0.073	0.1424	0.534	0.062	0.1426	0.548
0.075	0.1913	0.489	0.067	0.183	0.483
0.076	0.2186	0.474	0.069	0.2095	0.458
0.078	0.2578	0.442	0.073	0.238	0.4
0.08	0.3004	0.417	0.075	0.2788	0.365
0.082	0.3461	0.356	0.076	0.3343	0.33

state-of-the-art minutiae descriptor. The kernel learning-based transformation consists of three main parts, that is, kernel matrix computation using a set of training samples for MCC, generation of a projection matrix based on KPCA (Kernel Principal Component Analysis) [51], and transformation of MCC descriptor to fixed-length real-valued representation. For training data we used 300 samples, that is, for 100 users with 3 samples per user in each FVC2002 dataset. A projection matrix is defined by a 300×299 matrix and so the length of real-valued fixed-length fingerprint vector is set to be 299. This means that the length of a biometric vector to be taken as input for the BB-KDF is 299.

The accuracy performance of BB-KDF can be analyzed by using FAR and FRR. FAR is used to indicate error rate of accepting an imposter as a legitimate user. FRR is used to indicate error rate of rejecting a legitimate user as an imposter. Table 1 shows experimental results of FAR and FRR according to a threshold value t. In real applications, achieving low FRR is more important than reasonably high FRR because most users can bear with 2-4 retrials as long as the authentication system guarantees expected level of security against imposters. For example, when $t = 0.063$ and $t = 0.053$, we have FAR=0.05 (and FRR=0.66) for both of FVC2002-DB1 and FVC2002-DB2. Hence it shows the possibility that the BB-KDF can work effectively in practice.

The experiments were conducted under a PC with processor core i7-4790 CPU @ 3.60GHz with 8 GB RAM. The BB-KDF was implemented using Python hashlib library. For a PRF and a hash we used HMAC-SHA256 and SHA256, respectively. Given a real-valued biometric vector of length 299, the computation of the BB-KDF per sample took about 0.07 ms.

5. Security Analysis

In this section, we analyze the security of the BB-KDF constructed in Section 4. First, we prove the security of our construction in security models described in Section 3.2. We then present an example to show that the more the information about a biometric vector is revealed, the more the loss of entropy occurs. This implies the extended version of a security model described in Section 3.2.2, where more information related to a biometric vector is included in a side information $\mathcal{Z}$.

5.1. Security Proof

5.1.1. Privacy of a Biometric Vector. To ensure the privacy of a biometric vector, we first identify the information revealed to potential adversaries in the BB-KDF scheme. In the BB-KDF scheme described in Section 4, only a public parameter is revealed, which consists of a threshold vector, denoted by t, a PRF and its key, denoted by F_v, and a hash function, denoted by H. The public parameter is not related to users' biometric vectors, and, therefore, an adversary will not be able to extract any meaningful biometric information from it. As a result, the public parameter does not cause loss of entropy, as represented by the formula below:

$$\widetilde{\mathbf{H}}_{\infty}(\mathcal{B}) \approx \widetilde{\mathbf{H}}_{\infty}(\mathcal{B} \mid \mathcal{Z}) \tag{12}$$

where $\mathcal{B}$ is an input source (i.e., a biometric vector) and $\mathcal{Z}$ is side information (i.e., a public parameter).

Therefore, the privacy of a biometric vector in the BB-KDF scheme fits Definition 3 and is therefore considered secure.

5.1.2. Security of the Derived Key. To ensure that the derived key is cryptographically secure, the distribution of the output of the BB-KDF should be uniform. In other words, the distribution of the key derived by the BB-KDF should be statistically indistinguishable from uniform distribution over $\{0,1\}^k$. We can estimate the security of the derived key by computing the statistical distance (SD) between two distributions, the distribution of the key derived by the BB-KDF, and uniform distribution. In our security proof, we assume that an input source of BB-KDF has enough entropy to guarantee the security of the derived key.

Assumption. We assume that there is an input source of BB-KDF whose entropy is large enough to derive a secure key, which is a quite strong assumption since it is hard to be satisfied in practice. However, the gap between reality and

assumption can be bridged by using multifactor biometrics as an input source [1–4]. In other words, we combine several biometrics to create an input biometric vector, which may have higher entropy than the one using only one biometrics modality.

Theorem 5. *Let $\mathcal{Z}$ be a side information which consists of a threshold vector $t = (t_1, \ldots, t_n) \in [0,1)^n$, a PRF and its key, denoted by F_v and a hash function H. Let H be a randomly chosen universal hash function from $\{0,1\}^{\ell}$ to $\{0,1\}^k$ and let $\mathcal{B}$ be an input source of BB-KDF. Then* $\mathbf{SD}(\mathsf{BB\text{-}KDF}(\mathcal{B}); U_k \mid \mathcal{Z})$*, the statistical distance between* $\mathsf{BB\text{-}KDF}(\mathcal{B})$ *and the uniform distribution over $\{0,1\}^k$ given $\mathcal{Z}$, is at most*

$$\frac{1}{2} \cdot \sqrt{2^{k+n(1-\log(s))+\sum_{i=1}^{n}\log(t_i)}}, \tag{13}$$

where n is length of biometric vector, s is upper bound of biometric space, d is the number of decimal places, and t_i is i-th threshold value over $[0,1)$.

Proof. Let $\mathcal{B} = (b_1, \ldots, b_n) \in [0,s)^n \subset \mathbb{R}^n$ be a biometric vector of length n. Then $\mathsf{BB\text{-}KDF}(\mathcal{B}) = H(F_v(f_{h_1}(b_1) \parallel \ldots \parallel f_{h_n}(b_n))$, where $f_{h_i}(b_i) = \lceil b_i \cdot h_i \rfloor$ and $h_i = 1/2t_i$. Let $\mathcal{X} = F_v(f_{h_1}(b_1) \parallel \ldots \parallel f_{h_n}(b_n))$. Note that $\mathcal{X}$ represents an input of a hash function, which is generated by computing a PRF on inputs a biometric vector and a threshold vector. If we assume that H is a randomly chosen universal hash function from ℓ to k bits, using LHL (Leftover Hash Lemma), we can calculate the statistical distance between $H(\mathcal{X})$ and U_k given $\mathcal{Z}$.

$$\mathbf{SD}\left(H(\mathcal{X});\ U_k \mid \mathcal{Z}\right) \le \frac{1}{2} \cdot \sqrt{2^{-L}} \tag{14}$$

where $L = \widetilde{\mathbf{H}}_{\infty}(\mathcal{X} \mid \mathcal{Z}) - k$ is the entropy loss.

We first compute $\widetilde{\mathbf{H}}_{\infty}(\mathcal{X} \mid \mathcal{Z})$, the min-entropy of $\mathcal{X}$ given $\mathcal{Z}$. The min-entropy of $\mathcal{X}$ given $\mathcal{Z}$ is defined as

$$\begin{aligned} &\widetilde{\mathbf{H}}_{\infty}(\mathcal{X} \mid \mathcal{Z}) \\ &\quad = -\log\left(\mathbf{E}_{z \leftarrow \mathcal{Z}}\left[\max_{x} \Pr(\mathcal{X} = x \mid \mathcal{Z} = z)\right]\right) \end{aligned} \tag{15}$$

Since F_v is a PRF where its key v is randomly chosen, $\mathcal{X}$, the output of F_v is uniformly random over $\{0,1\}^{\ell}$, with d decimal places. If not, we can construct a distinguisher that distinguishes the output of F_v from random string, where v is randomly chosen. Note that it does not matter whether the PRF key v is released in public or not. By using this property, we derive the following formula:

$$\begin{aligned} \max_{x} \Pr(\mathcal{X} = x \mid \mathcal{Z} = z) &= \prod_{i=1}^{n}\left(\frac{2t_i \cdot 10^d}{s \cdot 10^d}\right) \\ &= \prod_{i=1}^{n}\left(\frac{2t_i}{s}\right) \end{aligned} \tag{16}$$

where t_i is the i-th threshold value over $[0,1)$, s is the upper bound of input space, and n is the length of biometric vector.

Let us think about the i-th component of a biometric vector b_i. First, since b_i is included in $[0,s)$, for some integer s with d decimal places, the size of the space of b_i is $(s \cdot 10^d)$. Second, the range of a biometric value b_i' which satisfies $\lceil b_i \cdot h_i \rfloor = \lceil b_i' \cdot h_i \rfloor$ is $[b_i - t_i, b_i + t_i)$, where b_i is a biometric value, t_i is a threshold value, and $h_i = 1/2t_i$. Therefore, the size of the space of b_i' is $(2t_i \cdot 10^d)$. In conclusion, if a threshold vector t and a PRF F_v are public, the probability that $\mathcal{X} = F_v(\lceil b_1 \cdot h_1 \rfloor \parallel \ldots \parallel \lceil b_n \cdot h_n \rfloor)$ is equal to $\prod_{i=1}^{n}((2t_i \cdot 10^d)/(s \cdot 10^d))$. Using the leftover hash lemma, the statistical distance between $H(\mathcal{X})$ and U_k given $\mathcal{Z}$ is as follows:

$$\begin{aligned} \mathbf{SD}\left(H(\mathcal{X});\ U_k \mid \mathcal{Z}\right) &\le \frac{1}{2} \cdot \sqrt{2^{-(\widetilde{\mathbf{H}}_{\infty}(\mathcal{X}\mid\mathcal{Z})-k)}} \\ &= \frac{1}{2} \cdot \sqrt{2^{k+\log(\prod_{i=1}^{n}(2t_i/s))}} \\ &= \frac{1}{2} \cdot \sqrt{2^{k+n(1-\log(s))+\prod_{i=1}^{n}\log(t_i)}} \end{aligned} \tag{17}$$

where $k = 128$ and $n = 50$ (where k is the length of the derived key in bits and n is a length of biometric vector); we can compute the statistical distance in various settings. To make it simpler, we assume that all the components of a threshold vector (i.e., t_i for all $i \in [1,n]$) have the same value, denoted by t. Table 2 represents the statistical distance in various settings. Therefore, if the statistical distance is negligible in a parameter setting, meeting Definition 4, then the key derived by the BB-KDF in that setting is secure. □

Table 2: Statistical distance in various settings.

Threshold (t)	Upper bound of biometric space (s)		
	10^2	10^3	10^4
0.1	2^{-137}	2^{-212}	2^{-287}
0.01	2^{-212}	2^{-287}	2^{-362}
0.001	2^{-287}	2^{-362}	2^{-437}

5.2. Leakage of a Biometric Vector. If an adversary is not able to obtain any information associated with a user's biometric vector, the probability that the adversary can get meaningful information about the biometric vector will be negligible. However, if any additional information, beyond a public parameter, about the biometric vector is revealed, the entropy of the biometric vector will be reduced and an adversary may be able to extract enough other biometric vectors to launch an effective attack.

We evaluate the probability that an adversary will succeed in being able to obtain the user's biometric vector based on the degree of leakage. We assume that the length of a biometric vector is equal to 1; that is, a biometric vector consists of only one component, denoted by b. We also assume that the biometric space $\mathcal{S}$ is $[0,100) \subset \mathbb{R}$ (i.e., $s = 100$) and that b has three decimal places. Let $b = 23.784$ and $t = 0.01$ (where t is a threshold value). Note that, when we say the adversary will succeed, we mean that the adversary can figure out a biometric vector b' to generate the same key as b does. In other words, a successful adversary figures out b' such that $K_b = K_{b'}$ where $\mathbf{KDF}(PP, b) \longrightarrow K_b$ and $\mathbf{KDF}(PP, b') \longrightarrow K_{b'}$.

In the next subsections, we give details to show how the degree of information leakage on a biometric vector affects its recovery, more concretely, Section 5.2.1 for the case where b is never leaked, Section 5.2.2 for the case where the integer part of b is leaked, and Section 5.2.3 for a generalized assumption where the length of a biometric vector is equal to $n(> 1)$.

5.2.1. No Leakage of a Biometric Vector. We assume that there is no leakage of a biometric vector. To compute the probability of success of an adversary, we first compute the key, denoted by K_b, derived from a biometric vector b. Recall that, given $b = 23.784$ and $h = 1/2t = 50$, $K_b = H(F_v(\lceil b \cdot h \rfloor)) = H(F_v(\lceil (23.784) \cdot (50) \rfloor)) = H(F_v(1189))$.

In order to derive the same key as K_b, a biometric vector b' should satisfy $\lceil b' \cdot h \rfloor = 1189$. In other words, an attack succeeds if an adversary guesses b' within the range of $[23.770, 23.790)$. Therefore, the probability that an adversary will succeed in guessing the correct b' (i.e., $b' \in [23.770, 23.790)$) is

$$\Pr[guess] = \frac{20}{100 \cdot 10^3} = 0.02\%. \tag{18}$$

Since the size of the space of b is equal to $100 \cdot 10^3$ and the number of biometric values within the range of $[23.770, 23.790)$ is equal to 20, the probability that an adversary will succeed is only 0.02%. As a result, if there is no leakage of a biometric vector, the probability that an adversary will figure out underlying biometric vector (i.e., a biometric vector from which a valid key is derived) is reasonably low.

5.2.2. Leakage of the Integer Part. We assume that the integer part of a biometric vector, denoted by b, is leaked. For example, let us assume that an adversary knows the integer part of b, which is 23. In this case, b is within the range of $[23, 24)$, and, therefore, it satisfies $1150 \leq \lceil b \cdot h \rfloor \leq 1200$, where $t = 0.01$ and $h = 1/2t = 50$. So there are fifty-one candidates for the derived key (i.e., from $H(F_v(1150))$ to $H(F_v(1200))$). Accordingly, there are fifty-one candidates for b. In the case where $\lceil b \cdot h \rfloor = 1181$, we can determine the range of b, which satisfies $\lceil b \cdot h \rfloor = 1181$, as follows:

$$1180.5 \leq b \cdot 50 < 1181.5 \Longrightarrow 23.61 \leq b < 23.63. \tag{19}$$

Since this range (i.e., $23.61 \leq b < 23.63$) means the range of $[b - t, b + t)$ for the threshold value $t = 0.01$, an adversary may infer that b is 23.62. In the same way, an adversary may compute fifty-one candidates for b. The detailed description is given in Table 3.

Given that the integer part of the biometric value b is equal to 23 and the threshold value t is equal to 0.01 (i.e., side information denoted by Z), the probability that an adversary will succeed in correctly guessing a particular piece of a biometric vector is given as follows:

$$\Pr[guess \mid Z] = \frac{1}{51} \approx 1.96\%. \tag{20}$$

Compared to the case where no leakage of a biometric vector occurs, the probability that an adversary will succeed is increased. In other words, it will be easier for an adversary (i.e., imposter) to guess the biometric vector of a genuine user when integer leakage occurs. In order to make the probability lower, we can decrease a threshold value t, since the probability that an adversary will succeed in guessing a valid piece of biometric vector is $1/(1/2t + 1)$. For example, in the above case, if we lower t from 0.01 to 0.001, the probability that an adversary will succeed in guessing is reduced from 1.96% to about 0.2%.

Table 3: Candidates for a biometric value (in the case of leakage of the integer part).

$h = 50$		
$\lceil b \cdot h \rfloor = 1150$	$23.000 \leq b < 23.010$	$b \approx 23.005$
$\lceil b \cdot h \rfloor = 1151$	$23.010 \leq b < 23.030$	$b \approx 23.020$
⋮	⋮	⋮
$\lceil b \cdot h \rfloor = 1199$	$23.970 \leq b < 23.990$	$b \approx 23.980$
$\lceil b \cdot h \rfloor = 1200$	$23.990 \leq b < 24.000$	$b \approx 23.995$

5.2.3. Analysis of the Generalized Case. We now generalize the assumption by extending the length of a biometric vector to $n(> 1)$. Let p be the probability of success in the case where the length of a biometric vector is equal to 1 (e.g., p=0.0002 in Section 5.2.1). Ideally, the probability that an adversary will succeed in guessing the biometric vector of size n is p^n.

However, since components of the biometric vector generally correlate with each other, entropy loss will occur, making the probability of success higher than p^n. Let e_i be a factor of entropy loss of the i-th component of the biometric vector. We can write the probability that an adversary will succeed in guessing the biometric vector of size n as follows.

$$\prod_{i=1}^{n} \left(p \cdot \frac{1}{e_i} \right) \quad where\ e_i \in [0, 1)\ \ \forall i \in [1, n] \tag{21}$$

If n is sufficiently large, we can expect that the probability of leakage of a user's biometric vector is almost negligible.

6. Application

6.1. Replacement of PB-KDF. Password-based key derivation functions (PB-KDFs) are widely used in a variety of applications [27–31]. These can all be replaced with the proposed BB-KDF. In fact, only the key generation part would need to be replaced, from the password-based to the biometric-based mechanism, and the remaining process would not change.

Since the BB-KDF works with noisy (biometric) data, it can also works well with a combination of noisy and deterministic data. That is, in the BB-KDF, a user is able to generate a cryptographic key using his/her biometric vector and a human memorable password together. Since users generally choose passwords that can easily be memorized, the entropy of a password is more likely to be low, and, therefore, a PB-KDF is vulnerable to offline password guessing attacks.

Using the BB-KDF based on two factors, i.e., biometrics and password, we can enhance the PB-KDF based on only a single factor, i.e., password in both efficiency and security.

6.2. Authentication System. In this section, we construct a biometric authentication system by using the BB-KDF. A secure and efficient authentication is required to support a trusted communication in various environments such as mobile cloud computing [52], IoT [53], and wireless sensor networks [54]. The proposed authentication system is highly feasible in these environments since it is not only comparatively efficient, but also applicable to low-performance devices which do not provide secure storage (such as TPM) for cryptographic keys.

Although the output of the BB-KDF is uniformly distributed, and, therefore, it is potentially applicable as a secret key of identification or a signature scheme; however, simple application of BB-KDF does not work. To clarify the problem, we consider a combination of the Schnorr identification scheme, which is proven to be secure, and our BB-KDF for a biometric authentication scheme.

The secret key space of the Schnorr identification scheme is $\mathbb{Z}_p$, and the output space of the BB-KDF is a range of the underlying hash function. At the 112-bit security level, p is 2048-bit and the output of hash function is 256-bit, and this gap makes it inappropriate to apply the output of the BB-KDF to the secret key of the Schnorr identification scheme. However, we can solve this problem by using a well-known key extension technique [55]. Let the output of the BB-KDF be a PRF key K. Note that the PRF is taken as HMAC-SHA-256. The following function $Ext(K)$ produces byte strings of (arbitrary) length that is a positive multiple of n and the output length of the underlying PRF F_K.

$$\text{ext}(K) = F_K(0) \parallel F_K(1, A(1)) \parallel F_K(2, A(2)) \parallel \cdots \quad (22)$$

where $A(i) = F_K(i-1, A(i-1))$ for $i = 1, 2, \ldots$ and $A(0) = 0$. By using the above function, we can adjust a size of the key derived from the BB-KDF to 2048-bit.

A biometric authentication system consists of two phases: an enrollment phase and an authentication phase. In the enrollment phase, the server sets up an authentication system. A user then generates a public-key that will be used in the authentication phase by using his/her biometric vector and registers it with an authentication server using his/her identity and public-key. In the authentication phase, a user generates a secret key by using his/her biometric vector and authenticates himself/herself as an authorized user (i.e., registered in advance in the enrollment phase) to the server.

(i) Enrollment Phase

(1) Server: on input of a security level, the server generates a public parameter, denoted by PP, according to system policies.

(i) Compute **BB-KDF.Setup**$(1^k) \longrightarrow \Omega$
(Ω consists of (t, F_v, H) where $t \in [0,1)^n$ is a threshold vector, F_v is a PRF, where its key is embedded, and H is a hash function used in BB-KDF)

(ii) Choose a generator g of a group $\mathbb{G}$ of order p

(iii) Let the public parameter $PP = (g, p, \Omega)$

(2) User: the user recognizes his/her biometric vector and extracts features from it. We denote the extracted vector by the biometric vector $b = (b_1, \ldots, b_n) \in \mathbb{R}^n$. On input of a biometric vector b and a public parameter Ω in PP, the user (with identity ID) derives a key, denoted by K_b, and generates a public verification key, denoted by PK_{ID}, for authentication.

(i) Compute **BB-KDF.KDF**$(\Omega, b) \longrightarrow K_b$

(ii) Compute g^{K_b} and set $PK_{ID} = (g^{K_b})$

The user transmits a public verification key PK_{ID}, along with his/her identity ID, to the server.

(3) Server: the authentication server stores a pair (ID, PK_{ID}), and completes the registration of the user.

(ii) Authentication Phase

(1) User: the user chooses a random r_u in $\mathbb{Z}_p$ and computes $R = g^{r_u}$. The user starts an authentication process by transmitting his/her identity, ID, and a user-chosen random value, R, to the server.

(2) Server: the server chooses a random value $r_s \in \mathbb{Z}_p$, and transmits it as a challenge to the user.

(3) User: the user recognizes his/her biometric vector and extracts features from it. We denote the extracted vector by the biometric vector $b' = (b'_1, \ldots, b'_n) \in \mathbb{R}^n$. On input of biometric vector b' and a public parameter Ω in PP, the user (with identity ID) derives a key, denoted by $K_{b'}$, and generates a response s with the derived key $K_{b'}$ and the random values r_u, r_s.

(i) Compute **BB-KDF.KDF**$(\Omega, b') \longrightarrow K_{b'}$

(ii) $s \coloneqq r_s K_{b'} + r_u \bmod p$

The user transmits a response s to the server.

(4) Server: the server imports the stored public verification key, PK_{ID}, corresponding to the user's identity ID. The server then verifies the response s (transmitted from the user) by using a challenge r_s, a user-chosen random value R, and a public verification key PK_{ID}.

(i) Import $PK_{ID} = g^{K_b} = h$, the public verification key corresponding to user's ID, stored in DB

(ii) Check whether $g^s \cdot h^{-r_s} \stackrel{?}{=} R$

If the verification succeeds, the server authenticates the user and completes the authentication process.

Computational Cost. In the enrollment phase, the user cost is $(C_{\mathsf{KDF}} + \mathsf{Exp})$ where C_{KDF} is the cost of running **BB-KDF.KDF** and Exp is the cost of exponentiation in $\mathbb{G}$. In the authentication phase, the user cost is $(\mathsf{Exp}+C_{\mathsf{KDF}}+\mathsf{Mul})$ where Mul is the cost of multiplication in $\mathbb{Z}_q$. Since Exp or Mul is not such a big burden to users (even on IoT devices), the user-side cost depends mainly on C_{KDF}. In Section 7, we demonstrate

Table 4: Computational costs of authentication system.

	Clock speed	Enrollment time(ms)	Authentication time(ms)
MCU 1	150MHz	0.5501	0.5504
MCU 2	300MHz	0.2748	0.2749
Smartphone	2500MHz	0.0138	0.0138
Desktop	3500MHz	0.0067	0.0067

that C_{KDF} is reasonably low. Table 4 describes computational costs of the proposed authentication system (in case of $n = 100$) on the user side. The specifications of device are given in Table 5.

Theorem 6. *Let the discrete logarithm in the above authentication scheme be derived from the secure BB-KDF. If the discrete-logarithm problem is hard in $\mathbb{G}$, then the authentication scheme above is secure.*

Proof. The authentication scheme above is constructed on the basis of the Schnorr identification scheme, which is based on hardness of the discrete-logarithm problem (DLP). In the Schnorr identification scheme, a secret key is chosen uniformly at random. Since we use the *secure* BB-KDF to derive a secret key in the above authentication scheme, the derived key is indistinguishable from a random string of the same length. Therefore, on the assumption that the DLP is hard in $\mathbb{G}$, the authentication scheme above is secure. For more details of the proof, see Theorem 12.11 in [56]. □

7. Experimental Results

In this section, we analyze the performance of the BB-KDF constructed in Section 4 under various conditions, that is, considering both device specifications and the length of biometric vector. We also compare the BB-KDF with a PB-KDF in terms of efficiency.

Note that we need to compute the inverse of each threshold value in our BB-KDF. Due to its slow computation time, we assume that each inverse element is included in a public parameter for efficient evaluation. Since the threshold vector is public, it is reasonable to assume that the server computes the inverse of each threshold value and releases it as a public parameter.

Algorithm 1 is a pseudocode for the **BB-KDF.KDF** algorithm. In the experiment, HMAC-SHA-256 and SHA-256 are used as the underlying PRF F_v and hash algorithm H, respectively.

7.1. Effects of Device Specification. In this subsection we analyze the computational overhead of the BB-KDF based on various device specifications. Table 5 specifies the devices evaluated and the software used for the evaluation.

Given a biometric vector and a public parameter, the computational overhead for running the BB-KDF on the MCU1, MUC2, and smartphone (listed in Table 5) are described in Figure 3(a). In the experiment, HMAC-SHA-256 and SHA-256 are used as the underlying PRF and hash algorithm, respectively. The size of the PRF key and the length of biometric vector are set to be 32 bytes and 100, respectively. In this experiment, the key generation time for the BB-KDF is measured in three phases. The first phase, denoted by *kGen*, measures computing time of the rounding function; namely, $f_{h_i}(b_i) = \lceil b_i \cdot h_i \rfloor$ for all $i \in [1, 100]$. The second phase, denoted by *PRF*, measures computing time of the PRF; namely, $F_v(f_{h_1}(b_1) \parallel \ldots \parallel f_{h_n}(b_n))$. The third phase, denoted by *hash*, measures computing time of the hash function, namely, $H(\mathcal{X})$, where $\mathcal{X} = F_v(f_{h_1}(b_1) \parallel \ldots \parallel f_{h_n}(b_n))$. *Total* indicates the sum of *kGen*, *PRF* and *hash*.

Input: public parameter $(t = (t[0], ..., t[n-1]), F_v, H)$ and biometric vector $b = (b[0], b[1], ..., b[n-1])$ of length n
Output: key K
(1) initialize array $z = (z[0], z[1], ..., z[n-1])$;
(2) **for** $i = 0$ *to* $(n-1)$ **do**
(3) $z[i] \longleftarrow (\mathsf{int})(b[i] * (0.5/t[i]) + 0.5)$;
(4) **end**
(5) $X \longleftarrow F_v(z[0] \parallel \ldots \parallel z[n-1])$;
(6) **return** $K \longleftarrow H(X)$

Algorithm 1: BB-KDF.KDF.

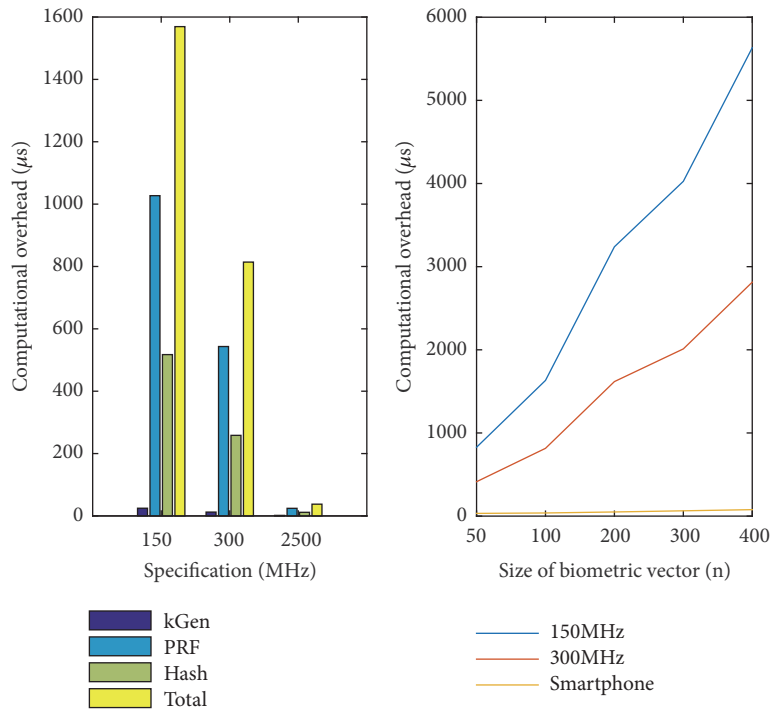
As illustrated in Figure 3(a), the better the performance of the device, the less computational overhead it takes to derive a key. *Total* computational overhead using the MCU1, MCU2, and smartphone is 1569.25μs, 814.09μs, and 37.7μs, respectively. From the result, we can show that the second phase takes much more computational overhead than the first phase does.

7.2. Effects of the Length of Biometric Vector. In this subsection, we analyze the computational overhead of the BB-KDF based on the length of the biometric vector, denoted by n. In this experiment, we consider five cases where n is equal to 50, 100, 200, 300, or 400. Since several biometrics can be combined to build a single biometric vector [1–4], we set the length of the biometric vector to be sufficiently large. Figure 3(b) illustrates computational overheads of the BB-KDF corresponding to vector size. For example, when $n = 200$, computational overheads of MCU1, MCU2, and smartphone are 3239.09μs,1617.85μs, and 49.87μs, respectively, making them deployable. From the result, we can show that the computational overhead increases linearly as the length of biometric vector gets bigger.

7.3. Comparison with PB-KDF. Computation time of the BB-KDF and PB-KDF on both smartphone and desktop is compared in Table 6. In the experiment, the PBKDF1 in PKCS♯5 is employed and the iteration number is set to be 1000 [26]. Only one PRF and one hash function are employed in the BB-KDF, resulting in a much faster key generation with the BB-KDF than with the PB-KDF. Feature extraction time varies according to extraction algorithms and types of biometric data [57, 58]. If we use a suitable algorithm for feature extraction, such as [16, 59], the BB-KDF can be an

Table 5: Specifications of device and software.

	Model	Specification
MCU 1	T1 TMS320F28335	Clock speed: 150MHz
MCU 2	T1 TMS320C28346	Clock speed: 300MHz
Smartphone	Samsung Galaxy S5	Clock speed: 2500MHz
Desktop	Intel Core i7 4770k	Clock speed: 3500MHz
SW 1	Code Composer V3.3	For T1 MCU: Compiler
SW 2	Eclipse + Android SDK	For Smartphone: Interpreter
SW 3	Eclipse + JDK 1.7	For Desktop: Interpreter



(a) Effects of device specification (b) Effects of the size of biometric vector

Figure 3: Computational overhead of the BB-KDF.

Table 6: Comparison with PB-KDF.

	BB-KDF	PB-KDF
Smartphone	37.69μs	5500μs
Desktop	18.14μs	500μs

efficient substitute for a PB-KDF across a wide range of devices.

8. Conclusions

In this paper, we have proposed the construction of a novel biometric-based key derivation function, called BB-KDF, and have proven its security. The proposed BB-KDF is computationally efficient, as shown in our experiments. As a result, it can be deployed in a wide range of devices for the Internet of Things (IoT) and can replace existing PB-KDFs in various environments. We have also introduced an authentication protocol using BB-KDF as an application.

For future work, it will be interesting to study the construction of robust BB-KDFs against outliers. Here the outliers imply some biased biometric points that are further away from what is deemed reasonable. Since biometric readings may differ from environment to environment, or sensor to sensor, outliers may exist with high probability.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00097, Development of Biometrics-Based Key Infrastructure Technology for Online Identification and No. 2018-0-01369, Developing Blockchain Identity Management System with Implicit Augmented Athentication and Privacy Protection for O2O Services.)

References

[1] A. Nagar, K. Nandakumar, and A. K. Jain, "Multibiometric cryptosystems based on feature-level fusion," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 255–268, 2012.

[2] S. Shekhar, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Joint sparse representation for robust multimodal biometrics recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 113–126, 2014.

[3] H. M. Sim, H. Asmuni, R. Hassan, and R. M. Othman, "Multimodal biometrics: Weighted score level fusion based on non-ideal iris and face images," *Expert Systems with Applications*, vol. 41, no. 11, pp. 5390–5404, 2014.

[4] C. Li, J. Hu, J. Pieprzyk, and W. Susilo, "A New Biocryptosystem-Oriented Security Analysis Framework and Implementation of Multibiometric Cryptosystems Based on Decision Level Fusion," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1193–1206, 2015.

[5] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Biometrics break-ins and band-aids," *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2105–2113, 2003.

[6] A. Ross, J. Shah, and A. K. Jain, "From template to image: Reconstructing fingerprints from minutiae points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 544–560, 2007.

[7] J. Galbally, A. Ross, M. Gomez-Barrero, J. Fierrez, and J. Ortega-Garcia, "Iris image reconstruction from binary templates: An efficient probabilistic approach based on genetic algorithms," *Computer Vision and Image Understanding*, vol. 117, no. 10, pp. 1512–1525, 2013.

[8] M. Savvides, B. Vijaya Kumar, and P. Khosla, "Cancelable biometric filters for face recognition," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, pp. 922–925 Vol.3, Cambridge, UK, August 2004.

[9] A. B. J. Teoh and D. C. L. Ngo, "Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 1892–1901, 2006.

[10] J. K. Pillai, V. M. Patel, R. Chellappa, and N. K. Ratha, "Sectored random projections for cancelable iris biometrics," in *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2010*, pp. 1838–1841, USA, March 2010.

[11] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 1999 6th ACM Conference on Computer and Communications Security (ACM CCS)*, pp. 28–36, November 1999.

[12] A. Juels and M. Sudan, "A fuzzy vault scheme," in *Proceedings of the IEEE International Symposium on Information Theory*, p. 408, Lausanne, Switzerland.

[13] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: how to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology—EUROCRYPT 2004*, pp. 523–540, Springer, Berlin, Germany, 2004.

[14] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain, "Biometric cryptosystems: issues and challenges," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 948–959, 2004.

[15] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith, "Reusable fuzzy extractors for low-entropy distributions," in *Proceedings of the EUROCRYPT, Part I*, pp. 117–146, 2016.

[16] Y. Sutcu, Q. Li, and N. Memon, "Protecting biometric templates with sketch: theory and practice," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 503–512, 2007.

[17] F. Guo, W. Susilo, and Y. Mu, "Distance-Based Encryption: How to Embed Fuzziness in Biometric-Based Encryption," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 247–257, 2016.

[18] B. Barak, R. Shaltiel, and E. Tromer, "True random number generators secure in a changing environment," in *Proceedings of the CHES*, pp. 166–180, 2003.

[19] B. Barak and S. Halevi, "A model and architecture for pseudo-random generation and applications to /dev/random," in *Proceedings of the CCS 2005 - 12th ACM Conference on Computer and Communications Security*, pp. 203–212, November 2005.

[20] R. Gennaro, H. Krawczyk, and T. Rabin, "Secure hashed diffie-hellman over non-ddh groups," in *Proceedings of the EUROCRYPT*, pp. 361–381, 2004.

[21] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Proceedings of the CRYPTO*, pp. 631–648, 2010.

[22] Y. Dodis, R. Gennaro, J. HÅtad, H. Krawczyk, and T. Rabin, "Randomness extraction and key derivation using the cbc, cascade and hmac modes," in *Proceedings of the CRYPTO*, pp. 494–510, 2004.

[23] L. Chen, "Recommendation for key derivation using pseudo-random functions (revised)," National Institute of Standards and Technology NIST SP 800-108, 2009.

[24] C. W. Chuah, E. Dawson, and L. Simpson, "Key derivation function: The SCKDF scheme," *IFIP Advances in Information and Communication Technology*, vol. 405, pp. 125–138, 2013.

[25] L. Chen, "Recommendation for key derivation through extraction-then-expansion," National Institute of Standards and Technology NIST SP 800-56c, 2011.

[26] RSA laboratories. PKCS#5 v2.1: Password based cryptography standard, 2012.

[27] "Android 5.0 full disk encryption," https://source.android.com/security/encryption/.

[28] "IEEE 802.11 WG:Part 11. wireless lan medium access control (mac) and physical layer (phy) specifications," IEEE Std 802.11 i-2004, 2004.

[29] "Best practices for deploying filevault 2. technical report," 2012, http://training.apple.com/pdf/WP_FileVault2.pdf.

[30] O. Choudary, F. Grobert, and J. Metz, "Infiltrate the vault: Security analysis and decryption of lion full disk encryption," Cryptology ePrint Archive 2012/374, 2012.

[31] "RAR archive format," http://www.rarlab.com/technote.htm.

[32] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, "Secure remote authentication using biometric data," in *Proceedings of the EUROCRYPT*, pp. 147–163, 2005.

[33] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs, "Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors," in *Proceedings of the EUROCRYPT*, pp. 471–488, 2008.

[34] Y. Dodis, B. Kanukurthi, J. Katz, and A. Smith, "Robust fuzzy extractors and authenticated key agreement from close secrets," *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 6207–6222, 2012.

[35] B. Fuller, X. Meng, and L. Reyzin, "Computational fuzzy extractors," in *Proceedings of the ASIACRYPT, Part I*, pp. 174–193, 2013.

[36] X. Boyen, "Reusable cryptographic fuzzy extractors," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*, pp. 82–91, October 2004.

[37] D. Apon, C. Cho, K. Eldefrawy, and J. Katz, "Efficient, reusable fuzzy extractors from lwe," in *Proceedings of the International Conference on Cyber Security Cryptography and Machine Learning*, pp. 1–18, Springer, 2017.

[38] Y. Wen, S. Liu, and S. Han, "Reusable fuzzy extractor from the decisional Diffie–Hellman assumption," *Designs, Codes and Cryptography. An International Journal*, vol. 86, no. 11, pp. 2495–2512, 2018.

[39] K. Takahashi, T. Matsuda, T. Murakami, G. Hanaoka, and M. Nishigaki, "A signature scheme with a fuzzy private key," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 105–126, Springer, 2015.

[40] T. Matsuda, K. Takahashi, T. Murakami, and G. Hanaoka, "Fuzzy signatures: relaxing requirements and a new construction," in *International Conference on Applied Cryptography and Network Security*, pp. 97–116, Springer, 2016.

[41] M. Yasuda, T. Shimoyama, M. Takenaka, N. Abe, S. Yamada, and J. Yamaguchi, "Recovering attacks against linear sketch in fuzzy signature schemes of acns 2015 and 2016," in *Proceedings of the International Conference on Information Security Practice and Experience*, pp. 409–421, Springer, 2017.

[42] J. HÅstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1364–1396, 1999.

[43] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct randolli functions," in *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, pp. 464–479, 1984.

[44] "Fingerprint verification competition (fvc2002)," 2002, http://bias.csr.unibo.it/fvc2002.

[45] R. Cappelli, M. Ferrara, and D. Maltoni, "Minutia Cylinder-Code: A new representation and matching technique for fingerprint recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2128–2141, 2010.

[46] S. Hohenberger and B. Waters, "Short and stateless signatures from the rsa assumption," in *Proceedings of the CRYPTO*, pp. 654–670, 2009.

[47] S. Hohenberger and B. Waters, "Realizing hash-and-sign signatures under standard assumptions," in *Proceedings of the EUROCRYPT*, pp. 333–350, 2009.

[48] D. Hofheinz, T. Jager, and E. Kiltz, "Short signatures from weaker assumptions," in *Proceedings of the ASIACRYPT*, pp. 647–666, 2011.

[49] F. Böhl, D. Hofheinz, T. Jager, J. Koch, J. H. Seo, and C. Striecks, "Practical signatures from standard assumptions," in *Proceedings of the EUROCRYPT*, pp. 461–485, 2013.

[50] Z. Jin, M.-H. Lim, A. B. J. Teoh, B.-M. Goi, and Y. H. Tay, "Generating Fixed-Length Representation from Minutiae Using Kernel Methods for Fingerprint Authentication," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 10, pp. 1415–1428, 2016.

[51] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[52] Z. Han, L. Yang, S. Wang, S. Mu, and Q. Liu, "Efficient Multi-factor Two-Server Authenticated Scheme under Mobile Cloud Computing," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–14, 2018.

[53] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication Protocols for Internet of Things: A Comprehensive Survey," *Security and Communication Networks*, vol. 2017, 2017.

[54] R. Riaz, T.-S. Chung, S. S. Rizvi, and N. Yaqub, "BAS: The Biphase Authentication Scheme for Wireless Sensor Networks," *Security and Communication Networks*, vol. 2017, 2017.

[55] M. Fischlin, A. Lehmann, and D. Wagner, "Hash function combiners in tls and ssl," in *Proceedings of the CT-RSA 2010*, pp. 268–283, 2010.

[56] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2nd edition, 2014.

[57] S. A. M. C. Vigila, K. Muneeswaran, and W. T. B. A. Antony, "Biometric security system over finite field for mobile applications," *IET Information Security*, vol. 9, no. 2, pp. 119–126, 2015.

[58] L. Bondi, L. Baroffio, M. Cesana, M. Tagliasacchi, G. Chiachia, and A. Rocha, "Rate-energy-accuracy optimization of convolutional architectures for face recognition," *Journal of Visual Communication and Image Representation*, vol. 36, pp. 142–148, 2016.

[59] Y. Xu, D. Zhang, and J.-Y. Yang, "A feature extraction method for use with bimodal biometrics," *Pattern Recognition*, vol. 43, no. 3, pp. 1106–1115, 2010.