Received April 6, 2019, accepted May 14, 2019, date of publication May 27, 2019, date of current version June 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919351

Fuzzy Vector Signature and Its Application to Privacy-Preserving Authentication

MINHYE SEO[1], JUNG YEON HWANG[2], DONG HOON LEE[1], (Member, IEEE), SOOHYUNG KIM[2], SEUNG-HYUN KIM[2], AND JONG HWAN PARK[3], (Member, IEEE)

[1]Center for Information Security Technologies, Korea University, Seoul 02841, South Korea
[2]Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea
[3]Department of Computer Science, Sangmyung University, Seoul 03016, South Korea

Corresponding author: Jong Hwan Park (jhpark@smu.ac.kr)

This work was supported in part by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea Government (MSIT) (Developing Blockchain Identity Management System with Implicit Augmented Authentication and Privacy Protection for O2O Services) under Grant 2018-0-01369.

ABSTRACT Fuzzy authentication uses non-deterministic or noisy data, like biometrics, as an authentication factor. Although the data is extracted from the same individual or source, it can be different for each measurement. As a result, one of the main issues in fuzzy authentication is the effective processing of the *fuzziness*, while guaranteeing the *privacy* of the fuzzy data. Biometric data is a typical user-generated fuzzy data and the fuzzy extractor is one of the most promising primitives for biometric authentication these days. In 2016, Canetti *et al.* proposed the *reusable* fuzzy extractor, in which multiple keys can be generated with the same biometric data. It can also handle some *outliers* which occur unexpectedly (owing to an external interference when acquiring the fuzzy data, for example, the presence of dust on a fingerprint image). However, the size of the user's helper data in the reusable fuzzy extractor is quite large. This makes the network bandwidth usage required in the *online* authentication phase (or the storage required on the user side) considerable, which inconveniences the user. In this paper, we present a new primitive for fuzzy authentication, called a fuzzy vector signature (FVS) scheme, which significantly alleviates the burden on the user side. This means that the network bandwidth usage (or the amount of storage required on the user side) is significantly reduced. The proposed FVS scheme is *reusable* and robust to *outliers* as well. Finally, we provide a privacy-preserving fuzzy authentication protocol based on the FVS scheme.

INDEX TERMS Biometric authentication, fuzzy vector signature, outlier, privacy, reusability.

I. INTRODUCTION

User authentication is an essential element that must be preceded for secure communication over a network. In the transport layer security (TLS) protocol, the public key infrastructure (PKI) enables authentication between users and secure session key sharing. Each user generates a public key and a private key pair, and to provide the robustness of the authentication system, the user must securely store his/her private key. Moreover, this private key should be a bit string of sufficient length chosen uniformly at random. However, it is challenging for a user to memorize such a long and random private key. Therefore, the user is required to use additional means to store and protect it. The user can activate a private key using a password that is easy to memorize, or carry a secure hardware device such as a security token or smart card that can store information securely. Either way, it may inconvenience the user to memorize or carry something. Occasionally, the loss of the security token or exposure of the password can fundamentally compromise the security of the private key.

Recently, tremendous research has been conducted to authenticate a user using his/her characteristics, i.e., features to characterize the user. This could alleviate the above inconveniences and significantly improve usability, since it relies solely on the user's unique features (i.e., who you *are*). The most popular features of the user are biometrics, such as fingerprint, iris, keystroke dynamics, and gait. Biometric information is unique to the individual, and therefore, is suitable for use in authentication of a user.

The associate editor coordinating the review of this manuscript and approving it for publication was Kaiping Xue.

The state–of–the–art technology of biometric-based authentication is a fuzzy extractor. Recently, a *reusable* fuzzy extractor, which better reflects real-life applications, has been proposed [1]. However, in many ICT applications, user devices are resource-constrained in terms of memory and computing power to deploy reusable fuzzy extractors. In (reusable) fuzzy extractors, a public *helper data* is essential in that it enables two similar biometric readings to generate the same cryptographic key, but its size is large enough for user devices, such as biometric sensors in access control systems, wearable devices in IoT environments, and smart cards in Fintech services, to handle.

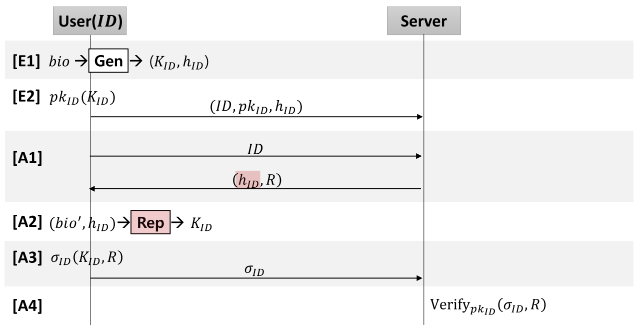
A. RESUSABLE FUZZY EXTRACTORS

A so-called fuzzy extractor [2] is considered as one of the representative primitives for biometric-based authentication. It generates a private key using a user's noisy biometric data with the aid of public helper data. The key can be used for cryptographic functions such as encrypting and signing a message. More precisely, the fuzzy extractor consists of two algorithms, Gen and Rep. The Gen algorithm takes the user's fuzzy data as input and outputs a key along with helper data. The Rep algorithm takes the user's fresh fuzzy data and the helper data as input and outputs a key. If the two pieces of user's fuzzy data are sufficiently similar in some underlying metric space, the Rep algorithm extracts the same key from the helper data.

Since the initial construction [2], a fuzzy extractor has been enhanced in terms of security and usability. A 'robust' fuzzy extractor has been studied to protect against malicious alteration of a helper data [3]. The notion of robustness is important in secure remote authentication using biometric data. A user needs to receive helper data for each authentication attempt from a server over a public channel, that is, in the presence of an active adversary who may modify arbitrary messages sent between the server and the user. Recently, Canetti *et al.* proposed a 'reusable' fuzzy extractor [1], in which multiple keys can be generated with the same biometric data and registered for numerous independent application servers. It can support all biometric sources of a sufficiently high minentropy and also many sources with an entropy rate much lower than the error rate. In addition, it can be easily made robust by the random-oracle-based transform of [3].

However, a fuzzy extractor is still limited because huge helper data is required for the user. This implies that the network bandwidth usage in the online authentication phase or the storage on the user side would be considerable (see Table 2).

Figure 1 shows a typical biometric-based authentication protocol using the reusable fuzzy extractor [1] and a standard public-key signature scheme. It works in a challenge-response manner, where R in Figure 1 stands for a random challenge. In the protocol, the key K_{ID} generated by the fuzzy extractor is assumed to be used as a secret signing key. In the



User(ID)
Server
[E1] $bio \rightarrow$ Gen $\rightarrow (K_{ID}, h_{ID})$
[E2] $pk_{ID}(K_{ID})$
(ID, pk_{ID}, h_{ID})
[A1]
ID
(h_{ID}, R)
[A2] $(bio', h_{ID}) \rightarrow$ Rep $\rightarrow K_{ID}$
[A3] $\sigma_{ID}(K_{ID}, R)$
σ_{ID}
[A4]
$\mathsf{Verify}_{pk_{ID}}(\sigma_{ID}, R)$

FIGURE 1. Authentication with the fuzzy extractor.

enrollment phase, the user $\mathcal{U}$ and the server $\mathcal{S}$ proceed as follows:

[E1] Given biometric data, *bio*, Gen is run to generate a secret key, K_{ID} and a helper data h_{ID};

[E2] $\mathcal{U}$ generates a public verification key, pk_{ID} by using K_{ID} and transmits a tuple (ID, pk_{ID}, h_{ID}) to $\mathcal{S}$;

In the authentication phase, $\mathcal{U}$ and $\mathcal{S}$ interact as follows:

[A1] $\mathcal{U}$ gives his/her identity ID to $\mathcal{S}$ and takes a tuple (h_{ID}, R) as a response;

[A2] $\mathcal{U}$ runs Rep to generate a secret key K_{ID} by using his/her biometric data bio' and h_{ID}. Note that Gen and Rep generate the same keys K_{ID} with the aid of h_{ID} if bio and bio' are within a threshold bound;

[A3] $\mathcal{U}$ generates a signature σ_{ID} of R by using K_{ID} and gives it to $\mathcal{S}$;

[A4] $\mathcal{S}$ verifies σ_{ID} by using pk_{ID} and R. If the verification succeeds, $\mathcal{S}$ authenticates $\mathcal{U}$ and completes the authentication process.

In the reusable fuzzy extractor, Rep runs d key reconstruction subroutines internally by decrypting digital lockers, i.e., ciphertexts that hide K_{ID}. Consequently, the helper data, h_{ID} must include at least d digital lockers. In other words, the size of the helper data, h_{ID} increases proportionally to d.

In step **[A1]**, the user needs to receive h_{ID} from the server in each authentication session. An increase in the size of helper data leads to an increase in network bandwidth usage in the online authentication phase. In addition, in step **[A2]**, the user runs Rep to generate the secret key K_{ID}, where d key reconstruction subroutines are repeated. This also would consume a considerable amount of computation on the user side.

B. OUR CONTRIBUTIONS

In this paper, we propose a new primitive for biometric-based authentication, called *fuzzy vector signature (FVS)*, which significantly alleviates the burden on the user side. In our FVS scheme, the user's fuzzy data such as biometric data can be used as a signing key. A signer is not required to memorize or carry anything on a regular basis, and only his/her fuzzy data is considered as secret information. A signature is generated directly from the measurement of the user's

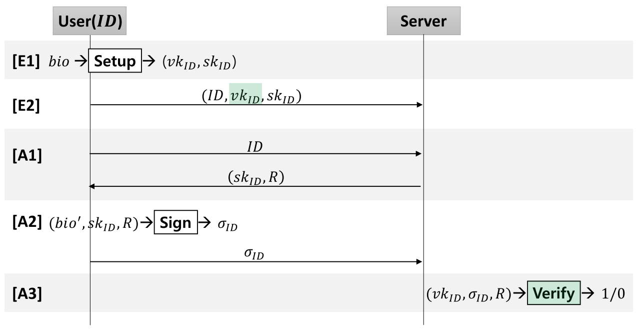

User(ID)
Server
[E1] bio → Setup → (vk_{ID}, sk_{ID})
[E2] (ID, vk_{ID}, sk_{ID})
[A1] ID
(sk_{ID}, R)
[A2] (bio', sk_{ID}, R) → Sign → σ_{ID}
σ_{ID}
[A3] $(vk_{ID}, \sigma_{ID}, R)$ → Verify → 1/0

FIGURE 2. Authentication with the fuzzy vector signature.

fuzzy data.[1] We formally prove that the proposed scheme achieves the anonymity of the signer and the unforgeability of the signature, as well as the privacy of the verification key. These properties enable our scheme to guarantee user privacy and be secure against impersonation.

Compared to the reusable fuzzy extractor [1], our scheme considerably improves efficiency on the user side. In the proposed FVS scheme, the user requires a signing key or parameter, which is public information similar to helper data, but its size is much smaller than helper data in [1]. This implies that the network bandwidth usage (or the amount of storage required on the user side) is significantly reduced. A detailed analysis is presented in Section V-C. Note that, since the application server has much better resources than the user's device, reduction of cost on the user side is a significant achievement.

Since we use *fuzzy* data as a private key, our scheme supports a threshold predicate, which means that a signer can control a threshold bound to validate a signature arbitrarily according to an acceptance policy. In addition, our construction exhibits tolerance to *outliers*. Even if the user's fuzzy data contains a few outliers, the user can generate a valid signature if it satisfies a threshold predicate. Furthermore, our FVS scheme is *reusable*, such that public verification keys can be generated multiple times with the same fuzzy data and registered for numerous independent application services.

To illustrate the high efficiency of our scheme, we present experimental results using bilinear map parameters. For a d224 curve parameter which can guarantee 1344-bit RSA security, a signature can be generated within a second on the signer's side.

1) COMPARISON WITH THE FUZZY EXTRACTOR

In order to show the advantages of our scheme, we consider an authentication protocol using our FVS in Figure 2. Our protocol is also designed based on a challenge-response mechanism, where R stands for a random challenge in Figure 2.

In Figure 2, a signing parameter sk_{ID} serves the same function as the helper data h_{ID} in Figure 1. That is, in the authentication phase, sk_{ID} is used to generate a signature, along with the user's biometric data. In a FVS, the signing parameter is public (unlike in a typical public-key signature), and only the user's biometric data is kept secret. In the enrollment phase, the user $\mathcal{U}$ and the server $\mathcal{S}$ interact as follows:

- **[E1]** $\mathcal{U}$ runs Setup algorithm to generate a signing key sk_{ID} and a verification key vk_{ID};
- **[E2]** $\mathcal{U}$ gives a tuple (ID, sk_{ID}, vk_{ID}) to $\mathcal{S}$;

In the authentication phase, $\mathcal{U}$ and $\mathcal{S}$ interact as follows:

- **[A1]** $\mathcal{U}$ gives his/her identity ID to $\mathcal{S}$ and takes a tuple (sk_{ID}, R) as a response;
- **[A2]** $\mathcal{U}$ runs Sign algorithm to generate a signature σ_{ID} of R by using his/her biometric data bio' and sk_{ID}, and gives it to $\mathcal{S}$;
- **[A3]** $\mathcal{S}$ runs Verify algorithm by using vk_{ID} and R. If the verification succeeds, $\mathcal{S}$ authenticates $\mathcal{U}$ and completes the authentication process.

In the fuzzy vector signature, d iterative subroutines are also deployed to deal with outliers. However, unlike in the reusable fuzzy extractor, these iterative subroutines are part of the Verify algorithm, which is run by the server. That is, the Verify algorithm internally runs the d iterative verification subroutines, and thus, the size of the verification key vk_{ID} increases proportionally to d. On the other hand, the size of the signing key sk_{ID} (transmitted to the user in the *online* authentication phase) and the computational cost of the Sign algorithm are independent of d. In the proposed FVS scheme, these are affordable at the user level, which mitigates inconveniences of the user when compared to the reusable fuzzy extractor.

There is, of course, a trade-off between the user and the server. In other words, the server is required to increase the amount of resources for authentication, while the burden on the user side is considerably alleviated. However, FVS serves the purpose of reducing network bandwidth usage and computational cost *on the user side*, which is more important when deployed in the real world.

In the proposed FVS scheme, the size of the user's helper data is significantly reduced (compared with that in [1]), which decreases the network bandwidth usage (or the amount of storage on the user side) required for authentication. For example, if the user input is a vector of length 128 and the error occurs by 15.6% between measurements, the user's helper data requires approximately 712,704 KB of storage in [1], whereas only 24 KB is required in our FVS scheme (for more details, see Section V-C).

2) TECHNICAL OVERVIEW

Intuitively, our FVS scheme is constructed by combining the so-called *hidden vector encryption* (HVE) [4]–[6] and one-time ElGamal-type signature using DL parameters.[2] A signature, $\sigma = (\sigma_1, \ldots, \sigma_n)$ of FVS is a kind of a ciphertext

[1]The fuzzy signature in [11], [12] is the closest notion to the newly proposed fuzzy vector signature (FVS), but there are differences in definition and security requirements.

[2]In the paper we present an example using a Schnorr signature scheme. However, the resulting scheme can be easily extended by replacing the Schnorr signature with any kind of ElGamal-type signature.

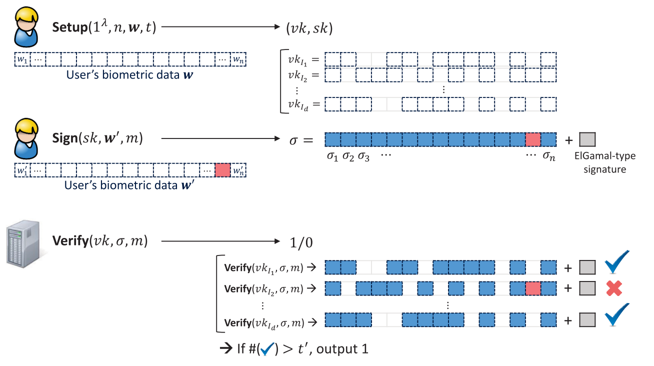

Setup$(1^\lambda, n, w, t)$
(vk, sk)
User's biometric data w
vk_{I_1} =
vk_{I_2} =
vk_{I_d} =
Sign(sk, w', m)
σ =
σ_1 σ_2 σ_3 ⋯ ⋯ σ_n
ElGamal-type signature
User's biometric data w'
Verify(vk, σ, m)
1/0
Verify(vk_{I_1}, σ, m)
Verify(vk_{I_2}, σ, m)
Verify(vk_{I_d}, σ, m)
If #(✓) > t', output 1

FIGURE 3. Overview.

of HVE, which is generated from a vector $w' = (w'_1, \ldots, w'_n)$. A one-time ElGamal-type signature is additionally included in the signature, to prevent a forgery by an attacker who is not aware of the legitimate biometric data. A verification key of FVS is defined by a set of tokens of HVE, which is generated from subvectors of an initial fuzzy vector, $w = (w_1, \ldots, w_n)$. In the Verify algorithm, the signature succeeds in verification if the two vectors w and w', which are used to generate the signature and the verification key, respectively, have a predefined number of common subvectors.

FVS is constructed with a subset-based design principle. More precisely, we first generate a set $I_j \subset \{1, \ldots, n\}$ of size $\ell (< n)$ uniformly at random for $j \in [1, d]$. And then, we set d subvectors of w', $\left(\{w'_i\}_{i \in I_1}, \ldots, \{w'_i\}_{i \in I_d}\right)$, and generate d verification keys, $\left(vk_{I_1}, \ldots, vk_{I_d}\right)$, from them. In the Verify algorithm of FVS scheme, d sub-signatures, $\left(\{\sigma_i\}_{i \in I_1}, \ldots, \{\sigma_i\}_{i \in I_d}\right)$, are generated and each sub-signature $\{\sigma_i\}_{i \in I_j}$ is verified with the verification key vk_{I_j}. If no error has occurred at all positions in I_j, that is, if $\{w_i\}_{i \in I_j}$ and $\{w'_i\}_{i \in I_j}$ are the same, the verification succeeds. A server or a verifier proceeds the verification for each pair $\left(\{\sigma_i\}_{i \in I_j}, vk_{I_j}\right)$. If the verification succeeds at least t' times in d times, the signature σ is regarded as a valid one.

C. ORGANIZATION

The remainder of this paper is organized as follows: In Section II, we review related work in the area. In Section III, we introduce the basic notations and the definition of cryptographic assumptions. In Section IV, we define the fuzzy vector signature (FVS) and its security model. We describe our proposed FVS scheme in Section V. In Section VI, we present a fuzzy authentication protocol as an application of the FVS scheme. Conclusions are given in Section VII.

II. RELATED WORK

In the early stages of research on biometric authentication, biometric templates were developed and used as such. That is, a template extracted from a user's biometrics would be stored intact on the server in the enrollment phase, and in the authentication phase, a newly extracted template would then be compared with the stored one. But biometric templates stored in the server (in facsimile) have resulted in a number of privacy issues. Once a biometric template is stored in the server or database, the raw biometric data can be recovered, compromising user privacy [7], [8]. Also, if a particular piece of biometric data has been compromised, it cannot be used again for authentication; however, as a wide array of biometrics is not available for authentication, it cannot continually be replaced.

TABLE 1. Comparison with biometric cryptosystems.

	No Decryption Key	Malicious Server	User Privacy
Reusable Fuzzy Extractor	O	O	O
Anonymous Fuzzy IBE	O	X	O
Fuzzy Signature	O	X	X
Homomorphic Encryption	X	O	O
Functional Encryption	O	X	O
Fuzzy Vector Signature	O	O	O

With the threat of compromise and limited biometric resources for use (for authentication), recent research has focused on how to authenticate users while protecting biometric templates, which is called a *biometric cryptosystem*. The main challenge in biometric cryptosystems is to address the *fuzziness* of the biometric data. As biometric data are noisy, biometric readings may differ from time to time, although they are derived from the same individual. Nevertheless, the authentication must be successful if the difference between two pieces of fuzzy biometric data is within a certain minimal threshold (i.e., two close versions of biometric data will be regarded as the same user's biometric data). Biometric cryptosystems are designed in different ways to deal with this problem:

- *Fuzzy extractor:* The fuzzy extractor, first introduced by Dodis *et al.* in 2004, generates a cryptographically secure key using the user's biometric data [2]. Although the fuzzy extractor is a useful cryptographic primitive, since it operates on the basis of error-correcting codes, there is a limit to the range of noise processing. What this indicates is that although only one outlier occurs in the measurement of biometric data, the Rep algorithm recognizes it as another user; thus, it cannot generate the same key.

  In 2016, Canetti *et al.* proposed a reusable fuzzy extractor, which can relieve the above difficulties [1]. That is, even if a few outliers are measured from the user's biometric reading, they may be permissible within a threshold range. But this scheme requires large helper data, which inconveniences the user. For example, if the user input is a vector of length 128, and the error occurs by 15.6% between measurements, the user's helper data requires approximately 696 MB of storage. If the probability of occurrence of an error is increased to 19.5%, the size of the helper data increases to approximately 34,600 MB, which is significantly large. The user can store his/her helper data in the server during enrollment phase and receive it when executing authentication online. However, even in this case, a tremendous network bandwidth is used online. In reality, when a large number of users participate in the system, such large bandwidth usage overloads the system traffic.

- *Fuzzy identity-based encryption:* In fuzzy identity-based encryption (IBE), an identity is represented as a set of attributes. The ciphertext with an identity w is decrypted with the secret key with an identity w' if w and w' overlap more than a threshold [9]. This error-tolerance property of fuzzy IBE allows it to be used for biometric authentication. An anonymous fuzzy IBE scheme [10] enables biometric data (i.e., identity) to be hidden from a ciphertext, which protects user privacy, but not from a secret key, which allows a malicious server to compromise the user's biometric data.
- *Fuzzy signature:* Fuzzy signature is a notion proposed by Takahashi *et al.* [11], which uses the user's fuzzy data (such as biometric data) as a signing key. Unlike the fuzzy extractor, this scheme does not require additional auxiliary information. Matsuda *et al.* have improved the fuzzy signature proposed in [11] by relaxing the requirements for construction and by increasing the efficiency [12]. However, the fuzzy signature is not robust when outliers are included in the user's fuzzy data. That is, a valid signature cannot be generated even if there is only one outlier in the user's fuzzy data when measured during the authentication phase. Moreover, the fuzzy signature scheme proposed in [11], [12] does not guarantee the privacy of the user. The user's biometric data can be directly recovered from the public verification key or signature, as shown in [13].
- *Homomorphic encryption:* Homomorphic encryption enables an evaluator to compute on encrypted data without decrypting, thereby preserving the confidentiality of the underlying data. This property has promoted a variety of research on biometric-based authentication using homomorphic encryption schemes [14]–[17]. However, with this type of encryption, the result of the computation is provided in an encrypted form, so a decryption key is essential at the end of authentication. Since the decryption key discloses biometric information from the ciphertext, the user must keep this securely. This means that the user must store additional secrets in addition to his/her biometric information. A method proposing trusted third party management of decryption keys alleviates this inconvenience to the user, but this is a strong assumption. In this paper, we seek a method that does not require any additional secrets other than biometric information for user authentication.
- *Functional encryption:* Functional encryption, like homomorphic encryption, can perform operations on encrypted data, but there is a big difference in that the result of the operation is provided in plaintext. There have been a number of attempts to apply efficient functional encryption schemes for inner products to biometric authentication since the inner product can induce several kinds of distance metrics [18]–[21]. However, the inner product has inherent vulnerability: if the functional key (stored in the server) is compromised, the corresponding biometric information can be easily revealed. Anyone (including the attacker) can generate a ciphertext for the data of his/her own choice,[3] and for example, if an attacker generates a ciphertext for the vector (1, 0, ..., 0) and decrypts it with a compromised key, then he/she can find out the first component of the legitimate user's biometric information (corresponding to the compromised key).

III. PRELIMINARIES

In this section, we briefly review the background knowledge for our construction and describe the complexity assumption required for our construction.

A. NOTATIONS

Let $\mathsf{poly}(\lambda)$ denote a polynomial in variable λ. We define that $\nu(\lambda)$ is a negligible function if $\nu(\lambda) < 1/\mathsf{poly}(\lambda)$ for any $\mathsf{poly}(\lambda)$ and sufficiently large λ. We denote the concatenation operation on strings by '$||$'. $|A|$ implies the cardinality of a set A, i.e., the number of elements of the set A. Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be vectors of length n. The hamming distance $d(x, y)$ is defined as the number of places at which x and y differ. That is, $d(x, y) = \big|\{i \in [1, n] \mid x_i \neq y_i\}\big| \in \{0, \ldots, n\}$.

B. BILINEAR MAPS

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p. We say that e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an admissible bilinear map (or a pairing) if the following properties are satisfied:

1) Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$.
2) Non-degeneracy: $e(g_1, g_2) \neq 1$.
3) Computability: There exists an efficient algorithm to compute $e(g_1^a, g_2^b)$ for all $g_1^a \in \mathbb{G}_1$ and $g_2^b \in \mathbb{G}_2$.

Bilinear maps can be classified in three types. Type I pairings have $\mathbb{G}_1 = \mathbb{G}_2$, called '*symmetric*'. Type II pairings have an efficiently computable isomorphism only in one direction, i.e., from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$ and none in the reverse direction. Type III pairings have no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Type II and III pairings are called '*asymmetric*'. For more details, refer to [22].

C. DISCRETE LOGARITHM ASSUMPTION

Let q be a prime such that $q|p - 1$ for a prime p with $2^{k-1} \leq p < 2^k$ (p is k bits long), and $\mathbb{G}_q$ be a subgroup of $\mathbb{Z}_p^*$ of order q. Let g be a generator of $\mathbb{G}_q$. For any polynomial-time algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$, denoted

[3] We rule out the symmetric-key setting since the user must have a secret key for encryption apart from his/her biometric information.

by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DLOG}}(\lambda)$, as follows:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DLOG}} = \Pr\left[a \leftarrow \mathcal{A}(\mathbb{G}_q, q, g, g^a); a \leftarrow_R \mathbb{Z}_q\right]$$

We say that the DLOG assumption holds if, for all probabilistic polynomial time (PPT) algorithms $\mathcal{A}$ and any security parameter λ, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DLOG}}(\lambda) < \epsilon(\lambda)$ for some negligible function ϵ.

D. DECISIONAL DIFFIE-HELLMAN ASSUMPTION

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups of order p; and g, h be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The security of our construction is proven based on the decisional Diffie-Hellman assumptions in groups $\mathbb{G}_1$ or $\mathbb{G}_2$, called DDH1 or DDH2, respectively. Assume that $a, b \in \mathbb{Z}_p^*$ and $T \in \mathbb{G}_1$ are selected randomly. To define *Decisional Diffie-Hellman 1 (*DDH1*) problem*, we consider the following two distributions:

- $\mathcal{D}_N := \left(g, g^a, g^b, h, g^{ab}\right) \in \mathbb{G}_1^3 \times \mathbb{G}_2 \times \mathbb{G}_1$
- $\mathcal{D}_R := \left(g, g^a, g^b, h, T\right) \in \mathbb{G}_1^3 \times \mathbb{G}_2 \times \mathbb{G}_1$

For any polynomial-time algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$, denoted by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH1}}(\lambda)$, in distinguishing these two distributions:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH1}}(\lambda) = \left|\Pr\left[\mathcal{A}(1^\lambda, \mathcal{N}) \to 1\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathcal{R}) \to 1\right]\right|$$

where $\mathcal{N}$ is sampled from $\mathcal{D}_N$ and $\mathcal{R}$ is sampled from $\mathcal{D}_R$.

We say that the DDH1 assumption holds for a bilinear group generator $\mathcal{G}$ if, for all probabilistic polynomial time (PPT) algorithms $\mathcal{A}$ and any security parameter λ, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH1}}(\lambda) < \epsilon(\lambda)$ for some negligible function ϵ. The DDH2 assumption also holds for the analogous distributions obtained from switching the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ (i.e., $\mathcal{D}_N := \left(h, h^a, h^b, g, h^{ab}\right) \in \mathbb{G}_2^3 \times \mathbb{G}_1 \times \mathbb{G}_2$ and $\mathcal{D}_R := \left(h, h^a, h^b, g, T\right) \in \mathbb{G}_2^3 \times \mathbb{G}_1 \times \mathbb{G}_2$).

IV. DEFINITIONS

In this Section, we introduce a new definition called a fuzzy vector signature (FVS) and its security model.

A. FUZZY VECTOR SIGNATURE

Definition 3.1 (Fuzzy Vector Signature): A fuzzy vector signature(FVS) scheme is defined by the following three PPT algorithms: Setup, Sign, and Verify.

- $\mathsf{Setup}\left(1^\lambda, n, \vec{x}, t\right)$: The setup algorithm takes the following as input: the security parameter 1^λ, a positive integer n (indicating a vector length that is a polynomial in λ), a vector $\vec{x}$ of length n, and a threshold value t. It outputs a signing key sk and a verification key $vk_{\vec{x},t}$ corresponding to the vector $\vec{x}$ and threshold value t.
- $\mathsf{Sign}\left(sk, \vec{y}, m\right)$: The sign algorithm takes the following as input: a signing key sk, vector $\vec{y} \in \mathbb{Z}_p^n$, and message m. Then, it outputs a corresponding signature $\sigma_{\vec{y}}$.
- $\mathsf{Verify}\left(vk_{\vec{x}}, \sigma_{\vec{y}}, m\right)$: The verify algorithm takes the following as input: a verification key $vk_{\vec{x},t}$ corresponding to the vector $\vec{x}$ and the threshold value t, signature $\sigma_{\vec{y}}$ corresponding to the vector $\vec{y}$, and message m. If $f_t(\vec{x}, \vec{y}) = 1$ and $\sigma_{\vec{y}}$ is a valid signature on m *w.r.t.* the verification key $vk_{\vec{x},t}$, it outputs 1. Otherwise, it outputs 0.

Correctness: We guarantee the correctness of a FVS scheme if the following condition holds:

For all $\vec{x}, \vec{y} \in \mathbb{Z}_p^n$ such that $f_t(\vec{x}, \vec{y}) = 1$ and $m \in \mathcal{M}$,

$$\Pr\left[(sk, vk_{\vec{x},t}) \leftarrow_R \mathsf{Setup}(1^\lambda, n, \vec{x}, t); \sigma_{\vec{y}} \leftarrow_R \mathsf{Sign}(sk, \vec{y}, m) : \mathsf{Verify}(vk_{\vec{x},t}, \sigma_{\vec{y}}, m) = 1\right] > 1 - \epsilon(\lambda)$$

where ϵ is a negligible function.

Threshold predicate: In the FVS scheme, for two attribute vectors $\vec{x}$ and $\vec{y}$ of a verification key and a signature, respectively, and for a threshold value t, a function f_t is defined as follows:

$$f_t(\vec{x}, \vec{y}) = \begin{cases} 1, & \text{if } d\left(\vec{x}, \vec{y}\right) \le t; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

B. SECURITY MODEL OF FVS

For the security of FVS, we consider three security games to capture i) VK privacy, ii) anonymity, and iii) existential unforgeability. Note that we implicitly assume that a certificate exists on sk and vk so that users know who they belong to.

1) VK PRIVACY

An adversary (semi-honest server) should not be able to obtain any information, beyond the absolute minimum necessary, about an input vector corresponding to a given verification key. We allow VK-private adversaries to make a polynomial number of queries to the *real-or-random VK-privacy oracle* (RoR^{VP}) if the input distributions of the oracle have a certain amount of min-entropy. We assume that the queries an adversary can make are *(T,k)-block-source*, as in [23].

Definition 3.2 (VK-private adversary): A *(T,k)-block-source* VK-private adversary $\mathcal{A}$ is an algorithm that is given as input a security parameter 1^λ and oracle access to $RoR^{VP}(\mathsf{mode}, \cdot)$ for some $\mathsf{mode} \in \{real, rand\}$, and each of its queries to RoR^{VP} is a random variable $\mathbf{X} = X_1, \ldots, X_T$. Note that $\mathbf{X}$ represents a joint distribution over $\mathcal{X}$ and, in our HVS scheme, $\mathcal{X} = \mathbb{Z}_p^n$. For every $i \in [T]$ and $x_1, \ldots, x_{i-1}$, it holds that $H_\infty(X_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \ge k$.

Definition 3.3 (Real-or-random VK-privacy oracle): The real-or-random VK-privacy oracle RoR^{VP} takes as input tuples of the form $(\mathsf{mode}, \mathbf{X})$, where $\mathsf{mode} \in \{real, rand\}$ and $X = (X_1, \ldots, X_T)$ is a $(T, k) - block - source$. If $\mathsf{mode} = real$ then the oracle samples $(x_1, \ldots, x_T) \leftarrow X^T$, and if $\mathsf{mode} = rand$ then the oracle samples $(x_1, \ldots, x_T) \leftarrow \mathcal{X}^T$ uniformly. It then invokes the algorithm Setup on each of $x_1, \ldots, x_T$ and outputs a tuple of pairs $\left((vk_{x_1}, sk_1), \ldots, (vk_{x_T}, sk_T)\right)$.

The experiments in real and random mode, $Expt_{VP,\Pi,\mathcal{A}}^{mode}(\lambda)$, are defined by the following game between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$:

- **Setup:** $\mathcal{S}$ selects a mode, real or random, in which it simulates the game.
- **Query:** $\mathcal{A}$ issues a polynomial number of queries to the oracle RoR^{VP}.
- **Guess:** $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ that represents a mode, real or random, selected by $\mathcal{S}$.

Definition 3.4 (VK Privacy): A fuzzy vector signature scheme $\Pi_f = (Setup, Sign, Vrfy)$ is (T, k)-*block-source* VK-private *if for any probabilistic polynomial-time* (T, k)-*block-source VK-private adversary* $\mathcal{A}$*, there exists a negligible function* $\nu(\lambda)$ *such that*

$$Adv^{VP}_{\Pi_f,\mathcal{A}}(\lambda) \triangleq \Big| \Pr\left[Expt^{real}_{VP,\Pi_f,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[Expt^{rand}_{VP,\Pi_f,\mathcal{A}}(\lambda) = 1\right] \Big| \leq \nu(\lambda) \quad (2)$$

2) ANONYMITY

An adversary (malicious external attacker) should not be able to obtain any information about an input vector corresponding to a given signature. The security notion of anonymity is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- **Init:** $\mathcal{A}$ commits to two target vectors $w_0^*, w_1^* \in \mathcal{W}$.
- **Setup:** $\mathcal{C}$ selects a vector $w \neq w_i^*$ for $i \in \{0, 1\}$ randomly, and runs Setup algorithm on w. It outputs a key pair (vk_w, sk) and gives them to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ sends a challenge message $m_{ch} \in \mathcal{M}$ to $\mathcal{C}$. $\mathcal{C}$ picks a random $b \in \{0, 1\}$ and runs Sign algorithm on a vector w_b^*. It outputs a signature $\sigma_{w_b^*}$ and gives it to $\mathcal{A}$.
- **Guess:** $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b' = b$.

Definition 3.5 (Anonymity): A fuzzy vector signature scheme $\Pi_f = (Setup, Sign, Vrfy)$ is anonymous *if for any probabilistic polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* $\nu(\lambda)$ *such that*

$$Adv^{Anon}_{\Pi_f,\mathcal{A}}(\lambda) \triangleq \left| \Pr\left[b = b'\right] - \frac{1}{2} \right| \leq \nu(\lambda) \quad (3)$$

3) STRONG EXISTENTIAL UNFORGEABILITY

An adversary (malicious external attacker) who does not know an input vector of either a verification key or a signature should not be able to forge a signature. That is, a forged signature output by an adversary should be incorrectly accepted as valid. The security notion of strong existential unforgeability under chosen message attacks is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- **Setup:** $\mathcal{C}$ gives a key pair $(vk_{w^*}, sk) \leftarrow \mathsf{Setup}(1^\lambda, n, w^*, t)$, for a target vector w^* of length n and a threshold t, to $\mathcal{A}$.
- **Signing Query:** $\mathcal{A}$ adaptively issues at most q signature queries $m_1, \ldots, m_q$. $\mathcal{C}$ answers each query by returning $\sigma_i(w^*) \leftarrow \mathsf{Sign}(sk, w^*, m_i)$.
- **Output:** $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*) and wins the game if $\mathsf{Verify}(vk_{w^*}, \sigma^*, m^*) = 1$ and if $(m^*, \sigma^*) \neq (m_i, \sigma_i)\ \forall i \in [1, q]$.

Definition 3.6 (Strong Existential Unforgeability): A fuzzy vector signature scheme $\Pi_f = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ is strongly existentially unforgeable under chosen message attacks *if for any probabilistic polynomial-time adversary* $\mathcal{A}$ *making at most q signature queries, the advantage that* $\mathcal{A}$ *wins this game, denoted by* $Adv^{EUF\text{-}CMA}_{\Pi_f,\mathcal{A}}$*, is negligible in* λ.

V. PROPOSED FVS SCHEME

In this section, we describe a fuzzy vector signature (FVS) scheme. In the FVS scheme, notwithstanding the presence of error between the fuzzy data corresponding to the verification key and the signature, the signature can be successfully verified if the error occurs below the threshold. More importantly, our FVS scheme is capable of addressing outliers, as we apply the concept of subsets to provide the threshold functionality. Another feature is that, in the FVS scheme, the signing key is also public information, and assuming the adversary has knowledge of the signing key, he/she can not generate a valid signature unless he/she has valid fuzzy data.

Assumptions: Throughout this paper, our FVS scheme assumes the following statements:

1) The user-generated data (i.e., biometrics, WiFi signal) can be represented in vector form [20], [24].
2) The user-generated data has sufficient entropy to provide "good enough" security. For example, in biometrics, two or more features can be used in combination, which is called *multimodal* biometric system [25]–[27].

Let t be the permissible threshold of error. When generating the verification key, select d subvectors of length $\ell(< n)$ for an input vector of length n. The selected d subvectors are used to generate d verification keys, and the set of d verification keys are output as the verification key of the FVS scheme.[4] If the number of errors is less than t, most of the subvectors will not contain components with errors. Thus, most of the d verification attempts will succeed. The correlation of each variable is represented in (4) of the scheme below, and a detailed explanation will be described later.

A. CONSTRUCTION

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p, and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be an asymmetric bilinear map. We assume that the user's fuzzy data is a vector of length n.

1) $\mathbf{Setup}(1^\lambda, n, \overrightarrow{w}, t)$: Let $\overrightarrow{w} = (w_1, \ldots, w_n) \in \mathbb{Z}_p^n$ and a hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. Generate a signing key sk and a verification key $vk_{\overrightarrow{w}}$ capable of handling up to t random errors, through the following steps:
    1) Determine the number of subsets d and the acceptance rate t' from the following formula:

$$\frac{n-\ell}{n} \times \frac{n-\ell-1}{n-1} \times \cdots \times \frac{n-\ell-t+1}{n-t+1} = \frac{t'}{d} \quad (4)$$

[4]Note that an attacker needs 2^ℓ attempts to get the user's fuzzy data that matches each verification key. For impersonation, the attacker should guess the user's fuzzy data that matches with more than t' verification keys, so we have to set ℓ and t' to be fairly difficult for an attacker to guess a valid data.

where ℓ is the number of elements included in each subset (parameterized by a security parameter λ) and t is a threshold value implying a tolerable error rate of an input vector.

2) Specify d subsets of an input vector $\overrightarrow{w}$ in the following ways:
   - Randomly select a set $I_j \subset \{1, \ldots, n\}$ where $|I_j| = \ell$ for $j \in [1, d]$;
   - Set a subset $W_j = \{w_i | i \in I_j\}$ of a vector $\overrightarrow{w}$ for $j \in [1, d]$;

3) Select random generators $g \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ and random elements $x_i, y_i, z_i \in \mathbb{Z}_p$ for $i \in [1, n]$.

4) Set $u_i = g^{x_i}, v_i = g^{y_i}, h_i = g^{z_i} \in \mathbb{G}_1$ for $i \in [1, n]$.

5) Select random elements $r_j, r_j' \in \mathbb{Z}_p$ for $j \in [1, d]$.

The signing key sk and the verification key $vk_{\overrightarrow{w}}$ are given by

$$sk = \left(g, H, \{u_i, v_i, h_i\}_{i\in[1,n]}\right),$$
$$vk_{\overrightarrow{w}} = \left(e, g, H, \{vk_{W_1}, \ldots, vk_{W_d}\}, \{I_1, \ldots, I_d\}, t'\right)$$

where $vk_{W_j} = \left(\left(\prod_{i\in I_j} g_2^{x_i+w_iy_i}\right)^{r_j} \cdot \left(\prod_{i\in I_j} g_2^{z_i}\right)^{r_j'}, g_2^{r_j}, g_2^{r_j'}\right) \in \mathbb{G}_2^3$.

2) **Sign**$(sk, \overrightarrow{w}', m)$: Let $\overrightarrow{w}' = (w_1', \ldots, w_n') \in \mathbb{Z}_p^n$. Generate a signature $\sigma_{\overrightarrow{w}'}$ through the following steps:

1) Compute $\left[\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3\right]$ as follows:
   - Select a random element $s \in \mathbb{Z}_p^*$;
   - $\sigma_{(1,i)} = \left(u_i \cdot (v_i)^{w_i'}\right)^s$ for $i \in [1, n]$;
   - $\sigma_{(2,i)} = h_i^s$ for $i \in [1, n]$;
   - $\sigma_3 = g^s$.

2) Compute σ_4 and σ_5 as follows:
   - Select a random element $k \in \mathbb{Z}_p^*$;
   - $\sigma_4 = H(\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3 \| g^k, m)$;
   - $\sigma_5 = k + \sigma_4 \cdot s$.

The signature $\sigma_{\overrightarrow{w}'}$ is given by

$$\sigma_{\overrightarrow{w}'} = \left(\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3, (\sigma_4, \sigma_5)\right).$$

3) **Verify**$(vk_{\overrightarrow{w}}, \sigma_{\overrightarrow{w}'}, m)$: Parse the signature $\sigma_{\overrightarrow{w}'}$ as $\left[\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3, (\sigma_4, \sigma_5)\right]$. To verify the given signature with a verification key $vk_{\overrightarrow{w}} = \left(\{vk_{W_1}, \ldots, vk_{W_d}\}, \{I_1, \ldots, I_d\}, t'\right)$, compute the following steps:

1) Set $vk_{W_j} = (k_{1,j}, k_{2,j}, k_{3,j})$ for $j \in [1, d]$.

2) Set $count = 0$. While $j \leq d$:
   - Compute $A_1 = \prod_{i\in I_j} \sigma_{(1,i)}, A_2 = \prod_{i\in I_j} \sigma_{(2,i)}$;
   - If $e(\sigma_3, k_{1,j}) = e(A_1, k_{2,j}) \cdot e(A_2, k_{3,j})$, set the result $R_j = 1$.
     Otherwise, set $R_j = 0$;
   - Compute $count = count + R_j$;

3) Compute $g^{k'} = g^{\sigma_5} \cdot \sigma_3{}^{-\sigma_4}$;

4) Check if $\sigma_4 = H\left(\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3 \| g^{k'}, m\right)$;

5) If $count \geq t'$, output 1. Otherwise, output 0.

Remark 1: Equation (4) presents the method to calculate the probability that t components with errors (out of n) are not included in the subset W_j for $j \in [1, d]$. In order for a verification key of a subset with ℓ elements to be successfully authenticated, t errored elements should belong to the remaining $(n - \ell)$ elements excluding the ℓ selected elements, from all the n elements. Moreover, this implies that at least one of the verification keys can be successfully authenticated with the probability in (4).

Remark 2: Note that (σ_4, σ_5) in a signature are in the form of the Schnorr signature [28]. One can use any one-time signature scheme, in place, in which the random s is used as a secret key.

Remark 3: Both a signing key sk and a verification key vk are public, and only the vectors $\overrightarrow{w}$ and $\overrightarrow{w}'$ are private.

Correctness: To verify that correctness holds, observe that for any signature $\sigma_{\overrightarrow{w}'} = \left(\left(\sigma_{(1,i)}, \sigma_{(2,i)}\right)_{i\in[1,n]}, \sigma_3, (\sigma_4, \sigma_5)\right)$ and verification key $vk_{\overrightarrow{w}} = \left\{k_{1,j}, k_{2,j}, k_{3,j}\right\}_{j\in[1,d]}$, we have

$$\begin{aligned}
\triangleright A_{1,j} &= \prod_{i\in I_j} \sigma_{(1,i)} \\
&= \prod_{i\in I_j} \left(g^{x_i+w_i'y_i}\right)^s = g^{s\cdot\Sigma_{i\in I_j}(x_i+w_i'y_i)} \\
\triangleright A_{2,j} &= \prod_{i\in I_j} \sigma_{(2,i)} \\
&= \prod_{i\in I_j} \left(g^{z_i}\right)^s = g^{s\cdot\Sigma_{i\in I_j} z_i} \\
\triangleright e(\sigma_3, k_{1,j}) &= e\left(g^s, \left(\prod_{i\in I_j} g_2^{x_i+w_iy_i}\right)^r \cdot \left(\prod_{i\in I_j} g_2^{z_i}\right)^{r'}\right) \\
&= e\left(g^s, \left(\prod_{i\in I_j} g_2^{x_i+w_iy_i}\right)^r\right) \cdot e\left(g^s, \left(\prod_{i\in I_j} g_2^{z_i}\right)^{r'}\right) \\
&= e\left(g^s, g_2^{r\cdot\Sigma_{i\in I_j}(x_i+w_iy_i)}\right) \cdot e\left(g^s, g_2^{r'\cdot\Sigma_{i\in I_j} z_i}\right) \\
&= e\left(g^{s\cdot\Sigma_{i\in I_j}(x_i+w_iy_i)}, g_2^r\right) \cdot e\left(g^{s\cdot\Sigma_{i\in I_j} z_i}, g_2^{r'}\right)
\end{aligned}$$

If $\{w_i'\}_{i\in I_j} = \{w_i\}_{i\in I_j}$, the equation $e(\sigma_3, k_{1,j}) = e(A_{1,j}, k_{2,j}) \cdot e(A_{2,j}, k_{3,j})$ holds for $j \in [1, d]$.

$$\triangleright g^{k'} = g^{\sigma_5} \cdot \sigma_3{}^{-\sigma_4} = g^{k+h\cdot s} \cdot (g^s)^{-h} = g^k$$

B. SECURITY PROOF

To prove the security of our FVS scheme, we demonstrate it in three respects, as described in Section IV-B.

Lemma 4.1:For (T, k)-block-sources where $T = poly(\lambda)$ and $k \geq d \cdot \log p + 2\log(\frac{1}{\epsilon})$, the FVS scheme is (statistically) VK private.

Proof: Let $\mathcal{A}$ be a computationally unbounded *(T,k)-block-source* VK-private adversary that makes a polynomial number $T = poly(\lambda)$ of queries to the RoR^{VP} oracle. We prove that $\mathcal{A}$'s view in the experiment $Expt^{real}_{VP,\mathsf{FVS},\mathcal{A}}$, denoted by $View_{real}$, is statistically indistinguishable from the view in the experiment $Expt^{rand}_{VP,\mathsf{FVS},\mathcal{A}}$, denoted by $View_{rand}$.

Let $W = (W_1, \ldots, W_T) \in \mathcal{W}^T$ be the random variable corresponding to the *(T,k)-block-source* with which $\mathcal{A}$ queries to the RoR^{VP} oracle. Then, we can assume that for $i \in [1, T]$, $View_{\mathsf{mode}} = \left[(S_1, V_1), \ldots, (S_T, V_T)\right]$, where

$$S_i = \left(\{g^{x_{i,j}}, g^{y_{i,j}}, g^{z_{i,j}}\}_{j\in[1,n]}\right),$$
$$V_i = \Bigg(\Big\{\Big(\prod_{j\in I_i^{(k)}} {g_2}^{x_{i,j}+w_{i,j}y_{i,j}}\Big)^{r_i^{(k)}} \cdot\Big(\prod_{j\in I_i^{(k)}} {g_2}^{z_{i,j}}\Big)^{\overline{r}_i^{(k)}}\Big\}_{k\in[1,d]}, \{I_i^{(k)}\}_{k\in[1,d]}\Bigg).$$

If $\mathsf{mode} = real$, $(w_{i,1}, \ldots, w_{i,n}) \leftarrow W_i$ and if $\mathsf{mode} = rand$, $(w_{i,1}, \ldots, w_{i,n})$ is uniformly distributed over $\mathcal{W}_\lambda$, for $i \in [1, T]$. We prove that the distribution $View_{mode}$ is statistically indistinguishable from the uniform distribution for $\mathsf{mode} \in \{real, rand\}$.

First, we prove that the collection of functions $\mathcal{F} = \{f_S : \mathcal{W} \rightarrow \mathbb{G}_2^d\}$ is *universal*. In our HVS scheme, $\mathcal{W}$ is set to $\mathbb{Z}_p^n$, and a function $f_S(w)$ is defined as follows:

$$f_S(w) = \Big\{\Big(\prod_{j\in I_i^{(k)}} {g_2}^{x_{i,j}+w_{i,j}y_{i,j}}\Big)^{r_i^{(k)}} \cdot \Big(\prod_{j\in I_i^{(k)}} {g_2}^{z_{i,j}}\Big)^{\overline{r}_i^{(k)}}\Big\}_{k\in[1,d]}$$

$\mathcal{F} = \{f_S\}$ is *universal* if for any $w_1, w_2 \in \mathcal{W}$ $(w_1 \neq w_2)$, it holds that $\Pr_{f_S\leftarrow\mathcal{F}}\left[f_S(w_1) = f_S(w_2)\right] = \frac{1}{|\mathbb{G}_2^d|}$. For $f_S(w_1)$ and $f_S(w_2)$,

$$\begin{aligned}
&\triangleright f_S(w_1) = f_S(w_2)\\
&\quad\Leftrightarrow r^{(k)} \cdot \Sigma_{i\in I^{(k)}}(x_i + w_{1,i}y_i) + \overline{r}^{(k)} \cdot \Sigma_{i\in I^{(k)}} z_i\\
&\quad= r^{(k)} \cdot \Sigma_{i\in I^{(k)}}(x_i + w_{2,i}y_i) + \overline{r}^{(k)} \cdot \Sigma_{i\in I^{(k)}} z_i\\
&\quad\Leftrightarrow \Sigma_{i\in I^{(k)}} w_{1,i}y_i = \Sigma_{i\in I^{(k)}} w_{2,i}y_i\\
&\quad\Leftrightarrow \Sigma_{i\in I^{(k)}}(w_{1,i} - w_{2,i})y_i = 0 \pmod p)\\
&\qquad\qquad\text{for all } k \in [1, d]
\end{aligned}$$

Thus, it holds that $\Pr\left[f_S(w_1) = f_S(w_2)\right] = \frac{1}{p^d} = \frac{1}{|\mathbb{G}_2^d|}$.

Secondly, as the range of min-entropy is $k \geq d \cdot \log p + 2\log(\frac{1}{\epsilon})$, we can derive the following formula: $\log|\mathbb{G}_2^d| \leq H_\infty(\mathcal{W}) - 2\log(\frac{1}{\epsilon})$. Since $\mathcal{F}$ is a universal collection of functions $f_S : \mathcal{W} \rightarrow \mathbb{G}_2^d$, by the left-over hash lemma [29], the distribution of $f_S(\mathcal{W})$ is statistically close to uniform, as proved in Lemma 2.3 of [23]. Therefore, the probability that $\mathcal{A}$ distinguishes the distribution $View_{\mathsf{mode}}$ from the uniform distribution is negligible. □

Lemma 4.2: If the decisional Diffie–Hellman assumption holds in $\mathbb{G}$*, the FVS scheme is anonymous in the random oracle model.*

Overview: We demonstrate that if the decisional Diffie–Hellman (DDH) assumption holds, the FVS scheme, described in Section V-A, is selectively secure. In the selective security model, an adversary commits two vectors, w_0^* and w_1^*, at the beginning of the game. For $w_0^* = (w_{0,1}^*, \ldots, w_{0,n}^*)$ and $w_1^* = (w_{1,1}^*, \ldots, w_{1,n}^*)$, we define the set of indexes $D = \{\ i \mid w_{0,i}^* \neq w_{1,i}^*, i \in [1, n]\}$. For simplicity, we assume that $D = \{1, \ldots, |D|\}$ where $|D| \leq n$. The proof proceeds by the sequence of games. In the hybrid games, w^* implies the vector used to generate the challenge signature σ^*. The vector w^* changes as follows from Game 0 to Game $|D|$:

- Game 0. $w^* = (w_{0,1}^*, \ldots, w_{0,n}^*) = w_0^*$
- Game 1. $w^* = (\mathbf{R_1}, w_{0,2}^*, \ldots, w_{0,n}^*)$
- Game 2. $w^* = (R_1, \mathbf{R_2}, w_{0,3}^*, \ldots, w_{0,n}^*)$

$\vdots$

- Game $|D|$. $w^* = (R_1, \ldots, \mathbf{R_{|D|}}, w_{0,|D|+1}^*, \ldots, w_{0,n}^*)$

Note that $w_{0,i}^* = w_{1,i}^*$ for $i \in [|D| + 1, n]$. From the Game $|D| + 1$ to Game $2|D|$, the hybrid game proceeds in reverse order from Game 0 to Game $|D|$.

- Game $|D| + 1$. $w^* = (R_1, \ldots, R_{|D|-1}, \mathbf{w_{1,|D|}^*}, w_{0,|D|+1}^*, \ldots, w_{0,n}^*)$
- Game $|D| + 2$. $w^* = (R_1, \ldots, R_{|D|-2}, \mathbf{w_{1,|D|-1}^*}, w_{1,|D|}^*, w_{0,|D|+1}^*, \ldots, w_{0,n}^*)$

$\vdots$

- Game $2|D|$. $w^* = (\mathbf{w_{1,1}^*}, \ldots, w_{1,|D|}^*, w_{0,|D|+1}^*, \ldots, w_{0,n}^*) = w_1^*$

We demonstrate the anonymity of the FVS scheme through hybrid games. We show that the distributions of Game $(j - 1)$ and Game j are computationally indistinguishable, for $j \in [1, |D|]$, by the following lemma. That is, if there exists an adversary that distinguishes $Game_{j-1}$ and $Game_j$, it is feasible to solve the decisional Diffie–Hellman (DDH) problem with a non-negligible probability in the random oracle model.

Proof: An adversary $\mathcal{A}$ has a non-negligible difference ϵ between its advantages in $Game_{j-1}$ and $Game_j$ of the FVS scheme. An adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subprotocol to solve the DDH problem. Given a random tuple $(g, g_2, g^a, g^b, T) \in \mathbb{G} \times \mathbb{G}_2 \times \mathbb{G}^3$, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

$\triangleright$ **Init:** $\mathcal{A}$ outputs two vectors $w_0^*, w_1^* \in \mathcal{W}$ at the beginning of the game. $\mathcal{C}$ internally selects a bit $\beta \in \{0, 1\}$.

$\triangleright$ **Setup:** $\mathcal{C}$ randomly selects a vector $w = (w_1, \ldots, w_n)$, which is not equal to either w_0^* or w_1^*. Note that in the security game, the vector w^* is set to $(w_1^*, \ldots, w_n^*) = (R_1, \ldots, R_j, w_{0,j+1}^*, \ldots, w_{0,n}^*) \in \mathbb{Z}_p^n$. $\mathcal{C}$ generates a signing key sk and a verification key $vk_{\overrightarrow{w}}$ as follows:

- Choose random values $\{r_{1,i}, r_{2,i}, r_{3,i}\}_{i\in[1,n]}$ and $\{R_j\}_{j\in[1,d]}$ in $\mathbb{Z}_p$
- Compute $g^{x_i} = (g^a)^{-w_i^*} \cdot g^{r_{1,i}}$, $g^{y_i} = g^a \cdot g^{r_{2,i}}$, $g^{z_i} = (g^a)^{-r_{3,i}}$ (Implicitly, $x_i = -aw_i^* + r_{1,i}$, $y_i = a + r_{2,i}$, and $z_i = -ar_{3,i}$)
- Set $sk = \{g^{x_i}, g^{y_i}, g^{z_i}\}$ for all $i \in [1, n]$
- Randomly choose sets $I_j \subset [1, n]$ of length ℓ and set $W_j = \{w_i | i \in I_j\}$ for $j \in [1, d]$
- Compute $vk = \{vk_{W_1}, \ldots, vk_{W_d}, \{I_1, \ldots, I_d\}, t'\}$ as follows:

- Set d and t' to satisfy (4)
- Compute $v_{j,1} = g_2^{\frac{R_j}{\Sigma_{i\in I_j}(w_i - w_i^*)} \cdot \Sigma_{i\in I_j}(r_{1,i}+w_i r_{2,i})}$
  (Implicitly, for some random values r_j and r'_j, $v_{j,1} = \left(\prod_{i\in I_j} g_2^{x_i+w_i y_i}\right)^{r_j} \cdot \left(\prod_{i\in I_j} g_2^{z_i}\right)^{r'_j}$)
- Compute $v_{j,2} = g_2^{\frac{R_j}{\Sigma_{i\in I_j}(w_i - w_i^*)}}$, $v_{j,3} = g_2^{\frac{R_j}{\Sigma_{i\in I_j} r_{3,i}}}$
  (Implicitly, $v_{j,2} = g_2^{r_j}$ and $v_{j,3} = g_2^{r'_j}$)
- Set $vk_{W_j} = (v_{j,1}, v_{j,2}, v_{j,3})$

▷ **H-query:** $\mathcal{A}$ gives a message X of any length to $\mathcal{C}$. $\mathcal{C}$ implicitly outputs a random value h as an output of a hash function H, namely, $h = H(X)$. $\mathcal{C}$ stores an input–output pair (X, h) in the H-list. If $\mathcal{A}$ queries the same message stored in H-list, $\mathcal{C}$ outputs a corresponding output stored together in H-list. $\mathcal{A}$ may perform H-query at any time during the game.

▷ **Challenge:** $\mathcal{A}$ gives $\mathcal{C}$ a message m^*. $\mathcal{C}$ outputs a challenge signature $\sigma_{w^*} = \left(\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3, (\sigma_4,\ \sigma_5)\right)$ as follows:

- $\sigma_{(1,i)} = (g^b)^{r_{1,i}+w_i^* r_{2,i}}$, $\sigma_{(2,i)} = T^{-r_{3,i}}$
  $\left(\text{Implicitly, } (\sigma_{(1,i)}, \sigma_{(2,i)}) = \left(g^{(x_i+w_i^* y_i)b}, g^{z_i b}\right)\right)$
- $\sigma_3 = g^b$
  (Implicitly, b is a random value used to generate a challenge signature.)
- $(\sigma_4, \sigma_5) = (h, t)$ for some random $h, t \in \mathbb{Z}_p$
  Note that $t = k + h \cdot b$ where $h = H\left(\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3 || g^k, m^*\right)$. $\mathcal{C}$ adds a pair (X^*, h) to the H-list, where $X^* = H\left(\{\sigma_{(1,i)}, \sigma_{(2,i)}\}_{i\in[1,n]}, \sigma_3, g^t(g^b)^{-h}, m^*\right)$.

▷ **Guess:** $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$ in response to the challenge signature. If $\beta' = \beta$, $\mathcal{C}$ outputs 1; otherwise, it outputs 0.

If $\mathcal{A}$ outputs a correct guess, it implies that $\sigma_{(2,i)} = T^{-r_{3,i}} = \left(g^{z_i}\right)^b = g^{-ab\cdot r_{3,i}}$, i.e., $T = g^{ab}$ in the decisional Diffie–Hellman problem. Therefore, $\mathcal{A}$'s advantage in distinguishing $Game_{j-1}$ and $Game_j$ is directly transferred to the advantage of $\mathcal{C}$ in solving the decisional Diffie–Hellman problem. □

Lemma 4.3: If the discrete logarithm assumption holds in $\mathbb{G}$, the FVS scheme is strongly existentially unforgeable under chosen message attacks in the random oracle model.

Overview: We demonstrate that if the one-time signature (OTS) is strongly existentially unforgeable in the *multi-user* setting against chosen message attacks (MU-EUF-CMA), the FVS scheme described in Section V-A is strongly existentially unforgeable. In multi-user unforgeability against chosen message attacks [30], N independent public keys are given to the attacker and the attacker is said to break the security of the scheme if he is able to generate (after obtaining q many signatures on public keys of his choice) a valid forgery that verifies under any of the public keys. In the notion of *strong* security, a new signature on a previously queried message is considered as a fresh forgery.

The fuzzy vector signature $\sigma = (\{\sigma_{1,i}\}, \{\sigma_{2,i}\}, \sigma_3, \sigma_4, \sigma_5)$ consists of two parts: (1) The first part corresponds to $(\{\sigma_{1,i}\}, \{\sigma_{2,i}\}, \sigma_3)$, which are generated by the user's fuzzy data. In order to generate a valid forgery, the attacker should obtain information about the user's fuzzy data, which is infeasible since we have proved the privacy of the verification key and the anonymity of the signature in Lemma 4.1 and 4.2, respectively. It is possible to forge $(\{\sigma_{1,i}\}, \{\sigma_{2,i}\}, \sigma_3)$ due to its malleability, i.e., $(\{\sigma_{1,i}^r\}, \{\sigma_{2,i}^r\}, \sigma_3^r)$ for a random $r \in \mathbb{Z}_p$, but it is still difficult to generate a valid forgery of the fuzzy vector signature σ, because it requires generating a forgery of the Schnorr signature, i.e., (σ'_4, σ'_5), on message $(\{\sigma_{1,i}^r\}, \{\sigma_{2,i}^r\}, \sigma_3^r)$. (2) The second part is a one-time signature, (σ_4, σ_5), and σ_3 is the corresponding public key. Since it is difficult to forge the first part of σ, an attacker $\mathcal{A}$ against the FVS scheme should generate a forgery for the second part of the signature to output the forgery of the FVS scheme.

Therefore, if the Schnorr OTS scheme is strongly MU-EUF-CMA secure, then the proposed FVS scheme is strongly EUF-CMA secure. An attacker $\mathcal{B}$ against the Schnorr OTS scheme outputs the forgery using the attacker $\mathcal{A}$ as a subroutine. In the multi-user setting, the attacker $\mathcal{B}$ is given q independent public keys, and one signing query can be done for each public key. $\mathcal{B}$ can then respond to q signing queries from $\mathcal{A}$. Since $\mathcal{A}$ can not determine the user's fuzzy data from the given information, $\mathcal{A}$ should forge one of the signatures received in response to the signing query, which is the forgery of the Schnorr OTS scheme.

Since the Schnorr OTS scheme is strongly MU-EUF-CMA secure under the discrete logarithm assumption in the random oracle model [30], we can say that the FVS scheme is strongly EUF-CMA secure if the discrete logarithm assumption holds.

Proof: An attacker $\mathcal{B}$ uses an adversary $\mathcal{A}$ (which forges a signature in the FVS scheme with probability ϵ) as a subroutine to forge a signature in the Schnorr OTS scheme by providing answers to $\mathcal{A}$'s oracle queries. Given q independent public keys $\left(g, g^{s_1}, \ldots, g^{s_q}\right)$ of the Schnorr OTS scheme, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

▷ **Setup.** $\mathcal{B}$ picks random values $\overrightarrow{w} = (w_1, \ldots, w_n) \in \mathbb{Z}_p^n$ and positive integers n, t. $\mathcal{B}$ runs $\mathsf{Setup}(1^\lambda, n, \overrightarrow{w}, t)$ to set a signing key sk and a verification key vk. $\mathcal{B}$ gives the key pair (sk, vk) to $\mathcal{A}$. Note that the exponents used to generate the signing key, i.e., $(x_1, \ldots, x_n)$, $(y_1, \ldots, y_n)$, and $(z_1, \ldots, z_n)$, are selected uniformly at random in $\mathbb{Z}_p$.

▷ **Signing query.** For $j \in [1, q]$, $\mathcal{A}$ queries a message m_j and $\mathcal{B}$ responds to the query as follows:

1) Generate $\sigma_{1,i}^{(j)} = (g^{s_j})^{x_i+w_i y_i}$ and $\sigma_{2,i}^{(j)} = (g^{s_j})^{z_i}$ for $i \in [1, n]$, and set $M_j = \left(\{\sigma_{1,i}^{(j)}\}_{i\in[1,n]}, \{\sigma_{2,i}^{(j)}\}_{i\in[1,n]}, g^{s_j}, m_j\right)$;
2) Query (j, M_j) to the signing oracle of the Schnorr OTS scheme, which means a signing query on message M_j under the j-th public key, and receive (h_j, c_j);
3) Set $\Sigma_j = \left(\left\{\sigma_{1,i}^{(j)}\right\}_{i\in[1,n]}, \left\{\sigma_{2,i}^{(j)}\right\}_{i\in[1,n]}, g^{s_j}, h_j, c_j\right)$ and gives Σ_j to $\mathcal{A}$.

TABLE 2. Comparison of online bandwidth of helper data with reusable fuzzy extractor [1].

	n	p	t	ℓ	d	The Size of Helper data (online bandwidth)
[1]	128	7.8%	10	100	24.7×10^2	288 KB
	128	11.7%	15	100	12.3×10^4	14.337 KB
	128	15.6%	20	100	6.11×10^6	696 MB
	128	19.5%	25	100	3.04×10^8	34,600 MB
Ours	128	7.8%	10	100	1.73×10^7 $(t' = 1)$	24 KB
	128	11.7%	15	100	3.53×10^{11} $(t' = 1)$	24 KB
	128	15.6%	20	100	3.85×10^{16} $(t' = 1)$	24 KB
	128	19.5%	25	100	7.66×10^{22} $(t' = 1)$	24 KB

$\triangleright$ **Output.** $\mathcal{A}$ outputs $(m^*, \Sigma^*) = \big(m^*, (\{\sigma^*_{1,i}\}_{i\in[1,n]}, \{\sigma^*_{2,i}\}_{i\in[1,n]}, \sigma^*_3, \sigma^*_4, \sigma^*_5)\big)$. $\mathcal{B}$ checks if $(m^*, \Sigma^*) \neq (m_j, \Sigma_j)$ for any $j \in [1, q]$, and if $\mathsf{FVS.Verify}(vk, \Sigma^*, m^*)$ outputs 1. In that case, $\mathcal{B}$ outputs a forgery of the Schnorr OTS scheme as follows:

1) Find j^* such that $\sigma^*_3 = g^{s_{j^*}}$;
2) Set $M^* = (\{\sigma^*_{1,i}\}_{i\in[1,n]}, \{\sigma^*_{2,i}\}_{i\in[1,n]}, \sigma^*_3, m^*)$;
3) Output $\big(j^*, M^*, (\sigma^*_4, \sigma^*_5)\big)$.

Since $\mathsf{FVS.Verify}(vk, \Sigma^*, m^*) = 1$ holds, it means that (σ^*_4, σ^*_5) is the Schnorr signature on message $(\{\sigma^*_{1,i}\}_{i\in[1,n]}, \{\sigma^*_{2,i}\}_{i\in[1,n]}, \sigma^*_3, m^*)$ under the public key σ^*_3. Therefore, if the Schnorr OTS scheme is strongly existentially unforgeable in the multi-user setting, the proposed FVS scheme is strongly existentially unforgeable under chosen message attack.

In [30], it is proved that the Schnorr OTS scheme is strongly existentially unforgeable against chosen message attacks in the multi-user setting under the discrete logarithm assumption in the random oracle model. As a result, the strong existential unforgeability against chosen message attacks of our proposed FVS scheme can be proven in the random oracle model by reduction to the discrete logarithm problem. $\square$

C. ANALYSIS

In [1], Canetti *et al.* proposed a reusable fuzzy extractor,[5] which can address outliers included in the input vector, similar to fuzzy vector signature (FVS). However, the reusable fuzzy extractor requires a large amount of storage for the user during *online* authentication. In order for the user to extract the cryptographic key for authentication using the reusable fuzzy extractor, he/she requires his/her own helper data in addition to his/her fuzzy data. The user's helper data consists of *sampling sets* and *digital lockers*. For an input string $w = (w_1, \ldots, w_n) \in \{0, 1\}^n$, the helper data H is generated as follows:

1) Sample $r \leftarrow_R \{0, 1\}^\lambda$
2) For $i = 1, \ldots, d$
   a) Choose $1 \leq j_{i,1}, \ldots, j_{i,\ell} \leq n$ uniformly at random;
   b) Set $v_i = w_{j_{i,1}}, \ldots, w_{j_{i,l}}$;
   c) Compute $c_i = \mathsf{lock}(v_i, r)$;
   d) Set $H_i = \{(j_{i,1}, \ldots, j_{i,\ell}), c_i\}$.
3) Set $H = (H_1, \ldots, H_d)$

[5]This scheme assumes the existence of "digital locker" which is a powerful tool. A method to obtain composable digital lockers based on a t-strong vector decisional Diffie-Hellman assumption, which is non-standard, is given in [31].

The size of each sampling set $(j_{i,1}, \ldots, j_{i,\ell})$ is $\ell \log n$ and thus, the size of the entire sampling set included in the helper data is $d \cdot \ell \log n$. The size of each digital locker c_i is equal to the size of the output of the lock function, which is designed using the hash function. If the output size of the underlying hash function is h, the size of the entire digital lockers included in the helper data is $d \cdot h$. Therefore, the size of the helper data required in the authentication phase is $\big(d \cdot (\ell \log n + h)\big)$-bit. Note that the size of each sampling set, denoted by ℓ, should be set to be larger than or equal to the security parameter λ in order to be secure against brute force attacks on each digital locker. If $n = 128$, $\lambda = 128$, and $h = 256$, the size of the helper data is approximately 34,600 MB, which is significantly large for the user to store (assuming that the probability of errors is 19.5%). The number of sampling sets, denoted by d, can be calculated using the equation provided in [1], i.e., $d \approx -\ln\delta \cdot e^{\frac{t\ell}{n}}$ where δ is an allowable error parameter. For simplicity, we assume that $\delta = e^{-1}$, and if δ becomes smaller, d becomes larger.

In the proposed FVS scheme, an additional user-specific signing key is used (similar to the helper data of the reusable fuzzy extractor) in addition to the user's fuzzy data during authentication. The signing key of the FVS scheme is public information, similar to helper data, and it consists of $(3n + 1)$ elements in the $\mathbb{G}_1$ group. Thus, the size of the signing key is $(3n + 1)|\mathbb{G}_1|$. Note that the size of the signing key depends only on the length of the user's input vector. At the 128-bit security level, the size of the element in the $\mathbb{G}_1$ group (used in the asymmetric bilinear group) is 512-bit, i.e., $|\mathbb{G}_1| = 512$ [33]. If $n = 128$ and $\lambda = 128$, the size of the signing key is approximately 24 KB. In Table 2, we compare the network bandwidth usage required to transmit the *helper data* of the reusable fuzzy extractor [1] and the signing key of the FVS scheme (which plays the same role as the helper data) to the user. If the user does not store his/her helper data (or signing key) in the server and stores it internally, the online

TABLE 3. Comparison with reusable fuzzy extractor in [32].

	n	p	The Size of Helper data
[32]	512 1024 2048	0.2	3.00 MB 1.59 MB 7.80 MB
Ours	512 1024 2048	0.2	0.031 MB 0.063 MB 0.125 MB

TABLE 4. Comparison with reusable fuzzy extractors.

	Reusability	Handling Ourliers	Standard Assumptions
Canetti *et al.* [1]	O	O	X
Apon *et al.* [34]	O	X	O
Wen *et al.* [35]	O	X	O
Cheon *et al.* [32]	O	O	X
Wen *et al.* [36]	O	X	O
Ours	O	O	O

bandwidth in Table 2 denotes the storage required on the user side.

Recently, several reusable fuzzy extractor schemes have been introduced [32], [34]–[36]. In [34]–[36], the size of the helper data has been reduced compared to [1], but they are not capable of handling the outliers in the user's fuzzy data. In [32], Cheon *et al.* have modified the reusable fuzzy extractor proposed in [1] and reduced the size of helper data by adopting a threshold scheme. However, this scheme still requires much more storage space as compared to ours. In Table 3, we describe the size of each user's helper data when the security level is 80 and the error rate is 0.2. At the 80-bit security level, $|\mathbb{G}_1|$ is 171-bit [33], and the size of helper data in our FVS scheme is $(3n + 1)|\mathbb{G}_1|$. As for the scheme of [32], the parameters shown in Table 2 of [32] are used as such.

1) TRADE-OFF WITH THE SIZE OF VERIFICATION KEY

In the proposed FVS scheme, the size of the verification key is $(|\mathbb{G}_1| + d \cdot 3|\mathbb{G}_2| + d \cdot \ell \cdot \log n)$-bit, which is sizeable even considering the server storage capacity. At the 128-bit security level, the size of the element in the $\mathbb{G}_2$ group (used in the asymmetric bilinear group) is 3,072-bit, i.e., $|\mathbb{G}_2| = 3{,}072$ [33]. If $d = 1.73 \times 10^7$, $n = 128$, and $\ell = 100$, the size of the verification key is approximately 20 GB. Nevertheless, FVS is desirable in that it significantly reduces storage and computational costs on the *user side*. And for the server, FVS can be advantageous in shortening verification time because it uses the *threshold* concept. The server only needs to repeat the verification process until there is *one* success. This means it does not have to repeat all d times. In addition, by applying various implementation-oriented techniques such as the multi-threading method or by adjusting parameters, the computational cost (of the server) can be brought down to a reasonable level (as shown in Section V-D).

TABLE 5. Experimental result for FVS (time: msec).

	Tests (60/80/100 bits)	Sign	Verify
d224-224	`Test1`: 60bits, p=1.6% `Test2`: 80bits, p=1.2% `Test3`: 100bits, p=1%	541 724 895	1144 1560 2002
d347-337	`Test1`: 60bits, p=1.6% `Test2`: 80bits, p=1.2% `Test3`: 100bits, p=1%	1437 1918 2430	2649 3689 4412

D. EXPERIMENTAL RESULT

We give some empirical results regarding computation overhead. For a private fuzzy data, we can consider binary strings extracted from biometrics with a reasonably low error rate. For example, as shown in [37], a sufficiently long bit-string can be extracted from a fingerprint information with about 1% EER (equal error rate). Our test assumes that the bit-length of a private fuzzy data is more than 60 bits and its error rate is about 1% EER. Note that we can extract fuzzy data of sufficient length (with enough entropy) to provide a high level of security by combining various features of the user, like the *multimodal* biometric system [25]–[27]. In addition, we can configure the system parameters so that the EER is close to 1%.

The test of generation of a fuzzy signature (for a client or a user) was performed on Intel Core(TM) i7-4770K CPU clocked at 3.50GHz and 16GB RAM on the Windows10 64bit OS. The test of verification of a fuzzy signature (for a server) was performed on Intel Core(TM) i7-6700U CPU clocked at 3.40GHz and 8GB RAM on the Windows10 64bit OS. It applied the JAVA multi-threading method (i.e., ExecutorService [38]) to improve the processing speed of the verification. We make use of two *d-type* curves called d224-224 and d347-337 from the PBC library [39] and the JPBC library [39], [40] running on top of Gnu GMP [41]. Refer to the parameters for d224-224 and d347-337 curve[6] in Appendix. Every test result is the average of 100 tests. Table 5 shows the running time of signature generation and signature verification of our FVS scheme.

- `Test1`: $n = 60, p = 1.6\%, \ell = 59, t = 1, d = 60$
- `Test2`: $n = 80, p = 1.2\%, \ell = 79, t = 1, d = 80$
- `Test3`: $n = 100, p = 1\%, \ell = 99, t = 1, d = 100$

VI. APPLICATION

In this section, we propose a secure fuzzy authentication protocol using the fuzzy vector signature (FVS) scheme with a challenge–response mechanism. In fuzzy authentication, noisy and fuzzy data (for example, biometric information such as fingerprint or iris) are used as authentication methods,

[6]d224-224 and d347-337 provide 1344-bits and 2022-bits of security which are comparable to that provided by 1344-bit and 2022-bit RSA cryptosystems, respectively.

and therefore, it is important to address errors that occur during the authentication process. Recently, fuzzy data in authentication have also resulted in a number of privacy issues [7], [8], [42]. The proposed fuzzy authentication protocol has the following advantages:

- Authentication can be performed while handling fine errors, as well as in the presence of *outliers* in fuzzy data.
- No information about the fuzzy data can be obtained from the values stored in the authentication server, thus ensuring user privacy for *malicious servers*.
- No information about the fuzzy data associated with the transmitted value can be obtained from the authentication process, so that a malicious external attacker can not compromise the user privacy.

Note that other fuzzy signature schemes can be employed in the fuzzy authentication protocol in a similar manner. However, none of them provide *VK privacy* (introduced in Section IV-B), and therefore fuzzy authentication protocols employing them are also vulnerable to malicious servers. On the other hand, the proposed fuzzy vector signature scheme provides *VK privacy*, and thus the fuzzy authentication protocol based on it is more secure.

The proposed fuzzy authentication protocol consists of two phases. In the enrollment phase, a user generates a signing key and a verification key and registers them with an authentication server. In the authentication phase, a user generates a signature and authenticates himself/herself as an authorized user (i.e., registered in advance in the enrollment phase) to the server. Let $\mathsf{FVS} = (\mathsf{FVS.Setup}, \mathsf{FVS.Sign}, \mathsf{FVS.Vrfy})$ be the underlying FVS scheme. The fuzzy authentication protocol is described as follows:

► **Enrollment Phase**

1) User: On the input of a security level 1^k, the user generates a signing key and a verification key, denoted by sk_{ID} and vk_{ID}, respectively, as follows:
   - Recognize his/her fuzzy data and extracts features $\overrightarrow{w} = (w_1, \ldots, w_n) \in \mathbb{Z}_p^n$ from it
   - Run $\mathsf{FVS.Setup}\ (1^k, n, \overrightarrow{w}, t) \rightarrow (sk_{ID}, vk_{ID})$

   Note that the vector length n and the threshold value t are set according to system policies. The user transmits a key pair (sk_{ID}, vk_{ID}) along with his/her identity ID to the server.
2) Server: The authentication server stores a pair (ID, sk_{ID}, vk_{ID}) and completes the registration of the user.

► **Authentication Phase**

1) User: The user starts an authentication process by transmitting his/her identity, ID, to the server.
2) Server: The server chooses a random value R and transmits it as a challenge to the user along with the corresponding signing key sk_{ID}.
3) User: The user generates a signature σ as follows:
   - Recognize his/her fuzzy data and extracts features $\overrightarrow{w}' = (w'_1, \ldots, w'_n) \in \mathbb{Z}_p^n$ from it
   - Run $\mathsf{FVS.Sign}\ (sk_{ID}, \overrightarrow{w}', R) \rightarrow \sigma$

   The user transmits a signature σ to the server.
4) Server: The server imports the verification key vk_{ID} corresponding to the user's identity ID, stored in DB. The server then verifies the signature σ (transmitted from the user) by using a challenge R and vk_{ID}.
   - Run $\mathsf{FVS.Vrfy}\ (vk_{ID}, \sigma, R) \rightarrow 1/0$

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

VII. CONCLUSION

In this paper, we propose a new primitive for fuzzy authentication, which is called fuzzy vector signature (FVS) scheme. In the FVS scheme, the verification key and signature are derived from the user's fuzzy data (such as biometrics). Even if the fuzzy data is extracted from the same user, subtle differences are likely to occur each time it is extracted. If the difference between two fuzzy data used to generate the verification key and the signature is within the threshold, the signature is successfully verified with the verification key, despite the existence of a few *outliers*. In addition, our FVS scheme is *reusable*, which means that multiple verification keys generated from the same fuzzy data can be registered in various application services. Compared to the reusable fuzzy extractor [1], the proposed FVS scheme significantly reduces network bandwidth usage (*or* the amount of storage required on the user side).

We also propose a privacy-preserving fuzzy authentication protocol based on the FVS scheme. Since our FVS scheme provides *VK privacy*, the user can publicly store his/her key pair in a (potentially malicious) server. That is, the user is

TABLE 6. The d224-224 curve.

q	15028799613985034465755506450771565229282832217860390155996483840017
n $(= r)$	15028799613985034465755506450771561352583254744125520639296541195021
a	1871224163624666631860092489128939059944978347142292177323825642096
b	979550172334338054714415200677665314930646613801273064011412560570 1
k	6
nk	1152247469502521737006260301379098033453809642945568911422202491218443231922839320465038366178186480607624725955637835054166999434487843013620271494576148838589061992555345766815850420278658055997094593665763685534671359888806751621463485933055463450576719841585715047934594472171035627404770753615629621557341276373513560095386541900039892029253521575729153930752563967520459793891950480742723873581 1520
hk	51014915936684265604900487195256160848193571244274648855332475661658304506316301006112887177277345010864012988127829655449256424871024500368597989462373813062189274150916552689262852603254011248502356739907870646157347208177913739804037628154293851397047399078706461573472 0
coef_0	11975189258259697166257037825227536931446707944682470951111859446192
coef_1	13433042200347934827742738095249546804006687562088254057411901362771
coef_2	8327464521117791238079105175448122006759863625508043495770887411614
nqr	14272136330217603734034693678007035353854159377030199293674061692 4

not required to store any information in a secure manner on the user side, and only the user's fuzzy data is the secret information required during the online authentication. In addition, since the FVS scheme provides *anonymity* and *(strong existential) unforgeability*, our fuzzy authentication protocol is secure against malicious external attackers as well as malicious servers. Therefore, the proposed fuzzy authentication protocol can be used in a variety of environments where the user's fuzzy data is used as an authentication means and also where its privacy needs to be protected.

APPENDIX. CURVE PARAMETERS FOR BILINEAR MAP

We give curve parameters of bilinear maps used for our experiments. In the tables, nk denotes the order of $E(\mathbb{F}_{q^k})$ and $hk = nk/r^2$ where $E(\mathbb{F}_{q^k})$ is the elliptic curve group over the finite filed $\mathbb{F}_{q^k}$. k is the embedding degree of $E(\mathbb{F}_q)$ and nqr is a quadratic non-residue, and coef_1 and coef_2 are used to define an irreducible polynomial. The curves are defined over the finite filed $\mathbb{F}_q$ with $y^2 = x^3 + ax + b$. The bilinear group $\mathbb{G}_1$ on this curve has order n [39].

TABLE 7. The d347-337 curve.

q	2497253197277309197004750455915457876829035078882 207313959589083032793711645453572215225471727318121 16737
n	2497253197277309197004750455915457876829035078882 207471986586690318530468547625454849612088278649529 0747198
r	484517800833765196058429300152394768597628117204 20 780970229267773588608458268668726831301066697377 41

a	123952954652503774560233700363751095608196706680 39
	413060643368826277303877113892566503688744219345 76
	18817

b	826353031016691830401558002425007304054644711202 62
	753737622458841848692514092617110024591628128971 74
	5878

k	6

nk	242535584663234070311274653505055491002480829760 28
	207656888507840309057379904888953160469190291720 21
	501592420776463208877578378516957446940340884150 50
	415576671618051539166792816455969134925647422299 12
	610243938286070449437871648400216753906939457521 57
	811368694574665224653131965727918044386221046635 32
	515615076010799724861531318739519787114916020598 88
	405241130061076238998818157348411074920783877182 24
	146521746759263388595283583925130801354107717579 61
	198047534924954265431039425575283602076418063628 05
	209566467513133140930510465995667569718162477566 94
	352813625171394116812917718682869608842244489637 17
	25968441024584341082021 3760

hk	103313242542119982791805544245137944478403376397 49
	388253135901177752220889564639690924134154612976 71
	295381802971007877049818089601757034009003304662 73
	000028114623657664127358658132938642895446871436 91
	162201428408833246805494438536225426008262992582 13
	974797998833202000573798433047915871389698479715 44
	732679240331655516528147708339066258461761495659 57
	919467927764526760216464741545691755875899964904 28
	227930466601199473619037 6960

coef_0	876268834271653048299806393569087827034783939952 70
	597031205380577293855178337986806486285094034258 68
	0019

coef_1	663525829462547541526953842406476202462169454395 31
	194153284257417385193817390876651693561980750016 08
	2297

coef_2	362508428612533157949404332781202236325747922876 09
	290133492769665886775692650592278218489495655710 44
	4419

nqr	112363121489069827117592041137747237174788621533 27
	401026659294832296311446178388396008365295044201 54
	09388

REFERENCES

[1] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith, "Reusable fuzzy extractors for low-entropy distributions," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, Apr. 2016, pp. 117–146.

[2] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2004, pp. 523–540.

[3] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, "Secure remote authentication using biometric data," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2005, pp. 147–163.

[4] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptogr. Conf.* Berlin, Germany: Springer, 2007, pp. 535–554.

[5] V. Iovino and G. Persiano, "Hidden-vector encryption with groups of prime order," in *Proc. Int. Conf. Pairing-Based Cryptogr.* Berlin, Germany: Springer, 2008, pp. 75–88.

[6] J. H. Park, K. Lee, W. Susilo, and D. H. Lee, "Fully secure hidden vector encryption under standard assumptions," *Inf. Sci.*, vol. 232, pp. 188–207, May 2013.

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

[8] A. Ross, J. Shah, and A. K. Jain, "From template to image: Reconstructing fingerprints from minutiae points," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 544–560, Apr. 2007.

[9] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 3494. Berlin, Germany: Springer, 2005, pp. 457–473.

[10] D. W. Cheung, N. Mamoulis, W. K. Wong, S. M. Yiu, and Y. Zhang, "Anonymous fuzzy identity-based encryption for similarity search," in *Proc. Int. Symp. Algorithms Comput.* Berlin, Germany: Springer, 2010, pp. 61–72.

[11] K. Takahashi, T. Matsuda, T. Murakami, G. Hanaoka, and M. Nishigaki, "A signature scheme with a fuzzy private key," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Cham, Switzerland: Springer, Jan. 2015, pp. 105–126.

[12] T. Matsuda, K. Takahashi, T. Murakami, and G. Hanaoka, "Fuzzy signatures: Relaxing requirements and a new construction," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Cham, Switzerland: Springer, Jun. 2016, pp. 97–116.

[13] 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 *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Cham, Switzerland: Springer, Dec. 2017, pp. 409–421.

[14] M. Blanton and P. Gasti, "Secure and efficient protocols for iris and fingerprint identification," in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2011, pp. 190–209.

[15] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, "New packing method in somewhat homomorphic encryption and its applications," *Secur. Commun. Netw.*, vol. 8, no. 13, pp. 2194–2213, Sep. 2015.

[16] A. Abidin, "On privacy-preserving biometric authentication," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Cham, Switzerland: Springer, Mar. 2016, pp. 169–186.

[17] M. Gomez-Barrero, E. Maiorana, J. Galbally, P. Campisi, and J. Fierrez, "Multi-biometric template protection based on homomorphic encryption," *Pattern Recognit.*, vol. 67, pp. 149–163, Jul. 2017.

[18] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, Jan. 2015, pp. 470–491.

[19] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," in *Proc. Public-Key Cryptogr.–PKC*. Berlin, Germany: Springer, Feb. 2016, pp. 164–195.

[20] F. Guo, W. Susilo, and Y. Mu, "Distance-based encryption: How to embed fuzziness in biometric-based encryption," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 247–257, Feb. 2016.

[21] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Proc. Int. Conf. Secur. Cryptogr. Netw.* Cham, Switzerland: Springer, Aug. 2018, pp. 544–562.

[22] K. Paterson, *Cryptography from Pairing-Advances in Elliptic Curve Cryptography*. London, U.K.: Cambridge Univ. Press, 2005.

[23] D. Boneh, A. Raghunathan, and G. Segev, "Function-private identity-based encryption: Hiding the function in functional encryption," in *Advances in Cryptology–CRYPTO* (Lecture Notes in Computer Science), vol. 8043. Berlin, Germany: Springer, 2013, pp. 461–478.

[24] Y. Sutcu, Q. Li, and N. Memon, "Protecting biometric templates with sketch: Theory and practice," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 3, pp. 503–512, Sep. 2007.

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

[26] S. Shekhar, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Joint sparse representation for robust multimodal biometrics recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 113–126, Jan. 2014.

[27] U. Gawande, K. Hajari, and Y. Golhar, "Efficient multimodal biometric feature fusion using block sum and minutiae techniques," in *Proc. Int. Conf. Comput. Vis. Image Process.* Singapore: Springer, Dec. 2017, pp. 215–225.

[28] C.-P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, Jan. 1991.

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

[30] E. Kiltz, D. Masny, and J. Pan, "Optimal security proofs for signatures from identification schemes," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, Jul. 2016, pp. 33–61.

[31] N. Bitansky and R. Canetti, "On strong simulation and composable point obfuscation," in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2010, pp. 520–537.

[32] J. H. Cheon, J. Jeong, D. Kim, and J. Lee, "A reusable fuzzy extractor with practical storage size: Modifying Canetti et al.'s construction," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, Jun. 2018, pp. 28–44.

[33] X. Boyen, "A tapestry of identity-based encryption: Practical frameworks compared," *Int. J. Appl. Cryptogr.*, vol. 1, no. 1, pp. 3–21, Feb. 2008.

[34] D. Apon, C. Cho, K. Eldefrawy, and J. Katz, "Efficient, reusable fuzzy extractors from LWE," in *Proc. Int. Conf. Cyber Secur. Cryptogr. Mach. Learn.* Cham, Switzerland: Springer, Jun. 2017, pp. 1–18.

[35] Y. Wen, S. Liu, and S. Han, "Reusable fuzzy extractor from the decisional Diffie–Hellman assumption," in *Proc. Designs, Codes Cryptogr.*, Jan. 2018, pp. 1–18.

[36] Y. Wen and S. Liu, "Reusable fuzzy extractor from LWE," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, Jun. 2018, pp. 13–27.

[37] Z. Jin, J. Y. Hwang, Y.-L. Lai, S. Kim, and A. B. J. Teoh, "Ranking-based locality sensitive hashing-enabled cancelable biometrics: Index-of-max hashing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 2, pp. 393–407, Feb. 2018.

[38] S. E. Java. *ExecutorService*. Accessed: Mar. 15, 2019. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ExecutorService.html

[39] B. Lynn. (2013). *PBC Library—The Pairing-Based Cryptography Library, Version 0.5.14.2013*. [Online]. Available: https://crypto.stanford.edu/pbc

[40] A. De Caro and V. Iovino, "JPBC: Java pairing based cryptography," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun./Jul. 2011, pp. 850–855.

[41] T. Granlund. (2016). *The GNU Multiple Precision Arithmetic Library, Verison 6.1.2.2016*. [Online]. Available: https://gmplib.org

[42] 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," *Comput. Vis. Image Understand.*, vol. 117, no. 10, pp. 1512–1525, Oct. 2013.

MINHYE SEO received the B.S. degree from the Department of Mathematics, Korea University, Seoul, South Korea, in 2012, where she is currently pursuing the Ph.D. degree in information security with the Graduate School of Information Security. Her research interests include applied cryptography, authentication, information security, functional encryption, and lattice-based cryptography.

JUNG YEON HWANG received the B.S. degree in mathematics, and the M.S. and Ph.D. degrees in information security from Korea University, Seoul, in 1999, 2003, and 2006, respectively. He was a Visiting Researcher with Columbia University, New York, from 2005 to 2006, and a Postdoctoral Researcher with Korea University, from 2006 to 2009. He is currently a Principal Researcher with the Electronics and Telecommunications Research Institute, Daejeon, South Korea. He has developed broadcast encryption schemes and anonymous signature schemes with opening and linking capability. His research interests include biometrics, fuzzy cryptosystems, identity management, privacy-enhancing technology, and cryptographic protocols and their applications.

DONG HOON LEE received the B.S. degree from the Department of Economics, Korea University, Seoul, in 1985, and the M.S. and Ph.D. degrees in computer science from The University of Oklahoma, Norman, in 1988 and 1992, respectively, where he is currently a Professor with the Graduate School of Information Security, Korea University, where he has been with the Faculty of Computer Science and Information Security, since 1993. His research interests include cryptographic protocols, applied cryptography, functional encryption, software protection, mobile security, vehicle security, and ubiquitous sensor network (USN) security.

SOOHYUNG KIM received the B.S. and M.S. degrees in computer science from Yonsei University, Seoul, South Korea, in 1996 and 1998, respectively, and the Ph.D. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2016. He is currently the Project Leader of the Information Security Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His research interests include biometrics, identity management, blockchain security, and financial security.

SEUNG-HYUN KIM was born in Daegu, South Korea, in 1979. He received the B.S. degree in computer engineering from the Kumoh National University of Technology, South Korea, in 2002, and the M.S. degree in computer engineering from POSTECH, South Korea, in 2004, and the Ph.D. degree in computer science from KAIST, South Korea, in 2017. Since 2004, he has been a Member of the Research Staff with the Electronics and Telecommunications Research Institute, Daejeon, South Korea. His main research interests include ID management, mobile security, mobile privacy, and personalization service.

JONG HWAN PARK received the B.S. degree from the Department of Mathematics, Korea University, Seoul, South Korea, in 1999, and the M.S. and Ph.D. degrees with the Graduate School of Information Security, Korea University, Seoul, South Korea, in 2004 and 2008, respectively. From 2009 to 2011, he was a Research Professor with Kyung Hee University, and from 2011 to 2013 he was a Research Professor with Korea University. Since 2013, he has been an Assistant Professor with the Department of Computer Science, Sangmyung University, Seoul, South Korea. His research interests include functional encryption, broadcast encryption, authenticated encryption, and various cryptographic protocols.