Lattice-Based Remote User Authentication from Reusable Fuzzy Signature

[1]Yangguang Tian, [1]Yingjiu Li, [1]Robert. H Deng, [1]Binanda Sengupta and [2]Guomin Yang

[1]School of Information Systems, Singapore Management University, Singapore.
[2]School of Computing and Information Technology
University of Wollongong, Australia
[1]{ygtian,yjli,robertdeng,binandas}@smu.edu.sg
[2]gyang@uow.edu.au

Abstract. In this paper, we introduce a new construction of lattice-based reusable fuzzy signature for remote user authentication that is secure against quantum computers. We define formal security models for the proposed construction, and we prove that it can achieve user authenticity, biometrics reusability and user privacy. In particular, the proposed new construction ensures that: 1) biometrics reusability is achieved such that fuzzy signatures remain secure even when the same biometrics is reused multiple times; 2) a third party having access to the communication channel between an authorized user and the authentication server cannot identify the authorized user.

Keywords: Lattice-Based Cryptography, Fuzzy Signatures, Biometrics Reusability, User Privacy

1 Introduction

Fuzzy signature (FS), also known as signature with a *fuzzy* secret key [32, 26], is a new type of digital signature that allows a signature to be generated using secret biometrics as a signing key, without relying on any additional data (e.g., the helper data as used in fuzzy extractors [12]). FS can further be explored in a reusable setting: reusable fuzzy signature (RFS), which deals with biometrics reuse. Specifically, a user may use the same biometrics to derive different signing/verification key pairs for user authentications. That is, the RFS may run many times on various messages under different signing keys, but with noisy versions of the same biometrics. Reusability allows the security of generated signature in a user authentication to remain secure even when the generated signatures in all other authentications are compromised (e.g., the signing keys may be leaked to attackers). Therefore, biometrics reusability is an essential security property required in RFS, but there is no such security guarantee in [32, 26].

The RFS is significantly useful in many real-world applications, such as biometric-based remote user authentication [9, 10, 4, 24]. Let us consider a setting where user Alice wishes to remotely authenticate herself to server Bob using her biometrics. A traditional method is to use digital signatures with fuzzy extractors [9, 10, 24]. That is, Alice relies on her enrolled helper data stored somewhere (e.g., server, smart card and USB token) to derive a signing key for generating a signature on a message (e.g., nonce), then Bob verifies the message-signature pair under Alice's enrolled verification key. In RFS, Alice's signature additionally includes a new verification key (her signature is generated by a new signing key) and a new helper data, where the new helper data is derived from the new signing key and her noisy biometrics. Bob verifies the message-signature pair under Alice's new verification key and *checks* whether the new verification key is related to Alice's enrolled verification key. This checking process relies on Alice's new as well as enrolled helper data, so a valid message-signature pair requires that Alice's noisy biometrics and her enrolled biometrics are close enough. A crucial point to stress is that, Alice's signature generation does not depend on her enrolled helper data, this is the key difference between digital signatures with fuzzy extractors [9, 10, 24] and RFS.

The security and privacy of RFS used in a remote user authentication are also essential. First, lattice-based cryptography resistance to quantum computers has been extensively studied in the literature [17, 29, 27, 28]. We note that a remote user authentication that exploits prior RFS [32, 26] is not secure since a quantum computer can efficiently solve some hard mathematical problems such as discrete logarithm (DL) problem [31]. So the DL-based fuzzy signature schemes proposed in [32, 26] are not desired in the post-quantum era. Second, user's privacy is compromised if we use fuzzy signatures [32, 26] to design a remote user authentication. Specifically, multiple message-signature pairs from the same user Alice can be easily linked by a third party eavesdropping the communication channel, since the fuzzy signatures [32, 26] are publicly verifiable (i.e., anyone with Alice's enrolled verification key can verify Alice's message-signature pairs). Therefore, the main goal of this work is to design a reusable fuzzy signature based remote user authentication (RFS-RUA) that satisfies *resistance to quantum computers, biometrics reusability and privacy guarantee against eavesdroppers.*

Technical Challenges. It is a non-trivial task to design a lattice-based RFS for RUA that is secure against quantum computers. We consider to use the lattice-based digital signatures without trapdoors [25, 14, 3], because this line of research supports a simple and efficient signing process [25] compared to lattice-based trapdoor signatures [17]. However, in the design of lattice-based signatures without trapdoors, the signature generation is independent of the signing key. This mechanism obviously contradicts to the *homomorphic* property required in RFS, which means that the difference ("shift") between two signatures is identical to the difference between two signing/verification keys [32, 26]. Such homomorphic property allows Bob to find the correct difference between Alice's new verification key and her enrolled verification key, so Alice's message-signature pair can be confirmed as valid, because she is the only party who can produce the correct difference between her new signing key and her enrolled signing key.

Our Contributions. The main contributions of this work are summarized as follows.

– *New Construction.* We propose a new construction of remote user authentication that is built on top of lattice-based digital signatures [25, 14, 3], reusable fuzzy extractors from learning with errors (LWE) [2, 35], a family of universal hash functions and lattice-based public key encryptions. The new construction in this work covers any lattice-based cryptographic primitives that include digital signatures and public key encryptions.
– *Quantum Resistant.* The proposed construction can withstand quantum computers due to the lattice-based cryptographic primitives we used. Its provable security relies on the worst-case intractability of standard lattice problems.
– *Biometrics Reusability.* We formulate the security definition of RFS. We prove that the lattice-based RFS can achieve biometrics reusability since the underlying LWE-based fuzzy extractors are reusable. In particular, the construction of RFS is the first cryptographic primitive with Hamming distance.
– *Privacy Protection.* We show a user privacy model to prevent the eavesdroppers from identifying any authorized user or linking any authorized user's multiple sessions. We prove that the proposed construction can achieve user privacy under standard assumptions.

Overview of Techniques. We now explain our key technical insights. First, we characterize the homomorphic property of lattice signatures without trapdoors [25, 14, 3] in order to ensure the correctness of RFS. We discover that the resulting lattice signatures depend on their chosen *randomness* whose (discrete Gaussian) distributions are always centered at zero. As a result, the "shift" between signing keys has no effect on lattice signatures, because the lattice signatures obtained using "shifted" signing keys are actually identical when their randomness come from the same distribution. Meanwhile, the distributions of randomness are also related to the distributions of signing keys, so the lattice signatures can be successfully verified under the corresponding "shifted" verification keys. Second, putting all building blocks together, which include lattice signatures [25, 14, 3], reusable fuzzy signatures [2, 35] and a family of universal hash functions, we can construct a lattice-based RFS that achieves biometrics reusability. Third, we use the lattice-based public key encryptions to ensure privacy guarantee, where the validity of lattice-based RFS is verified by authentication server only. Specifically, Alice's identity is encrypted under the public key of Bob. After extracting Alice's encrypted identity, Bob can identify her enrolled verification key and helper data. Then he verifies Alice's lattice-based RFS under a new verification key which is derived from enrolled verification key, enrolled and new helper data.

1.1 Related Work

Fuzzy Signatures. The concept of fuzzy signature was first introduced by Takahashi et al. [32], which is a signature scheme that inputs fuzzy data such as biometrics as a signing key. Specifically, the generation of a signature does not rely on auxiliary data such as helper data (or sketch). The proposed generic construction is built on top of a signature scheme with *homomorphic* (see Section

3.4) properties regarding keys and signatures, and a *linear* sketch. It is proven secure in the standard model. To relax some requirements on the building blocks used in the generic construction of [26], Matsuda et al. have proposed a new generic construction using some relaxed building blocks. For example, Waters signature scheme [34] is replaced by Schnorr signature scheme [30].

The input biometrics of linear sketch in [32] is assumed to be *uniformly* distributed over metric space. To relax such strong assumption on fuzzy data, Matsuda et al. [26] require only *high min-entropy* on the distribution of biometrics. Specifically, they consider linear sketches as real numbers that include integer and decimal (i.e., secret key and biometrics) parts. That is, if the noise between two linear sketches is less than a threshold t (a positive real number), then a difference algorithm (i.e., $\mathsf{DiffRec}$) can extract the correct difference (may include noise) between two linear sketches.

Yasuda et al. [37] introduced the "recovering attacks" to recover both the secret key and the biometrics from *linear sketch*. They claim that the "integer plus decimal" format is vulnerable to such attacks. In addition, they provided a trivial countermeasure (add small noisy data to the sketch) with informal security analysis. Meanwhile, Takahashi et al. [33] (merged version of [32, 26]) also provided the treatment to avoid the "recovering attacks". The remedy is to add a "rounding-down" operation (or truncation) on the decimal part of the real numbers. However, such extra truncation would bring a correctness loss to the proposed constructions in [32, 26].

Instead of using real numbers (with "integer plus decimal" format) to represent and process fuzzy data, in this work, we take binary strings over Hamming distance as fuzzy data input such as Iriscode [20]. We notice that constructing fuzzy signatures from Hamming distacne was an open problem, which was pointed out by prior works [26, 33].

Lattice Signatures. A lattice-based signature scheme with provable security was first constructed by Gentry et al [17]. Their construction is based on "hash-and-sign" paradigm, and its security relies on the *worst-case* hardness of standard lattice problems such as small integer solution (SIS). To achieve more practical and efficient constructions, another line of research on lattice-based digital signatures relies on the Fiat-Shamir heuristic [15] such as [25, 14, 3]. Their constructions are Schnorr-like identification protocols whose security is based on SIS or LWE, and the efficiency relies on a *rejection sampling* (see Section 3.4) rather than the *pre-image sampling* used in [17]. In this work, we use lattice-based signature schemes [25, 14, 3] as our building blocks.

Reusable Fuzzy Extractor. Fuzzy extractor (FE) is one of the promising approaches to construct a biometric-based remote user authentication [9, 10, 24]. Juels and Wattenberg [22] introduced a cryptography primitive called "fuzzy commitment". It is particularly useful for biometric-based remote authentication systems, because its error-correcting technique can correct certain errors within a suitable metric (Hamming distance). Dodis et al. [12] formally introduced the notions of "secure sketches" and "fuzzy extractors". In particular, they provided concrete constructions of secure sketches and fuzzy extractors in three metrics

(Hamming distance, set difference and edit distance), and the constructions are information-theoretically secure.

Boyen [9] introduced an important notion: reusable fuzzy extractor. It states that a user can produce multiple key and sketch pairs using the same biometrics w, i.e., $\{(R_i, P_i)\} \leftarrow \mathsf{Gen}(w)$. Later, Canetti et al. [11] proposed the first reusable FE from some low-entropy distributions. In particular, they have refined the security of *strongly* reusable FE such that each R_i remains secure even if all other secret keys R_j $(j \neq i)$ are revealed.

Computational FE was first introduced by Fuller et al. [16]. The proposed scheme is based on LWE [29], and the derived secret key equals to the entropy of the fuzzy biometrics. However, their computational FE is not reusable. To achieve reusable FE from LWE, Apon et al. [2] provided a generic transformation to convert non-reusable (resp. *weak* reusable) fuzzy extractors to *weak* reusable (resp. *strong* reusable) ones. In addition, based on the definitions of reusability [9, 11], Apon et al. formalized both *weak* and *strong* reusability. Recently, Wen and Liu [35] proposed a new reusable (and robust [10]) FE. Its reusability also follows the *strong* reusability defined in [2]. In this work, we follow the *strong* reusability defined in [2, 35], where the "shifts" between different readings are controlled by the adversary (i.e., the perturbation attacks defined in [9]).

To highlight our distinction, we show the function (feature) difference between our proposed new construction and some existing works in Table 1: it shows that our proposed construction has quantum resistance, user authenticity, biometrics reusability and user privacy. The new construction is proven secure in the random oracle model. We stress that the proposed RFS can be regarded as a step forward from fuzzy signatures [32, 26] and fuzzy extractors [2, 35] in the lattice-based setting.

Table 1. The comparison between different functionalities of lattice-based signature (Sig), fuzzy extractor (FE) and fuzzy signature (FS) schemes. Reusability shows whether the biometrics reuse is formally addressed or not. User privacy means user's privacy concern with respect to the eavesdroppers (see Appendix A). N/A means that the scheme did not consider this functionality.

Functionality/Scheme	[25]	[16]	[32]	[26]	[2]	[35]	Ours
Sig/FS/FE	Sig	FE	FS	FS	FE	FE	FS
Reusability	N/A	$\times$	N/A	N/A	✓	✓	✓
User Privacy	$\times$	N/A	$\times$	$\times$	N/A	N/A	✓
Lattice Based	✓	✓	$\times$	$\times$	✓	✓	✓
Standard Model	$\times$	✓	✓	$\times$	✓	✓	$\times$

1.2 Paper Organization

In the next section, we present the formal security models to capture the security requirements of a RFS based remote user authentication. In Section 3, we present some preliminaries which will be used in our proposed construction. In Section 4, we present the notion of RFS and prove its reusability. We present our detailed construction in Section 5 and formally prove its security. The paper is concluded in Section 6.

2 Security Model

In this section, we present the security models (user authenticity and user privacy) for our proposed remote user authentication construction based on RFS (RFS-RUA).

States. We define a system entity set $\mathcal{U}$ with N users and M servers. We say an instance oracle Π^l_{ID} (e.g., session l of user ID) may be *used* or *unused*, and a user ID has unlimited number of instances called *oracles*. The oracle is considered as unused if it has never been initialized. Each unused oracle Π^l_{ID} can be initialized with a secret key $\mathtt{sk}$. The oracle is initialized as soon as it becomes part of a protocol execution. After the initialization the oracle is marked as used and turns into the *stand-by* state where it waits for an invocation to execute a protocol operation. Upon receiving such invocation the oracle Π^l_{ID} learns its partner id pid^l_{ID} and turns into a *processing* state where it sends, receives and processes messages according to the description of the protocol. During that stage, the internal state information $state^l_{ID}$ is maintained by the oracle. The oracle Π^l_{ID} remains in the processing state until it collects enough information to finalize the user authentication procedure. As soon as the user authentication is accomplished, Π^l_{ID} *accepts* and *terminates* the protocol execution in the sense that it would not send or receive further messages. If the protocol execution fails, then Π^l_{ID} terminates without having accepted.

We denote the l-th session established by a server as $\Pi^l_{ID_{\mathbb{S}}}$ and identities of all the users recognized by $\Pi^l_{ID_{\mathbb{S}}}$ during the execution of that session by partner identifier $\mathsf{pid}^l_{ID_{\mathbb{S}}}$. We define $\mathsf{sid}^l_{ID_{\mathbb{S}}}$ as the unique session identifier belonging to the session l established by a server $ID_{\mathbb{S}}$. Specifically, we have $\mathsf{sid}^l_{ID_{\mathbb{S}}} = \{msg_i\}^n_{i=1}$, where $msg_i \in \{0,1\}^*$ is the message transcript transmitted between users and servers.

2.1 Definition

A RFS-RUA consists of the following algorithms:

- Setup: The algorithm takes security parameter λ as input, outputs a public parameter pp.
- KeyGen: The algorithm takes public parameter pp as input, outputs secret/public key pairs $\{(\mathtt{sk}, \mathtt{pk})\}$ for users, and secret/public key pairs $\{(ssk, spk)\}$ for authentication servers.
- $\mathsf{Enrollment}$. This is a non-interactive protocol between a user and an authentication server over a secure channel. The user enrolls her identity ID, public key $\mathtt{pk}$ and sketch $\mathsf{SS}(w, \mathtt{sk})$ to the authentication server. The enrolled users

become authorized ones after enrollment, and the sketch $\mathsf{SS}(w, \mathtt{sk})$ means that it derives from a biometrics w and a secret key $\mathtt{sk}$. We assume a *uniform*[1] biometrics source $\mathcal{M}$ and $w \in \mathcal{M}$.

- Authentication. This is an interactive protocol between an authorized user and an authentication server over a public channel. The protocol takes public parameter pp, an authorized user's identity ID and biometrics w', and an authentication server's public key spk as input, outputs a new secret/public key pair $(\mathtt{sk}', \mathtt{pk}')$, a new sketch $\mathsf{SS}(w', \mathtt{sk}')$ and a message-signature pair (msg, σ). The authentication server *accepts* the user if: 1) the message-signature pair (msg, σ) is verified as valid under her new public key $\mathtt{pk}'$; and 2) the new public key $\mathtt{pk}'$ is linked to new sketch $\mathsf{SS}(w', \mathtt{sk}')$, and her enrolled public key $\mathtt{pk}$ and sketch $\mathsf{SS}(w, \mathtt{sk})$. The message msg denotes the transmitted ephemeral data, and the biometrics satisfies $\mathsf{dist}(w', w) \le t$.

2.2 User Authenticity

We define Δ as the set of functions $f : \mathcal{M} \to \mathcal{M}$ satisfies: for any $w \in \mathcal{M}$, $f(w)$ is "close" to w (i.e., $\mathsf{dist}(w, f(w)) \le t$). The perturbations $\delta \in \Delta$ are specified by the adversary and applied by the simulator (i.e., challenger). Let $\mathsf{dist}(w) = \mathsf{dist}(w, 0)$ denote the Hamming weight of distribution $w \in \mathcal{M}$. Furthermore, the simulator specifies a class of functions $\Phi = \{\phi\}^{|\mathtt{sk}|}$ whose domain and range are the secret key space of RFS-RUA. In the user authenticity game, an adversary attempts to impersonate an authorized user and authenticate herself to an authentication server. We define a formal authenticity game between a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ as follows.

- Setup. $\mathcal{S}$ first generates identities $\{ID_{\mathbb{S}_j}\}$ and secret/public key pair $(ssk_j, sspk_j)$ $(j \in [1, M])$ for M servers, and identities $\{ID_i\}$ $(i \in [1, N])$ for N users in the system. $\mathcal{S}$ also generates user's biometrics information $\{w_i\}$ $(w_i \in \mathcal{M})$, generates a set of public/secret key pairs $\{\mathtt{pk}_i^j, \mathtt{sk}_i^j\}_{j=1}^{M}$ and their corresponding sketches $\mathsf{SS}(w_i, \mathtt{sk}_i^j)$ for each user i with respect to M servers. Eventually, $\mathcal{S}$ sends all identities, public keys and sketches to $\mathcal{A}$. Let KG be a key generation algorithm which takes public parameter pp and a secret key $\mathtt{sk}$ as input, outputs a public key $\mathtt{pk}$.
- Training. $\mathcal{A}$ can make the following queries in an arbitrary sequence to $\mathcal{S}$.
  - Send: If $\mathcal{A}$ issues a send query in the form of (ID, l, msg) to simulate a network message for the l-th session of user ID, then $\mathcal{S}$ would simulate the reaction of instance oracle Π_{ID}^{l} upon receiving message msg, and return to $\mathcal{A}$ the response that Π_{ID}^{l} would generate; If $\mathcal{A}$ issues a send query in the form of (ID', '*start*'), then $\mathcal{S}$ creates a new instance oracle $\Pi_{ID'}^{l}$ and returns to $\mathcal{A}$ the first protocol message.

[1] One may question a uniform source is not practical, we stress that the uniform source can be replaced by a non-uniform source (e.g., symbol-fixing source [23]) while the security of FE is held. We use a uniform source here just for simplicity, and the case of non-uniform source was explicitly discussed in [16, 36, 35].

  - Biometrics Reveal: If $\mathcal{A}$ issues a biometrics reveal query to user i, then $\mathcal{S}$ returns user i's biometric information w_i to $\mathcal{A}$.
  - Secret Key Reveal: If $\mathcal{A}$ issues a secret key reveal query to user i with respect to server $\mathbb{S}_j$, then $\mathcal{S}$ returns the enrolled secret key $\mathtt{sk}_i^j$ to $\mathcal{A}$.
  - Biometrics Shift: If $\mathcal{A}$ issues a biometrics shift query with δ to user i, then $\mathcal{S}$ returns user i's *shifted* biometric information $\mathsf{SS}(w_i + \delta, \mathtt{sk}_i^j)$ to $\mathcal{A}$.
  - Secret Key Shift: If $\mathcal{A}$ issues a secret key shift query with ϕ to user i, then $\mathcal{S}$ returns user i's *shifted* public key $\mathtt{pk}_i' \leftarrow \mathsf{KG}(pp, \phi(\mathtt{sk}_i^j))$ to $\mathcal{A}$.
  - Server Secret Key Reveal: If $\mathcal{A}$ issues a server secret key reveal query to server $ID_{\mathbb{S}_j}$, then $\mathcal{S}$ will return server $\mathbb{S}_j$'s secret key ssk_j to $\mathcal{A}$.
- Challenge. $\mathcal{A}$ outputs a message msg and wins the game if all of the following conditions hold.
  1. $ID_{\mathbb{S}_j}$ *accepts* user i. It implies $\mathsf{pid}_{\mathbb{S}_j}^s$ and $\mathsf{sid}_{\mathbb{S}_j}^s$ exist.
  2. $\mathcal{A}$ did *not* issue Biometrics Reveal query to user i.
  3. $\mathcal{A}$ did *not* issue Secret Key Reveal query to user i with respect to $\mathbb{S}_j$.
  4. $msg \in \mathsf{sid}_{\mathbb{S}_j}^s$, *but* there exists *no* instance oracle $\varPi_{ID_i}^s$ which has sent msg (msg denotes the *message transcript* from user i).

  $\mathcal{A}$ is allowed to reveal user i's secret keys associate with M-1 servers. We also allow $\mathcal{A}$ to adaptively issue Biometrics Shift queries to challenge user i. We define the advantage of an adversary $\mathcal{A}$ in the above game as

$$\mathtt{Adv}_{\mathcal{A}}^{\text{RFS-RUA}}(\lambda) = |\Pr[\mathcal{A}\ wins]|.$$

Definition 1. *We say that a RFS-RUA has user authenticity if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}^{RFS\text{-}RUA}(\lambda)$ is a negligible function of the security parameter λ.*

Design Rationale. The perturbation $\delta \in \varDelta$ from an adversary is to capture the adaptive chosen perturbation attacks [9]. For example, an adversary may query an oracle to perform extractions and re-generations based on the chosen perturbations of the secret biometrics. The function $\varPhi = \{\phi\}^{|\mathsf{sk}|}$ from simulator is to capture the related key attacks such that an adversary may "modify" the secret (or signing) keys [7]. In particular, the simulator should "update" the simulated signatures under the new public (or verification) keys [26].

2.3 User Privacy

Informally, an adversary (i.e., outsiders) attempts to identify the authenticated users involved in a RFS-RUA. We define a formal user privacy game between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ as follows:

- Setup: $\mathcal{S}$ first generates identities $\{ID_{\mathbb{S}_j}\}$ and secret/public key pair (ssk_j, spk_j) ($j \in [1, M]$) for M servers, and identities $\{ID_i\}$ ($i \in [1, N]$) for N users in the system. $\mathcal{S}$ also generates biometrics information $\{w_i\}$ for N users, generates a set of public/secret key pairs $\{\mathtt{pk}_i^j, \mathtt{sk}_i^j\}_{j=1}^M$ and their corresponding sketches $\mathsf{SS}(w_i, \mathtt{sk}_i^j)$ with respect to M servers. Eventually, $\mathcal{S}$ sends all identities, public keys and sketches to $\mathcal{A}$. $\mathcal{S}$ also tosses a random coin b which will be used later in the game.

- Training: $\mathcal{A}$ is allowed to issue Execute queries to $\mathcal{S}$. In addition, $\mathcal{A}$ can issue at most N-2 Biometrics Reveal, N-2 Secret Key Reveal queries, and M-1 Server Secret Key Reveal queries to $\mathcal{S}$. We denote the *honest* (i.e., uncorrupted) user and server set as $(\mathcal{U}'_0, \mathcal{U}'_1)$, respectively. The *honest* user requires that her biometrics as well as secret key are not corrupted.
  The Execute query [8] allows $\mathcal{A}$ to invoke an honest execution of the protocol and returns a transcript of exchanged messages to $\mathcal{A}$.
- Challenge: $\mathcal{S}$ randomly selects two users $ID_i, ID_j \in \mathcal{U}'_0$ as challenge candidates. $\mathcal{S}$ removes them from $\mathcal{U}'_0$ and simulates ID^*_b by either $ID^*_b = ID_i$ if $b = 1$ or $ID^*_b = ID_j$ if $b = 0$. $\mathcal{S}$ also selects a server $ID_{\mathbb{S}} \in \mathcal{U}'_1$ at random, and let $ID_{\mathbb{S}}$ interact with the challenge user ID^*_b. $\mathcal{A}$ can access all the communication transcripts among them.

$$ID_{\mathbb{S}} \leftrightarrow ID^*_b = \begin{cases} ID_i & b = 1 \\ ID_j & b = 0 \end{cases}$$

  $\mathcal{A}$ is allowed to issue Secret Key Shift queries to challenge candidates. Finally, $\mathcal{A}$ outputs b' as its guess for b. If $b' = b$, then $\mathcal{S}$ outputs 1; Otherwise, $\mathcal{S}$ outputs 0. We define the advantage of an adversary $\mathcal{A}$ in the above game as

$$\mathtt{Adv}^{\text{RFS-RUA}}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{S} \rightarrow 1] - 1/2|.$$

Definition 2. *We say that a RFS-RUA has user privacy if for any PPT $\mathcal{A}$, $\mathtt{Adv}^{RFS\text{-}RUA}_{\mathcal{A}}(\lambda)$ is a negligible function of the security parameter λ.*

3 Preliminaries

3.1 Complexity Assumptions

Definition 3 (SIS$_{q,n,m,d}$ Distribution). *Choose a random matrix $\mathbb{A} \in \mathbb{Z}_q^{n\times m}$ and a vector $X \in \{-d, \cdots, 0, \cdots, d\}^m$ and output $(\mathbb{A}, \mathbb{A}\cdot X)$. Note that d denotes the absolute value of random integers.*

Definition 4 (Decisional SIS [25]). *Given a pair $(\mathbb{A}, t)$ such that $t = \mathbb{A} \cdot X$, with non-negligible probability, an PPT adversary is to decide whether the pair is from SIS$_{q,n,m,d}$ distribution $(\mathbb{A}, t)$ or whether it is uniformly generated from a random distribution $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$.*

If $d \gg q^{n/m}$ (high-density), then the SIS$_{q,n,m,d}$ distribution is *statistically* close to a uniform distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ and there are many solutions X such that $\mathbb{A} \cdot X = t$. If $d \ll q^{n/m}$ (low-density), then there is only one solution X.

Definition 5 (Decisional LWE [29]). *Given a random matrix $\mathbb{A} \in \mathbb{Z}_q^{m\times n}$, $X \in \mathbb{Z}_q^n$ and χ be an arbitrary distribution on $\mathbb{Z}_q^m$, the decisional LWE (D-LWE$_{q,n,m,\chi}$) problem is to distinguish the distribution $(\mathbb{A}, \mathbb{A} \cdot X + \chi)$ from a random distribution over $(\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$. We say that D-LWE$_{q,n,m,\chi}$ is (ϵ, s_{sec}) secure if no PPT distinguisher $\mathcal{D}_{s_{sec}}$ of size s_{sec} can distinguish the LWE instances from uniform except with probability ϵ, where $s_{sec} = poly(\lambda)$ and ϵ is a negligible function of the security parameter λ.*

Dottling and Muller-Quade [13] showed that one can encode biometrics w as the error term in a LWE problem by splitting it into m blocks. Furthermore, to extract the pseudorandom bits, we rely on the result from Akavia et al. [1] such that $X \in \mathbb{Z}_q^n$ has simultaneously many hardcore bits.

Lemma 1. *If* D-LWE$_{q,n-k,m,\chi}$ *is* (ϵ, s_{sec}) *secure, then*

$$\delta^{\mathcal{D}_{s'_{sec}}}((X_{1,\cdots,k}, \mathbb{A}, \mathbb{A}\cdot X + \chi), (U, \mathbb{A}, \mathbb{A}\cdot X + \chi)) \leq \epsilon,$$

where $U \in \mathbb{Z}_q^k$ *and* $X_{1,\cdots,k}$ *denotes the first* k *coordinates of* x *and* $s'_{sec} \approx s_{sec} - n^3$.

3.2 Universal Hash Function

Let $\mathbb{H}$ be a universal hash function family whose domain is $\mathbb{Z}_{q^m}$ and whose range is $\mathbb{Z}_{q^n}$. Let $\mathbb{Z}_{q^n}$ be a vector space, which consists of n dimensional of finite field with prime order q. We define an isomorphism $\psi : (\mathbb{Z}_q)^m \to \mathbb{Z}_{q^m}$ (ψ^{-1} is its inverse), and $n, m \in \mathbb{N}$. Note that $(\mathbb{Z}_q)^m = \mathbb{Z}_q^m$.

A family of universal hash functions is defined as $\mathbb{H} = \{\mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{m\times k} | \mathbb{A}_1 \in \mathbb{Z}_{q^{mk}}, \mathbb{A}_2 \in \mathbb{Z}_q^{mk\times n}\}$, where k is a positive constant. Specifically, for each vector $\mathbb{A}_1$ (resp. matrix $\mathbb{A}_2$) in the seed space $Z_1 \in \mathbb{Z}_{q^{mk}}$ (resp. $Z_2 \in \mathbb{Z}_q^{mk\times n}$), define the hash function $\mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}$ as follows: on *input* $x \in \mathbb{Z}_q^n$, $\mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(x)$ computes $y \leftarrow \mathbb{A}_1 \cdot \psi(\mathbb{A}_2 \cdot x)$, where "$\cdot$" denotes the multiplication in the extension vector $\mathbb{Z}_{q^{mk}}$. *Notice that we can interpret a vector in* $\mathbb{Z}_q^{mk}$ *as a matrix in* $\mathbb{Z}_q^{m\times k}$. That is, $\mathbb{A}_2 \cdot x = \mathbb{Z}_q^{mk\times n} \cdot \mathbb{Z}_q^n = \mathbb{Z}_q^{mk}$, which shows that a vector in $\mathbb{Z}_q^{mk}$ is a matrix in $\mathbb{Z}_q^{m\times k}$. The *output* of $\mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(x)$ is $y \in \mathbb{Z}_q^{m\times k}$. The universal hash function family $\mathbb{H}$ consists of the hash functions $\{\mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}\}_{\mathbb{A}_1\in Z_1, \mathbb{A}_2 \in Z_2}$. Furthermore, the universal hash function family $\mathbb{H}$ has *linearity*, i.e.,

$$\forall x, x' \in \mathbb{Z}_q^n : \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(x) + \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(x') = \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(x + x').$$

Leftover Hash Lemma (LHL). The LHL [19] states that universal hash functions are good randomness extractors. Below we recall the generalized LHL (Lemma 2.4 [12]), which is defined in terms of conditional min-entropy.

Lemma 2. *Assume a family of functions* $\{\mathtt{H}_z : \mathbb{Z}_q^n \to \mathbb{Z}_q^{m\times k}\}_{z\in Z}$ *is universal, for any random variable* W *taking values in* $\mathbb{Z}_q^n$ *and any random variable* Y,

$$\mathbf{SD}((U_Z, \underline{\mathtt{H}_z(W)}, Y), (U_Z, \underline{U}, Y)) \leq \frac{1}{2}\sqrt{2^{-\tilde{\mathbf{H}}_\infty(W|Y)} \cdot q^{m\times k}},$$

where U_Z *and* U *are uniformly distributed over* $\mathbb{Z}_q^{mk}$ *(i.e.,* Z*) and* $\mathbb{Z}_q^{m\times k}$ *respectively. In particular, such universal hash functions are (average-case, strong) extractors with* ϵ*-statistically close to uniform.*

Remark. If a family of functions $\mathbb{H} = \{\mathtt{H}_{(z_1,z_2)} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{m\times k}\}$ has two independent seed spaces $z_1 \in Z_1$ and $z_2 \in Z_2$, then the statistical distance between two distributions is shown as

$$\mathbf{SD}((U_{Z_1}, U_{Z_2}, \underline{\mathtt{H}_z(W)}, Y), (U_{Z_1}, U_{Z_2}, \underline{U}, Y)) \leq \frac{1}{2}\sqrt{2^{-\tilde{\mathbf{H}}_\infty(W|Y)} \cdot q^{m\times k}},$$

where U_{Z_1} and U_{Z_2} are uniformly distributed over $\mathbb{Z}_q^{mk}$ and $\mathbb{Z}_q^{mk\times n}$ respectively. Observe that the statistical distance between two distributions is not affected by the two public seeds, so a family of functions $\mathbb{H}$ with *two seeds* is also (average-case, strong) extractors with ϵ-statistically close to uniform.

3.3 Computational Fuzzy Extractors

Let $m \geq n$ and q be a prime number, define two algorithms Gen, Rep of computational FE [16] below.

```
1. Input: $w \leftarrow \mathcal{M}$. $\triangleright$ suppose $\mathcal{M}$ is a uniform distribution over $\mathbb{Z}_q^m$.
2. Sample $\mathbb{A} \in \mathbb{Z}_q^{m\times n}$, $x \in \mathbb{Z}_q^n$ uniformly.
3. Compute $p = (\mathbb{A}, \mathbb{A}\cdot x + w), r = x_{1,\cdots,n/2}$.
4. Output: $(r, p)$.
```

```
1. Input: $w', p$. $\triangleright$ where Hamming distance between $w'$ and $w$ is at most $t$.
2. Parse $p$ as $(\mathbb{A}, c)$; let $b = c - w'$.
3. Let $x = \mathsf{Decode}_t(\mathbb{A}, b)$.
4. Output: $r = x_{1,\cdots,n/2}$.
```

Note that p denotes the public helper string, while r denotes the secret string. The correctness of computational FE relies on the $\mathsf{Decode}_t(\mathbb{A}, b)$ algorithm [16], which is explicitly shown as follows.

```
1. Input: $(\mathbb{A}, b = \mathbb{A}\cdot x + w - w')$.
2. Select $2n$ distinct indices $i_1, \cdots, i_{2n} \leftarrow [1, \cdots, m]$.
3. Restrict $\mathbb{A}, b$ to rows $i_1, \cdots, i_{2n}$; Denote these by $\mathbb{A}_{i_1}, \cdots, \mathbb{A}_{i_{2n}}, b_1, \cdots, b_{i_{2n}}$.
4. Find $n$ linearly independent rows of $\mathbb{A}_{i_1}, \cdots, \mathbb{A}_{i_{2n}}$ (if no such rows exist,
   output abort and stop), and restrict $\mathbb{A}_{i_1}, \cdots, \mathbb{A}_{i_{2n}}, b_{i_1}, \cdots, b_{i_{2n}}$ to $n$ rows.
   Denote the result by $\mathbb{A}', b'$.
5. Compute $x' = \mathbb{A}'^{-1} \cdot b'$.
6. If $b - \mathbb{A}\cdot x'$ has at most $t$ non-zero coordinates, then outputs $x'$; Otherwise,
   it returns to step 2.
```

Recall that $\mathbb{A} \in \mathbb{Z}_q^{m\times n}, b \in \mathbb{Z}_q^m$, and Decode_t algorithm can correct *at most* $t = \mathcal{O}(\log n)$ errors (of Hamming distance) in a random linear code. Also note that with probability at least $1/poly(\lambda)$, none of the $2n$ rows selected in step 2 have errors (i.e., noisy biometrics w and w' agree on these rows), thus x' is a solution to the linear system. Furthermore, we notice that the sketch from LWE satisfies the *linearity* defined in [32, 26]. That is,

$$\mathsf{SS}(w_i, \mathtt{sk} + \Delta(\mathtt{sk})) = \mathsf{SS}(w_i, \mathtt{sk}) + \mathbb{A}\cdot\Delta(\mathtt{sk}).$$

The computational fuzzy extractors (FE) from LWE has an inherent property: "key privacy", which is introduced by Bellare et al. [5]. Informally, it means that an adversary in possession of a ciphertext cannot tell which specific key, out of a set of known public keys, is the one under which the ciphtertext was created. In particular, they have formalized a new model: "indistinguishability of keys" (IK). We formally prove that the computational FE is secure in the IK-CPA model. In addition, we discover that both computational FE [16] and its variant reusable FEs [2, 35] have such inherent property.

Lemma 3. *The computational fuzzy extractors from LWE achieves the IK-CPA security if the* D-LWE$_{q,n,m,\chi}$ *assumption is* (ϵ, s_{sec}) *secure.*

Informally, we can essentially think of the sketch $\mathbb{A} \cdot x + w$ as an "encryption" of x that where decryption works from any close w' (i.e., decryption key). Furthermore, we can also think of any two "encryptions" $\mathbb{A}_0 \cdot x_0 + w_0$ and $\mathbb{A}_1 \cdot x_1 + w_1$ (in a multi-user setting, we set $\mathbb{A}_0 = \mathbb{A}_1$ which is shared among all users) are indistinguishable by any third parties without having decryption keys (w'_0, w'_1).

Definition 6. *The IK-CPA experiment between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ is defined below [5].*

```
$Experiment\ \mathsf{Exp}_{PKE}^{IK\text{-}CPA}(\lambda)$
$(\mathtt{pk}_0, \mathtt{sk}_0), (\mathtt{pk}_1, \mathtt{sk}_1) \leftarrow \mathsf{KG}(1^\lambda)$
$(msg^*, st) \leftarrow \mathcal{A}(find, \mathtt{pk}_0, \mathtt{pk}_1)$
$c^* \leftarrow \mathsf{Enc}_{\mathtt{pk}_b}(msg^*)$
$b' = \mathcal{A}(guess, st, c^*)$
$If\ b' = b, return\ 1; else, return\ 0.$
```

Note that st denotes some state information. We define the advantage of $\mathcal{A}$ as

$$\mathtt{Adv}_{\mathcal{A}}^{IK\text{-}CPA}(\lambda) = |\Pr[\mathcal{S} \to 1] - 1/2|.$$

Definition 7. *A public key encryption scheme* $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be IK-CPA secure if* $\mathtt{Adv}_{\mathcal{A}}^{IK\text{-}CPA}(\lambda)$ *is negligible in λ for any PPT $\mathcal{A}$.*

Proof of Lemma 3.

```
Adversary $\mathcal{S}(\mathbb{A}, \mathbb{A} \cdot X + \chi)$
$b \in \{0,1\}, u \xleftarrow{R} \mathbb{Z}_q^n$
$\mathtt{pk}_0 \leftarrow \mathbb{A} \cdot X + \chi; \mathtt{pk}_1 \leftarrow \mathbb{A} \cdot (X + u) + \chi$
$(m^*, st) \leftarrow \mathcal{A}(\text{find}, \mathtt{pk}_0, \mathtt{pk}_1)$
$b' = \mathcal{A}(\text{guess}, st, c^*)$
$If\ b' = b, return\ 1; else, return\ 0.$
```

Fig. 1. Description of adversary $\mathcal{S}$ for the proof

Proof. Assume that there exists a PPT $\mathcal{A}$ breaking the IK-CPA security of the computational fuzzy extractors from LWE, then we can construct an algorithm $\mathcal{S}$ to break the decisional LWE (*D-LWE*$_{q,n,m,\chi}$) assumption. The algorithm $\mathcal{S}$

has almost the same time complexity with $\mathcal{A}$. For simplicity, we first consider the shared public parameter by all users such that $\mathbb{A}_0 = \mathbb{A}_1$.

The algorithm $\mathcal{S}$ uses $\mathcal{A}$ as a subroutine (see Fig. 3.3). $\mathcal{S}$ first generates another distribution which has the same property and distribution as its own challenge distribution $(\mathbb{A}, \mathbb{A} \cdot X + \chi)$ using self-reducibility technique [6]. That means if its challenge is a *real* distribution, then it is the computed distribution; Otherwise, it is a *random* distribution over $(\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$. Then using its challenge and computed distributions, $\mathcal{S}$ outputs two challenge public keys for $\mathcal{A}$ (in the find stage). At the end of find stage, $\mathcal{A}$ submits a challenge message msg^* and some state information st to $\mathcal{S}$. $\mathcal{S}$ takes challenge message msg^* and st as input, outputs a challenge ciphertext which is an encryption of msg^* under $\mathtt{pk}_b$ according to the bit b.

We then analyze the behaviour of $\mathcal{S}$ on $\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-REAL}}$ and $\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-RAND}}$ respectively. In the $\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-REAL}}$, the input $(\mathbb{A}, \mathbb{A} \cdot X + \chi)$ to $\mathcal{S}$ satisfy the Rep algorithm of FE described in Section 3.3. Notice that the computed distribution $(\mathbb{A}, \mathbb{A} \cdot (X + u) + \chi)$ is also valid and they are uniformly and independently distributed over $(\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$, because $\mathbb{A} \cdot (X + u) + \chi = \mathbb{A} \cdot X + \chi + \mathbb{A} \cdot u$ and u is a randomly element in $\mathbb{Z}_q^n$. Thus, $\mathcal{S}$ can simulate the proper distribution of two challenge public keys (i.e., $\mathtt{pk}_0 \leftarrow \mathbb{A} \cdot X + \chi$ and $\mathtt{pk}_1 \leftarrow \mathbb{A} \cdot (X + u) + \chi$), and the challenge ciphertext c^* is distributed exactly like a real encryption of message on msg^* under public key $\mathtt{pk}_b$.

$$
\begin{aligned}
&c_0 \leftarrow \mathbb{A} \cdot (X + msg^*) + \chi. \triangleright\ \ if\ \ b = 0\\
&c_1 \leftarrow \mathbb{A} \cdot (X + u + msg^*) + \chi. \triangleright\ \ otherwise
\end{aligned}
$$

Therefore, we have

$$
\begin{aligned}
\Pr[\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-REAL}}(\lambda) = 1] &= 1/2 \cdot \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{IK-CPA-1}}(\lambda) = 1]\\
&\quad + 1/2 \cdot (1 - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{IK-CPA-0}}(\lambda) = 1])\\
&= 1/2 + 1/2 \cdot \mathtt{Adv}_{\mathcal{A}}^{\text{IK-CPA}}(\lambda).
\end{aligned}
$$

As for $\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-RAND}}$, the input distributions to $\mathcal{S}$ in Fig. 3.3 are all uniformly distributed over $(\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$. Therefore, the corresponding computed distribution above are also uniformly and independently distributed over $(\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$. In particular, the challenge ciphertext is a random distribution over $(\mathbb{Z}_q^{m\times n}, \mathbb{Z}_q^m)$, and independent of bit b. Hence we have

$$
\Pr[\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-RAND}}(\lambda) = 1] \leq 1/2 + 1/2^{\lambda-1}.
$$

The last term indicates that the random distribution to $\mathcal{S}$ happen to have the distribution of a real distribution, which is bounded by $1/2^{\lambda-1}$ since $2^{\lambda-1} < q < 2^{\lambda}$. By combing all equations above, we have

$$
\begin{aligned}
\mathtt{Adv}_{\mathcal{S}}^{\text{LWE}}(\lambda) &= \Pr[\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-REAL}}(\lambda) = 1] + \Pr[\mathsf{Exp}_{\mathcal{S}}^{\text{LWE-RAND}}(\lambda) = 1]\\
&\geq 1/2 \cdot \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{IK-CPA}}(\lambda) - 1/2^{\lambda-1}.
\end{aligned}
$$

We can also show that $\mathbb{A}_0 \neq \mathbb{A}_1$ when public parameter is chosen at random over $\mathbb{Z}_q^{m\times n}$, while $\mathcal{S}$ will slightly change to $\mathtt{pk}_0 \leftarrow (\mathbb{A}_0 = \mathbb{A}, \mathbb{A}_0 \cdot X + \chi); \mathtt{pk}_1 \leftarrow (\mathbb{A}_1 = \mathbb{A}_0 \cdot \mathbb{A}^*, \mathbb{A}_1 \cdot (X + u) + \chi)$, where $\mathbb{A}^* \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{m\times n}$.

3.4 Lattice-based Signatures

Definition 8. *The continuous Gaussian distribution over $\mathbb{R}^m$ centered at $\boldsymbol{v}$ with standard deviation σ is defined by the function $\rho^m_{v,\sigma}(x) = (\frac{1}{\sqrt{2\pi\sigma^2}})^m e^{\frac{-||x-v||^2}{2\sigma^2}}$.*

When the center $\mathbf{v} = 0$ we write $\rho^m_\sigma(x)$, and the $||\cdot||$ denotes the Euclidean norm. The discrete Gaussian distribution over $\mathbb{Z}^m$ is defined as follows.

Definition 9. *The discrete Gaussian distribution over $\mathbb{Z}^m$ centered at some $\boldsymbol{v} \in \mathbb{Z}^m$ with standard deviation σ is defined as $D^m_{v,\sigma}(x) = \rho^m_{v,\sigma}(x)/\rho^m_\sigma(\mathbb{Z}^m)$.*

We say that a lattice-based digital signature scheme $\Sigma = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Sign}, \mathsf{Verify})$ is *homomorphic*, if the following conditions are held.

1. *Simple Key Generation.* $pp \leftarrow \Sigma.\mathsf{Setup}(\lambda)$ and $(\mathtt{sk}, \mathtt{pk}) \leftarrow \Sigma.\mathsf{KG}(pp)$, where $\mathtt{pk}$ derives from $\mathtt{sk}$ via a deterministic algorithm $\mathtt{pk} \leftarrow \mathsf{KG}'(pp, \mathtt{sk})$.
2. *Linearity of Keys.* $\mathtt{pk}' \leftarrow \mathsf{KG}'(pp, \mathtt{sk} + \Delta(\mathtt{sk})) = M_{\mathtt{pk}}(pp, \mathsf{KG}'(pp, \mathtt{sk}), \Delta(\mathtt{sk}))$, where $M_{\mathtt{pk}}$ denotes a deterministic algorithm which takes pp, a public key $\mathtt{pk}$ and a "shifted" value $\Delta(\mathtt{sk})$, outputs a new public key $\mathtt{pk}'$.
3. *Linearity of Signatures.* Two distributions are identical: $\{\sigma' \leftarrow \Sigma.\mathsf{Sign}(pp, \mathtt{sk} + \Delta(\mathtt{sk}), msg)\}$ and $\{\sigma' \leftarrow M_\Sigma(pp, \mathtt{pk}, msg, \sigma, \Delta(\mathtt{sk}))\}$, where $\sigma \leftarrow \Sigma.\mathsf{Sign}(pp, \mathtt{sk}, msg)$ and M_Σ denotes a deterministic algorithm which takes pp, a public key $\mathtt{pk}$, a message-signature pair (msg, σ) and a "shifted" value $\Delta(\mathtt{sk})$, outputs a new signature σ'.
4. *Linearity of Verifications.* We require that $\Sigma.\mathsf{Verify}(pp, M_{\mathtt{pk}}(pp, \mathtt{pk}, \Delta(\mathtt{sk})), msg, M_\Sigma(pp, \mathtt{pk}, msg, \sigma, \Delta(\mathtt{sk}))) =$ "1", and $\Sigma.\mathsf{Verify}(pp, \mathtt{pk}, msg, \sigma) =$ "1".

We show that the lattice-based Schnorr-like signature schemes [25, 14, 3] have the *homomorphic* properties regarding keys and signatures. We first present the simplest version of the lattice-based signature scheme based on SIS [25], then we show Lemma 4 afterwards.

– the user's signing key is $\mathtt{sk} \leftarrow \mathbb{Z}_q^{m\times k}$, and its verification key is $\mathtt{pk} \leftarrow \mathbb{A} \cdot \mathtt{sk} \mod q$. These exists a hash function $\mathtt{H} : \{0,1\}^* \rightarrow \{-d, 0, d\}^k$ and $\mathbb{A} \in \mathbb{Z}_q^{n\times m}$.
– the signer generates a potential message-signature pair $(msg, \sigma) = (msg, c, z)$, where σ includes $c \leftarrow \mathtt{H}(\mathbb{A} \cdot y \mod q, msg)$ and $z \leftarrow \mathtt{sk} \cdot c + y$ (for $y \in \mathcal{D}^m_\sigma$).
– the verifier accepts the signature iff $||z|| \leq 2\sigma\sqrt{m}$ and $c = \mathtt{H}(\mathbb{A} \cdot z - \mathtt{pk} \cdot c \mod q, msg)$.

Lemma 4. *The lattice-based Schnorr-like signature schemes satisfy the homomorphic property.*

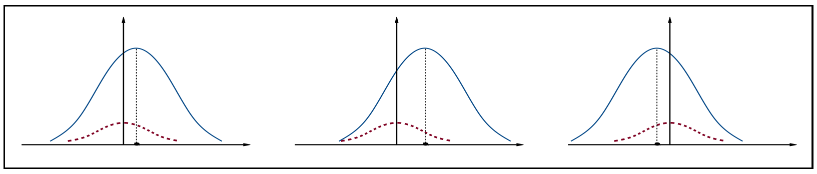

Fig. 2. Rejection Sampling with "Shifted" Gaussian Distribution. In blue is the distribution of z, with $\mathbf{v} = \mathtt{sk} \cdot c$ (left figure) and "shifted" $\mathbf{v}' = (\mathtt{sk} + \Delta(\mathtt{sk})) \cdot c$ (middle and right figures) over the spaces of all y *before* rejection sampling. In dashed red is the *common* target distribution $\mathcal{D}^m_\sigma$.

Proof. The secret/public key is a pair of integer matrixes such that $(\mathtt{sk}, \mathtt{pk}) = (\mathtt{sk}, \mathbb{A} \cdot \mathtt{sk} \mod q)$, where $\mathtt{sk} \in \mathbb{Z}_q^{m \times k}$ and $\mathbb{A} \in \mathbb{Z}_q^{n \times m}$ is a public parameter generated by a trusted party. The "shifted" $\mathtt{pk}' = \mathtt{pk} + \mathbb{A} \cdot \Delta(\mathtt{sk}) = \mathsf{KG}'(\mathbb{A}, \mathtt{sk} + \Delta(\mathtt{sk}))$. Therefore, the condition 1 and 2 are immediate held. As for the condition 3, by definition, $\sigma = (c, z) = (c \leftarrow \mathtt{H}(\mathbb{A} \cdot y \mod q, msg), z \leftarrow \mathtt{sk} \cdot c + y)$. If $\sigma' = \underline{(c', z')}$ is output by $M_\Sigma(pp, \mathtt{pk}, msg, \sigma, \Delta(\mathtt{sk}))$, then it holds that

$$z' \leftarrow \mathtt{sk} \cdot c + y + \underline{\Delta(\mathtt{sk}) \cdot c}, \quad c' = c \leftarrow \mathtt{H}(\mathbb{A} \cdot y \mod q, msg).$$

The output of z (resp. z') depends on the vector $\mathbf{v} = \mathtt{sk} \cdot c$ (resp. $\mathbf{v}' = (\mathtt{sk} + \Delta(\mathtt{sk})) \cdot c'$) as well as the secret key $\mathtt{sk}$ (resp. $\mathtt{sk} + \Delta(\mathtt{sk})$). To ensure that the signatures do not leak the secret key, the *rejection sampling* (Theorem 3.4 [25]) aims to remove such dependence. Informally, rejection sampling is to "re-center" the distribution of z to be a discrete Gaussian distribution centered at "zero" rather than at $\mathbf{v} = \mathtt{sk} \cdot c$. To show the condition 3 is held such that two distributions are identical in the case of "shifted" Gaussian distributions (i.e., $\mathcal{D}^m_{\mathbf{v}',\sigma}$ where $\mathbf{v}' = (\mathtt{sk} + \Delta(\mathtt{sk})) \cdot c$), we present Figure 2 in a *two-dimensional space* for correctness check.

When applying the rejection sampling (with same parameters pp, message msg and randomness $y \in \mathcal{D}^m_\sigma$), we have $z = z' \leftarrow \Sigma.\mathsf{Sign}(pp, \mathtt{sk}+\Delta(\mathtt{sk}), msg; y) \in \mathcal{D}^m_\sigma$. *That is, the target distribution $\mathcal{D}^m_\sigma$ is independent of the "shifted" Gaussian distributions.* It is easy to check that the two distributions we considered are identical, which shows that the condition 3 holds. Furthermore, the randomness $y \in \mathcal{D}^m_\sigma$ ensures that the discrete Gaussian distributions centered at "zero" and $\mathbf{v}/\mathbf{v}'$ have sufficient *common overlap*, hence the condition 4 regarding $\Sigma.\mathsf{Verify}$ is also held. Combing the above conditions together, the lemma is complete.

4 Reusable Fuzzy Signature

In this section, we introduce the formal definition and security model of RFS, and show the reusability of RFS. Essentially, the RFS is built on top of a family of universal hash functions $\mathtt{H}$, digital signatures Σ and fuzzy extractors FE (both Σ and FE have homomorphic properties as described in Section 3).

4.1 Definition

A RFS consists of the following algorithms:

- Setup: The algorithm takes security parameter λ as input, outputs a public parameter pp.
- $\mathsf{Fuz\text{-}KeyGen}$: The user takes public parameter pp and biometrics w as input, outputs a secret/public key pair $(\mathtt{sk}, \mathtt{pk}) \leftarrow \Sigma.\mathsf{KG}$, and a sketch $P \leftarrow \mathsf{FE.Gen}(\mathtt{sk}, w)$. We denote the sketch as $P = \mathsf{SS}(w, \mathtt{sk})$, and let the value pair $(\mathtt{pk}, P)$ be a reference.
- $\mathsf{Fuz\text{-}Sign}$: The randomized algorithm takes public parameter pp, a message msg and a new biometrics w' as inputs, outputs a tuple $(\mathtt{pk}', msg, \sigma, P')$, where $(\mathtt{sk}', \mathtt{pk}') \leftarrow \Sigma.\mathsf{KG}, \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathtt{sk}', msg)$, $P' \leftarrow \mathsf{FE.Gen}(\mathtt{sk}', w')$.
- $\mathsf{Fuz\text{-}Verify}$: The deterministic algorithm takes public parameter pp, a message-signature pair (msg, σ), a new sketch P' and a reference $(\mathtt{pk}, P)$ as input, outputs "1" if $1 = \Sigma.\mathsf{Verify}(\mathtt{pk}', \sigma, msg)$ and $\mathtt{pk}' \leftarrow M_{\mathtt{pk}}(pp, \mathtt{pk}, \Delta(\mathtt{sk}))$, where $\Delta(\mathtt{sk}) \leftarrow \mathtt{H}(\Delta(P))$ and $\Delta(P) = P' - P$; Otherwise, it outputs "0".

Correctness. The correctness of $\mathtt{pk}' \leftarrow M_{\mathtt{pk}}(pp, \mathtt{pk}, \Delta(\mathtt{sk}))$ holds if public keys have the *homomorphic* property and $\mathsf{dist}(w, w') \leq t$. Recall that $M_{\mathtt{pk}}$ is a deterministic algorithm, and $\mathtt{H}$ is a universal function which links digital signatures to fuzzy extractors.

4.2 Reusability

Informally, an adversary attempts to learn whether the reference pair is real or not. The formal reusability game between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ is defined as follows.

- Setup: $\mathcal{S}$ samples a biometrics $w^* \in \mathcal{M}$, generates a secret/public key pair $(\mathtt{sk}, \mathtt{pk})$, a secret string and public sketch pair (R, P) by running key generation algorithm $\mathsf{Fuz} - \mathsf{KeyGen}(pp, w^*)$. Then $\mathcal{S}$ sends a reference pair $(\mathtt{pk}, P)$ to $\mathcal{A}$. Let $N(\lambda)$ be the total number of executions of $\mathsf{Fuz} - \mathsf{KeyGen}(pp, w^*)$.
- $\mathsf{Training}$ $\mathcal{A}$ may adaptively make $\mathsf{Fuz\text{-}Sign}$ queries of the following form: a message msg_i and a shift $\delta \in \Delta$, while $\mathcal{S}$ performs the following
  - sample a new secret/public key pair $(\mathtt{sk}_i, \mathtt{pk}_i)$.
  - obtain (R_i, P_i) by running $\mathsf{Fuz} - \mathsf{KeyGen}(pp, w^* + \delta)$, where $R_i \leftarrow \mathtt{H}(\mathtt{sk}_i)$.
  - generate a message-signature pair (msg_i, σ_i) using secret key $\mathtt{sk}_i$ (i.e., $\sigma_i \leftarrow \mathsf{Sign}(\mathtt{sk}_i, msg_i)$).
  - Return a tuple $(\mathtt{pk}_i, msg_i, \sigma_i, P_i)$ to $\mathcal{A}$.

  $\mathcal{A}$ is allowed to reveal at most $N(\lambda)$-2 secret keys $\mathtt{sk}_i$ (including $\mathtt{sk}$), and denote the uncorrupted pair set as $\mathcal{U}'$.
- $\mathsf{Challenge}$: $\mathcal{S}$ tosses a random coin b first. Then, $\mathcal{S}$ randomly selects two pairs $(\mathtt{pk}_0, P_0), (\mathtt{pk}_1, P_1) \in \mathcal{U}'$ as the challenge references, and $\mathcal{S}$ removes them from $\mathcal{U}'$. Eventually, $\mathcal{S}$ sends a challenge reference to $\mathcal{A}$. That is

  $$\mathcal{A} \leftarrow (\mathtt{pk}_b, P_b) \quad \textit{where } (R_b, P_b) \leftarrow \mathsf{Fuz} - \mathsf{KeyGen}(pp, w^*) \textit{ and } R_b \leftarrow \mathtt{H}(\mathtt{sk}_b).$$

$\mathcal{A}$ is not allowed to reveal the secret key $\mathtt{sk}_b$, but $\mathcal{A}$ can continue to issue the $\mathsf{Fuz\text{-}Sign}$ queries with respect to the challenge reference $(\mathtt{pk}_b, P_b)$. Eventually, $\mathcal{A}$ outputs a bit b', $\mathcal{A}$ wins if $b' = b$. We define the advantage of an adversary $\mathcal{A}$ in the above game as

$$\mathtt{Adv}_{\mathcal{A}}^{\mathrm{RFS}}(\lambda) = |\Pr[\mathcal{S} \to 1] - 1/2|.$$

4.3 Reusability Analysis

We first review the *strongly* reusability of fuzzy extractors (FE). In the strongly reusability [2, 35], an adversary, who is given the pair (R_i, P_i) generated by several independent executions of $\mathsf{Gen}(w^* + \delta)$ on a series of inputs *related* to the biometrics w^*, attempts to distinguish an extracted secret string R^* from a uniform string. The biometrics reusability game between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ is defined as follows.

1. $\mathcal{S}$ samples $w^* \in \mathcal{M}$, and obtains (R^*, P^*) by running algorithm $\mathsf{Gen}(w^*)$. The value P^* is given to $\mathcal{A}$.
2. $\mathcal{A}$ may adaptively make queries of the following form:
   – $\mathcal{A}$ outputs a shift $\delta \in \Delta$.
   – $\mathcal{S}$ obtains (R, P) by running $\mathsf{Gen}(w^* + \delta)$, and returns them to $\mathcal{A}$.
3. $\mathcal{S}$ tosses a random coin b. If $b = 0$, $\mathcal{S}$ gives R^* to $\mathcal{A}$; Otherwise, $\mathcal{S}$ chooses $U \xleftarrow{\mathrm{R}} \{0,1\}^{|R^*|}$ and gives U to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b', $\mathcal{A}$ wins if $b' = b$.

Below, we reduce the reusability of RFS to the strongly reusability of FE.

Theorem 1. *The fuzzy signature schemes achieve the reusability if the underlying fuzzy extractor is strongly reusable.*

Proof. Let $\mathcal{S}$ denote an adversary against the reusability as defined above, who is given a public sketch P^* and a $\mathsf{Gen}(w^*)$ oracle $\mathcal{O}$, aims to distinguish a real secret string R^* from a random string U. $\mathcal{S}$ simulates the reusability game for $\mathcal{A}$ as follows.

– $\mathcal{S}$ obtains the value pair (R, P) by invoking his oracle $\mathcal{O}$, and $\mathcal{A}$ is given a reference pair $(\mathtt{pk}, P)$, where $\mathtt{pk} \leftarrow \mathsf{KG}(pp, \mathtt{sk}), \mathtt{sk} \leftarrow \mathtt{H}(R)$.
– If $\mathcal{A}$ issues a $\mathsf{Fuz\text{-}Sign}$ query in the form of (msg_i, δ_i) $(\delta_i \in \Delta)$ to $\mathcal{S}$, then $\mathcal{S}$ performs the following
  • obtain the values (R_i, P_i) by invoking his oracle $\mathcal{O}$ on input δ_i (i.e., $\mathsf{Gen}(w^* + \delta_i)$);
  • compute a secret key $\mathtt{sk}_i \leftarrow \mathtt{H}(R_i)$, and generates its corresponding public key $\mathtt{pk}_i \leftarrow \mathsf{KG}(pp, \mathtt{sk}_i)$;
  • generate a message-signature pair (msg_i, σ_i) using the secret key $\mathtt{sk}_i$ (i.e., $\sigma_i \leftarrow \mathsf{Sign}(\mathtt{sk}_i, msg_i)$);
  • return $(\mathtt{pk}_i, msg_i, \sigma_i, P_i)$ to $\mathcal{A}$.

Note that $\mathcal{S}$ can return at most $N(\lambda)$-2 secret keys to $\mathcal{A}$.
- $\mathcal{S}$ follows the reusability game to select a challenge reference pair $(\mathtt{pk}_b, P_b)$. After that, $\mathcal{S}$ obtains a pair (R^*, P^*) from his challenger oracle; Finally, $\mathcal{S}$ replaces the challenge reference pair $(\mathtt{pk}_b, P_b)$ by $(\underline{\mathtt{pk}^*}, P^*)$ and sends it to $\mathcal{A}$. Note that the corresponding secret key $\underline{\mathtt{sk}^*} = \mathtt{sk}^{*'} + \mathtt{sk}_b + \Delta(\mathtt{sk}^*)$, where $\Delta(\mathtt{sk}^*) \leftarrow \mathtt{H}[\Delta(P_b, P^*)]$ and $\mathtt{sk}^{*'}$ derives from either real secret string R^* or a random string U. In particular, upon receiving further Fuz-Sign queries w.r.t. challenge reference pair $(\underline{\mathtt{pk}^*}, P^*)$, $\mathcal{S}$ performs the following.
  - invoke his oracle $\mathcal{O}$ to obtain the values (R_i, P_i);
  - compute a "shift" $\Delta(\mathtt{sk}_i) \leftarrow \mathtt{H}[\Delta(P_i, P^*)]$ and computes a key $\mathtt{sk}'_i \leftarrow \mathtt{H}(R_i)$;
  - generate a secret key $\mathtt{sk}_i = \mathtt{sk}'_i + \mathtt{sk}^* + \Delta(\mathtt{sk}_i)$ (as well as public key $\mathtt{pk}_i$);
  - generate a message-signature pair (msg_i, σ_i) using the secret key $\mathtt{sk}_i$ (i.e., $\sigma_i \leftarrow \mathsf{Sign}(\mathtt{sk}_i, msg_i)$);
  - return $(\mathtt{pk}_i, msg_i, \sigma_i, P_i)$ to $\mathcal{A}$.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break the strongly reusability of FE.

5 Proposed Construction

The proposed construction consists of the following building blocks.

- A LWE-based computational fuzzy extractor scheme $\mathsf{FE} = (\mathsf{Gen}, \mathsf{Rep})$.
- An EUF-CMA secure digital signature scheme $\Sigma = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Verify})$.
- An IND-CPA secure public key encryption scheme $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

We use user i and server j to illustrate our construction below.

- Setup. Let λ be the security parameter, and we generate a universal hash function family $\mathbb{H} : \{y = \mathtt{H}_{\mathbb{A}}(x)\}$, which includes a public matrix $\mathbb{A}$.
- KeyGen. User i obtains a secret/public key pair $(\mathtt{sk}_i, \mathtt{pk}_i) \leftarrow \Sigma.\mathsf{KG}$ by running the key generation algorithm of Σ. The server j obtains a secret/public key pair $(ssk_j, spk_j) \leftarrow \mathsf{PKE.KG}$ by running the key generation algorithm of PKE.
- Enrollment. A user i wants to enroll herself to an authentication server j, performs the following steps
  - compute a secret $s_i \xleftarrow{\mathrm{R}} \mathtt{H}_{\mathbb{A}}^{-1}(\mathtt{sk}_i)$ from the secret key $\mathtt{sk}_i$, and derive a sketch $\mathsf{SS}(w_i, \mathtt{sk}_i) \leftarrow \mathsf{FE.Gen}(\mathbb{A}, w_i, s_i)$.
  - send a reference tuple (ID_i, VK_i) to server j (note that server j maintains a database DB of all enrolled users which includes verification keys $\{VK_i\} = \{(\mathtt{pk}_i, \mathsf{SS}(w_i, \mathtt{sk}_i))\}$).
- Authentication. User i and server j perform the following steps.
  - User i generates a ciphertext $C_i \leftarrow \mathsf{PKE.Enc}(spk_j, \mathrm{ID}_i)$ on her identity ID_i under the public key spk_j of server j, and sends it to server j.
  - Server j obtains the identity $\mathrm{ID}_i \leftarrow \mathsf{PKE.Dec}(ssk_j, C_i)$ by running the decryption algorithm of PKE, and sends a challenge nonce n_j to user i.

$$\begin{array}{ccc}
\text{User } i & & \text{Server } j \\
 & & \mathrm{DB} = \{(\mathrm{ID}_i, VK_i)\} \\
C_i \leftarrow \mathsf{PKE.Enc}(spk_j, \mathrm{ID}_i) & \xrightarrow{\quad C_i \quad} & \mathrm{ID}_i \leftarrow \mathsf{PKE.Dec}(ssk_j, C_i) \\
 & \xleftarrow{\quad n_j \quad} & \text{Challenge} : n_j \\
\text{Response} : n_i & & \\
(\mathtt{sk}'_i, \mathtt{pk}'_i) \leftarrow \Sigma.\mathsf{KG} & & \\
msg_{(i,j)} = (n_i, n_j) & & \\
\sigma_i \leftarrow \Sigma.\mathsf{Sign}(\mathtt{sk}'_i, msg_{(i,j)}) & & \\
\text{Sketch} : \mathsf{SS}(w'_i, \mathtt{sk}'_i) & \xrightarrow{msg_{(i,j)}, \sigma_i, \mathsf{SS}(w'_i, \mathtt{sk}'_i)} & \\
 & & VK'_i \leftarrow (VK_i, \mathsf{SS}(w'_i, \mathtt{sk}'_i)) \\
 & & \Sigma.\mathsf{Verify}(VK'_i, msg_{(i,j)}, \sigma_i) \overset{?}{=} 1
\end{array}$$

Fig. 3. Description of Authentication.

- User i performs the following steps
  - ∗ obtain a new secret/public key pair $(\mathtt{sk}'_i, \mathtt{pk}'_i) \leftarrow \Sigma.\mathsf{KG}$ and derive a new secret $s'_i \overset{R}{\longleftarrow} \mathtt{H}_{\mathbb{A}}^{-1}(\mathtt{sk}'_i)$ from the new secret key $\mathtt{sk}'_i$.
  - ∗ choose a response nonce n_i and generate a message-signature pair $(msg_{(i,j)}, \sigma_i) \leftarrow \Sigma.\mathsf{Sign}(\mathtt{sk}'_i, msg_{(i,j)})$, where $msg_{(i,j)} = (n_i, n_j)$.
  - ∗ derive a new sketch $\mathsf{SS}(w'_i, \mathtt{sk}'_i) \leftarrow \mathsf{FE.Gen}(\mathbb{A}, w'_i, s'_i)$, and send $(msg_{(i,j)}, \sigma_i, \mathsf{SS}(w'_i, \mathtt{sk}'_i))$ to server j.
- Server j performs the following steps
  - ∗ compute a "shift" secret between enrolled sketch and new sketch, where $\Delta(s) \leftarrow \mathsf{SS}(w_i, \mathtt{sk}_i) - \mathsf{SS}(w'_i, \mathtt{sk}'_i)$ (see correctness below).
  - ∗ compute a "shift" secret key $\Delta(\mathtt{sk}) = \mathtt{H}_{\mathbb{A}}(\Delta(s))$, and derive a new public key $\mathtt{pk}'_i$ from the enrolled public key $\mathtt{pk}_i$ and the "shift" secret key $\Delta(\mathtt{sk})$.
  - ∗ verify the message-signature pair $(msg_{(i,j)}, \sigma_i)$ under the new public key $\mathtt{pk}'_i$. If the signature passes the verification, it *accepts*; Otherwise, it *aborts.*

Instantiations and Correctness. We hereby try to instantiate the underlying cryptographic building blocks which are able to resist with quantum computers. First, to instantiate the computational fuzzy extractors from LWE, we rely on the reusable fuzzy extractors proposed by Apon et al. [2] (or the one called robustly reusable fuzzy extractor [35] with a non-uniform source). Second, we can implement the lattice-based digital signatures [25, 14, 3] as described in Section 3.4. Third, we could use the passively secure (i.e., IND-CPA) encryptions such as Regev's LWE-based cryptosystem [29] to instantiate the underlying public key encryptions. We notice that the passively secure encryptions will suffice for our defined user privacy model, and we refer to [28] for passively and actively secure cryptosystems which might be alternatively applicable to instantiate our proposed construction.

The public matrix includes $\mathbb{A} = (\mathbb{A}_1 \in \mathbb{Z}_{q^{mk}}, \mathbb{A}_2 \in \mathbb{Z}_q^{mk \times n}, \mathbb{A}_3 \in \mathbb{Z}_q^{m \times n})$ (the transpose of $\mathbb{A}_3$ is $\mathbb{A}_3^\top \in \mathbb{Z}_q^{n \times m}$). We set $m \geq c \cdot n$, where c is a positive constant, $n = n(\lambda)$ and $q = q(\lambda) \geq 2$ are two integers. Let the universal hash function be $\{\mathtt{H}_{(\mathbb{A}_1, \mathbb{A}_2)} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{m \times k}\}_{\mathbb{A}_1 \in Z_1, \mathbb{A}_2 \in Z_2}$. For each seed pair $(\mathbb{A}_1, \mathbb{A}_2) \in (\mathbb{Z}_{q^{mk}}, \in$

$\mathbb{Z}_q^{mk\times n}$) and $y \in \mathbb{Z}_q^{m\times k}$, we define "$\mathtt{H}^{-1}_{(\mathbb{A}_1,\mathbb{A}_2)}(y)$" as the *set* of pre-images of y under $\mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}$. That is, $\mathtt{H}^{-1}_{(\mathbb{A}_1,\mathbb{A}_2)}(y) = \{x \in \mathbb{Z}_q^n : \mathtt{H}_{\mathbb{A}_1,\mathbb{A}_2}(x) = y\}$. In particular, $x \overset{\text{R}}{\leftarrow} \mathtt{H}^{-1}_{(\mathbb{A}_1,\mathbb{A}_2)}(y)$ means that we choose a vector x uniformly at random from set $\mathtt{H}^{-1}_{(\mathbb{A}_1,\mathbb{A}_2)}(y)$, and its size is $c \cdot k$.

The sketch $\mathsf{SS}(w_i, \mathtt{sk}_i)$ is instantiated with a random linear code over a finite field $\mathbb{Z}_q$. That is, $\mathsf{SS}(w_i, \mathtt{sk}_i) \leftarrow \mathbb{A}_3 \cdot s_i + w_i$, where biometrics $w_i \in \mathbb{Z}_q^m$ and secret $s_i \in \mathbb{Z}_q^n$. The correctness of $\Delta(s)$ relies on an algorithm $\mathsf{Decode}_t(\mathbb{A}_3, b)$ [16]. According to the Decode_t algorithm, we have the following equation with respect to the "shift" secret $\Delta(s) = s_i - s_i'$.

$$\begin{aligned}\mathsf{SS}(w_i, \mathtt{sk}_i) - \mathsf{SS}(w_i', \mathtt{sk}_i') &= \mathbb{A}_3 \cdot s_i + w_i - (\mathbb{A}_3 \cdot s_i' + w_i') \\ &= \mathbb{A}_3 \cdot (s_i - s_i') + (w_i - w_i') \\ &= \mathbb{A}_3 \cdot \Delta(s) + \underline{\delta}.\end{aligned}$$

We assume that δ has *at most* t non-zero coordinates (i.e., at most t of non-zero coordinates can be zeroed out by $w_i - w_i'$), and notice that $\mathbb{A}_3 \cdot \Delta(s)$ is a linear system with m equations and $2n$ unknowns. If $m \geq 6n$ (we set $c = 6$ as suggested by [2] in order to ensure the Decode_t works with expected running time), then one can recover $\Delta(s)$ using the Decode_t algorithm with *high* probability (we refer to [16] for success probability and time complexity of Decode_t). Furthermore, we emphasize that the biometrics w_i, w_i' are computationally secure, because the server j is allowed to learn the "shift" secret $\Delta(s) = s_i - s_i'$ only.

5.1 Security Analysis

The security result of our proposed construction is shown by the following theorems.

Theorem 2. *The proposed RFS-RUA achieves user authenticity in the random oracle model if the family of universal hash functions* $\mathbb{H} = \{\mathtt{H}_{(z_1,z_2)} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{m\times k}\}_{z_1\in Z_1, z_2\in Z_2}$ *is ϵ-statistically secure, the D-LWE$_{q,n-k,m,\chi}$ assumption is (ϵ, s'_{sec}) secure and the digital signature scheme Σ is EUF-CMA secure.*

Proof. We define a sequence of games $\{\mathbb{G}_i\}$ and let $\mathtt{Adv}_i^{\text{RFS-RUA}}$ denote the advantage of the adversary $\mathcal{A}$ in game $\mathbb{G}_i$. Assume that $\mathcal{A}$ activates at most $m(\lambda)$ sessions in each game. We highlight the differences between adjacent games by underline.

- $\mathbb{G}_0$: This is the original game for user authenticity security.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_0$ except that the simulator $\mathcal{S}$ will output a random bit if server j *accepts* user i, but $\mathsf{sid}_i \neq \mathsf{sid}_j$. Since N users involved in this game, we have:

$$\left|\mathtt{Adv}_0^{\text{RFS-RUA}} - \mathtt{Adv}_1^{\text{RFS-RUA}}\right| \leq N \cdot m(\lambda)^2/2^\lambda. \quad (1)$$

– $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except the following difference: $\mathcal{S}$ randomly chooses $g \in [1, m(\lambda)]$ as a guess for the index of the Challenge session. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s challenge query does not occur in the g-th session. Therefore we have

$$\mathtt{Adv}_1^{\text{RFS-RUA}} = m(\lambda) \cdot \mathtt{Adv}_2^{\text{RFS-RUA}}. \tag{2}$$

– $\mathbb{G}_3$: This game is identical to game $\mathbb{G}_2$ except that in the g-th session, the k-size pseudorandom bit of encrypted secret in the sketch $\mathsf{SS}(w_i, \mathtt{sk}'_i)$ of user i w.r.t. server j is replaced by a random value. Below we show that the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible under the D-LWE$_{q,n-k,m,\chi}$ assumption.
Let $\mathcal{S}$ denote a distinguisher against the D-LWE$_{q,n-k,m,\chi}$ assumption, who is given a tuple $(\underline{X_{1,\cdots,k}}, \mathbb{A}, \mathbb{A} \cdot X + \chi)$, aims to distinguish the real LWE tuple from a random tuple $(\underline{U}, \mathbb{A}, \mathbb{A} \cdot X + \chi)$ where $U \in_R \mathbb{Z}_q^k$. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- Setup. $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating N users and M servers with the corresponding identities. $\mathcal{S}$ randomly selects indexes (i, j) and guesses that the g-th session will happen with regard to user i at server j. $\mathcal{S}$ sets the sketch of user i w.r.t. server j as $\mathsf{SS}(w_i, \mathtt{sk}_b)$ such that $\mathsf{SS}(w_i, \mathtt{sk}_b) = \mathbb{A} \cdot \underline{X_b} + \chi$, where $\mathtt{sk}_b = \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(\underline{X_b})$ and $X_b = \underline{X_{1,\cdots,k}}$. $\mathcal{S}$ generates user i's enrolled public/secret key pair $(\mathtt{pk}_i^l, \mathtt{sk}_i^l)$ w.r.t. l servers $(l \neq j)$, and their corresponding sketches $\{\mathsf{SS}(w_i, \mathtt{sk}_b{+}\mathtt{sk}_l)\}$. In addition, $\mathcal{S}$ honestly generates biometrics for N-1 users, and generates enrolled public/secret key pairs and sketches as Enrollment specified for N-1 users w.r.t. M servers. Eventually, $\mathcal{S}$ sends all enrolled public keys and references to $\mathcal{A}$. $\mathcal{S}$ sets $\mathbb{A}_3 = \mathbb{A}$, and generates rest public parameters (including matrixes $\mathbb{A}_1, \mathbb{A}_2$) for the system. $\mathcal{S}$ also chooses a random vector from $\mathbb{Z}_q^{n-k}$ to construct $X_b \in \mathbb{Z}_q^n$, we omit it in the following proof for simplicity.
- Training. $\mathcal{S}$ answers $\mathcal{A}$'s queries as follows.
  - $*$ If $\mathcal{A}$ issues a Send query in the form of n_j to $\mathcal{S}$, $\mathcal{S}$ chooses a response nonce n_i first, then $\mathcal{S}$ honestly generates the protocol transcript T_i using user i's enrolled secret key $\mathtt{sk}_b$ and sketch $\mathsf{SS}(w_i, \mathtt{sk}_b)$. Specifically, $T_i = (msg_{(i,j)}, \sigma_i, \mathsf{SS}(w_i, \mathtt{sk}'_i))$, where $\sigma_i \leftarrow M_\Sigma(pp, \mathtt{pk}_b, \mathsf{Sign}(\mathtt{sk}_b, msg_{(i,j)}), \Delta(s_i))$, $\mathsf{SS}(w_i, \mathtt{sk}'_i) \leftarrow \mathbb{A}{\cdot}X_b{+}\chi{+}\mathbb{A}{\cdot}\Delta(s_i)$, where $\mathtt{sk}'_i \leftarrow \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(s_b{+}\Delta(s_i))$, $s_b \overset{\text{R}}{\leftarrow} \mathtt{H}^{-1}_{(\mathbb{A}_1,\mathbb{A}_2)}(\mathtt{sk}_b)$ and $\Delta(s_i) \in_R \mathbb{Z}_q^n$ is chosen by $\mathcal{S}$.
  As for user i's g-th session w.r.t. server j, $\mathcal{S}$ first generates $\mathsf{SS}(w_i, \mathtt{sk}'_b) \leftarrow \mathbb{A}{\cdot}X_b{+}\chi{+}\mathbb{A}{\cdot}\Delta(s)$, and denotes $X'_b = X_b{+}\Delta(s)$ and $X_b = \underline{U}$; $\mathcal{S}$ then generates a secret/public key pair $(\mathtt{sk}'_b, \mathtt{pk}'_b)$ from X'_b, where $\mathtt{sk}'_b = \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(X'_b)$ and $\mathtt{pk}'_b = \mathbb{A}^\top{\cdot}\mathtt{sk}'_b$; eventually, $\mathcal{S}$ honestly generates the message-signature pair according to the protocol specification, and returns the protocol transcript to $\mathcal{A}$.
  - $*$ If $\mathcal{A}$ issues a Biometrics Reveal query to user i, then $\mathcal{S}$ aborts.
  - $*$ If $\mathcal{A}$ issues a Secret Key Reveal query to an instance oracle $\Pi^g_{ID_i}$ (g-th session of user i w.r.t. server j), then $\mathcal{S}$ returns new secret key $\mathtt{sk}'_b$ to $\mathcal{A}$.

  * If $\mathcal{A}$ issues $\mathsf{Biometrics\ Shift}$ query in the form of δ to $\mathcal{S}$, then $\mathcal{S}$ returns $\mathsf{SS}(w_i+\delta, \mathtt{sk}) = \mathbb{A}\cdot \underline{X_b} + \chi + \delta$ by the linearity of the sketch, where $\mathtt{sk}$ can be either enrolled secret key $\mathtt{sk}_b$ or new secret key $\mathtt{sk}'_b$ that involves at g-th session.
  * If $\mathcal{A}$ issues $\mathsf{Secret\ Key\ Shift}$ query in the form of ϕ to $\mathcal{S}$, then $\mathcal{S}$ returns new public key $\mathtt{pk}'_b \leftarrow \mathsf{KG}(pp, \phi(\mathtt{sk}_b))$. Notice that $\mathcal{A}$ is not allowed to obtain user i's enrolled secret key $\mathtt{sk}_b$.

  Note that in the $\mathsf{Challenge}$ session of user i w.r.t. server j, if the challenge of $\mathcal{S}$ is $\underline{X_{1,\cdots,k}}$, then the simulation is consistent with $\mathbb{G}_2$; Otherwise, the simulation is consistent with $\mathbb{G}_3$. If the advantage of $\mathcal{A}$ is significantly different in $\mathbb{G}_2$ and $\mathbb{G}_3$, then $\mathcal{S}$ can break the D-LWE$_{q,n-k,m,\chi}$. Since at most N users involved in the system, hence we have

$$\left|\mathtt{Adv}_2^{\text{RFS-RUA}} - \mathtt{Adv}_3^{\text{RFS-RUA}}\right| \leq N \cdot \mathtt{Adv}_{\mathcal{S}}^{\text{D-LWE}_{q,n-k,m,\chi}}(\lambda). \tag{3}$$

– $\mathbb{G}_4$: This game is identical to game $\mathbb{G}_3$ except that in the g-th session, the enrolled secret key $\mathtt{sk}_i^j$ w.r.t., server j is replaced by a random value. Below we show that the difference between $\mathbb{G}_3$ and $\mathbb{G}_4$ is bounded by a negligible probability.
  Let $\mathcal{S}$ simulate the whole environment honestly according to the protocol specification, and it is easy to see that all the queries made to a user can be simulated perfectly using the user's secret keys and biometrics. In particular, the enrolled secret key of user i w.r.t. server j is $\underline{\mathtt{sk}_i^j}$. In the g-th session of user i w.r.t server j, to answer the Send query from $\overline{\mathcal{A}, \mathcal{S}}$ will simulate the protocol transcript T'_i as follows. $\mathcal{S}$ first simulates the sketch as $\mathsf{SS}(w_i, \mathtt{sk}'_i) \leftarrow \mathbb{A}_3\cdot(s_i + \Delta(s)) + w_i$, where $\mathtt{sk}'_i \leftarrow \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(s_i+\Delta(s)), s_i \xleftarrow{\text{R}} \mathtt{H}^{-1}_{(\mathbb{A}_1,\mathbb{A}_2)}(\underline{u}), u \in_R \mathbb{Z}_q^{m\times k}$, and $\Delta(s)$ is randomly chosen by $\mathcal{S}$; $\mathcal{S}$ then generates a secret/public key pair $(\mathtt{sk}'_i, \mathtt{pk}'_i)$ from $s_i + \Delta(s)$; eventually, $\mathcal{S}$ honestly generates the message-signature pair using the same method described in previous game $\mathbb{G}_3$.
  We then analyze the statistical distance between two distributions $T'_i = (msg_{(i,j)}, \sigma_i, \mathsf{SS}(w_i, \mathtt{sk}'_i))$ and T_i (of previous game $\mathbb{G}_4$). We notice that the only difference is the simulated value $s_i \xleftarrow{\text{R}} \mathtt{H}^{-1}_{(\mathbb{A}_1,\mathbb{A}_2)}(\underline{u})$ instead of taking the real enrolled secret key $\mathtt{sk}_i^j$ as input, and according to Lemma 2, we have the statistically distance between $\mathtt{sk}_i^j \leftarrow \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(s_i)$ and $u \in_R \mathbb{Z}_q^{m\times k}$ with probability no greater than ϵ. Since at most M servers involved in the system, hence we have

$$\left|\mathtt{Adv}_3^{\text{RFS-RUA}} - \mathtt{Adv}_4^{\text{RFS-RUA}}\right| \leq M \cdot \mathtt{Adv}_{\mathcal{S}}^{\mathbb{H}}(\lambda). \tag{4}$$

– $\mathbb{G}_5$: This game is identical to game $\mathbb{G}_4$ except that in the g-th session, $\mathcal{S}$ outputs a random bit if **Forge** event happens where $\mathcal{A}$'s Send query includes a valid forgery σ^* while user i's secret key w.r.t server j is not corrupted. Then we have

$$\left|\mathtt{Adv}_4^{\text{RFS-RUA}} - \mathtt{Adv}_5^{\text{RFS-RUA}}\right| \leq \Pr[\textbf{Forge}]. \tag{5}$$

Let $\mathcal{F}$ denote a forger against a (lattice-based) signature scheme Σ with EUF-CMA security, who is given a verification key $\mathtt{pk}^*$ and a signing oracle $\mathcal{O}$, and aims to find a forgery σ^*. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- Setup. $\mathcal{F}$ sets up the game for $\mathcal{A}$ by creating N users and M servers with the corresponding identities and biometrics. $\mathcal{F}$ sets up the verification key of user i w.r.t. server j as $VK_i^j = (\mathtt{pk}^*, \mathsf{SS}(w_i, \mathtt{sk}_i^j))$, where $\mathsf{SS}(w_i, \mathtt{sk}_i^j) = \mathbb{A}_3 \cdot s_i + w_i, s_i \in_R \mathbb{Z}_q^n$. $\mathcal{F}$ also honestly generates public/secret key pairs and sketches as Enrollment specified for N-1 users and M servers. Eventually, $\mathcal{F}$ sends all enrolled public keys and references to $\mathcal{A}$. Note that $\mathcal{F}$ honestly generates all the public parameters (including matrixes $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3$) in the system. Also note that $\mathcal{A}$ cannot link the simulated sketch $\mathsf{SS}(w_i, \mathtt{sk}_i^j)$ and public key $\mathtt{pk}^*$ since $\mathcal{A}$ is not allowed to access biometrics w_i.
- Training. $\mathcal{F}$ answers $\mathcal{A}$'s queries as follows.
  - ∗ If $\mathcal{A}$ issues a Send query in the form of n_j to $\mathcal{F}$, $\mathcal{F}$ chooses a response nonce n_i first, then $\mathcal{F}$ simulates the protocol transcript $T_i = (\sigma_i', \mathsf{SS}(w_i, \mathtt{sk}_i'))$ as follows.
    1. invoke the signing oracle $\mathcal{O}$ to obtain a message-signature pair $(msg_{(i,j)}, \sigma_i)$, where $msg_{(i,j)} = (n_i, n_j)$;
    2. generate a sketch $\mathsf{SS}(w_i, \mathtt{sk}_i') \leftarrow \mathsf{SS}(w_i, \mathtt{sk}_i^j) + \mathbb{A}_3 \cdot \Delta(s_i)$, where $\Delta(s_i)$ is randomly chosen by $\mathcal{F}$;
    3. generate a signature $\sigma_i' \leftarrow M_\Sigma(pp, \mathtt{pk}^*, \sigma_i, \Delta(s_i))$ by using the deterministic algorithm M_Σ described in Section 4;
    4. returen $(m_{(i,j)}, \sigma_i', \mathsf{SS}(w_i, \mathtt{sk}_i'))$ to $\mathcal{A}$.
  - ∗ If $\mathcal{A}$ issues a Secret Key Reveal query to an instance oracle $\Pi_{ID_i}^i$, then $\mathcal{F}$ returns new secret key $\mathtt{sk}_i' \in_R \mathbb{Z}_q^{m \times k}$ to $\mathcal{A}$. Since $\mathcal{A}$ is not allowed to reveal the enrolled secret key $Dlog(\mathtt{pk}^*)$ (of user i w.r.t. server j), the simulation is perfect.
  - ∗ If $\mathcal{A}$ issues Biometrics Shift query in the form of δ to $\mathcal{F}$, then $\mathcal{F}$ returns $\mathsf{SS}(w_i + \delta, \mathtt{sk}_i')$ to $\mathcal{A}$. Notice that $\mathcal{A}$ is not allowed to obtain user i's biometrics w_i.
  - ∗ If $\mathcal{A}$ issues Secret Key Shift query in the form of ϕ to $\mathcal{F}$, then $\mathcal{F}$ returns new public key $\mathtt{pk}^{*'} \leftarrow \mathsf{KG}(pp, Dlog(\mathtt{pk}^*), \phi(\Delta(\mathtt{sk})))$, where $\Delta(\mathtt{sk})$ is chosen by $\mathcal{A}$.
- When **Forge** event occurs (i.e., $\mathcal{A}$ outputs $(msg^*, \sigma^{*'}, \mathsf{SS}(w_i, \mathtt{sk}^{*'})))$, $\mathcal{F}$ checks whether:
  - ∗ the **Forge** event happens at g-th session;
  - ∗ the message-signature pair $(msg^*, \sigma^{*'})$ is not previously simulated by $\mathcal{S}$;
  - ∗ verifies $\Sigma.\mathsf{Verify}(\mathtt{pk}^{*'}, msg^*, \sigma^{*'}) \overset{?}{=} 1$, where $\mathtt{pk}^{*'} \leftarrow \mathtt{pk}^* + \mathbb{A}_3^\top \cdot \underline{\Delta(\mathtt{sk}^*)}$, $\underline{\Delta(\mathtt{sk}^*)} = \mathtt{H}_{(\mathbb{A}_1,\mathbb{A}_2)}(\Delta(s^*)), \Delta(s^*) \leftarrow \mathsf{SS}(w_i, \mathtt{sk}^{*'}) - \mathsf{SS}(w_i, \mathtt{sk}_i^j)$. Note that $\underline{\Delta(\mathtt{sk}^*)}$ is the correct "shift" between $Dlog(\mathtt{pk}^{*'})$ and $\mathtt{sk}_i^j$.

  If all the above conditions hold, $\mathcal{F}$ confirms that it as a successful forgery from $\mathcal{A}$, then $\mathcal{F}$ extracts the forgery via $\sigma^* \leftarrow M_\Sigma(pp, \mathtt{pk}^*, \sigma^{*'}, \underline{\Delta(\mathtt{sk}^*)})$ by using the homomorphic property of Σ (Lemma 3). To this end, $\mathcal{F}$ outputs σ^* as its own forgery; Otherwise, $\mathcal{F}$ aborts the game. Therefore, we have

$$|\Pr[\mathbf{Forge}]| \leq \mathtt{Adv}_{\mathcal{F}}^{\text{EUF-CMA}}(\lambda). \tag{6}$$

It is easy to see that in game $\mathbb{G}_5$, $\mathcal{A}$ has no advantage, i.e.,

$$\mathtt{Adv}_5^{\text{RFS-RUA}} = 0. \tag{7}$$

Combining the above results together, we have

$$\begin{aligned}\mathtt{Adv}_{\mathcal{A}}^{\text{RFS-RUA}}(\lambda) \leq N \cdot m(\lambda)^2/2^\lambda + m(\lambda)[N \cdot \mathtt{Adv}_{\mathcal{S}}^{\text{D-LWE}_{q,n-k,m,\chi}}(\lambda) \\ + M \cdot \mathtt{Adv}_{\mathcal{S}}^{\mathbb{H}}(\lambda) + \mathtt{Adv}_{\mathcal{F}}^{\text{EUF-CMA}}(\lambda)].\end{aligned}$$

Theorem 3. *The proposed RFS-RUA achieves user privacy in the random oracle model if the decisional $SIS_{q,n,m,d}$ assumption is (ϵ, s_{sec}) secure, the public key encryption is IND-CPA secure and the computational fuzzy extractor is IK-CPA secure.*

Proof. We define a sequence of games $\{\mathbb{G}_i\}$ and let $\mathtt{Adv}_i^{\text{RFS-RUA}}$ denote the advantage of the adversary $\mathcal{A}$ in game $\mathbb{G}_i$. We also highlight the differences between adjacent games by underline.

– $\mathbb{G}_0$ This is the original game for user privacy.
– $\mathbb{G}_1$ This game is identical to game $\mathbb{G}_0$ except that at challenge stage, $\mathcal{S}$ replaces the real identity (message of ciphertext C_i) by random string R. Below we show that the difference between $\mathbb{G}_0$ and $\mathbb{G}_1$ is negligible under the assumption that the public key encryption scheme is IND-CPA secure.
  Let $\mathcal{S}$ denote an attacker who is given a public key $\mathtt{pk}^*$, aims to break the IND-CPA security of the public key encryption scheme. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.
  - Setup. $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating N users and M servers. $\mathcal{S}$ honestly generates biometrics for N users, generates secret/public key pairs with respect to M servers and their corresponding sketches. $\mathcal{S}$ randomly selects index j and guesses the challenge will happen with regard to server j. $\mathcal{S}$ sets the public key of server j as $\mathtt{pk}^*$, and honestly generates secret/public key pairs for M-1 servers. It is obvious that $\mathcal{S}$ can easily simulate the protocol execution of N users and M-1 servers except server j. Below we mainly focus on the simulation of server j.
  - Training. If $\mathcal{A}$ issues an Execute query between user i and server j, then $\mathcal{S}$ randomly chooses nonces (n_i, n_j) and new secret/public key pairs, and performs the session execution honestly according to the protocol specification.
  - Challenge. Upon receiving challenge candidates $(ID_0, ID_1) \in \mathcal{U}'_0$ from $\mathcal{A}$, $\mathcal{S}$ follows the security game to select ID_b^*. Then $\mathcal{S}$ executes the RFS-RUA protocol to generate the protocol transcript. After that, $\mathcal{S}$ generates another random string $\underline{R}$ and sends ID_b and $\underline{R}$ as the challenge messages to its own oracle. After receiving the challenge ciphertext $\underline{C^*}$ from his own challenger, $\mathcal{S}$ replaces the ciphertext generated by ID_b in the first message of the transcript by $\underline{C^*}$. Eventually, $\mathcal{S}$ sends the complete transcript to $\mathcal{A}$.

  Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break the IND-CPA security of public key encryption scheme. Since at most M servers in the system, and at most $\mathbb{K} = \{0,1\}^{|ID|}$ encryptions are executed (because each ciphertext encrypts a single bit of real identity), we have

$$\left|\mathtt{Adv}_0^{\text{RFS-RUA}} - \mathtt{Adv}_1^{\text{RFS-RUA}}\right| \leq M \cdot \mathbb{K} \cdot \mathtt{Adv}_{\mathcal{S}}^{\text{IND-CPA}}(\lambda). \tag{8}$$

– $\mathbb{G}_2$ This game is identical to game $\mathbb{G}_1$ except that at challenge stage, $\mathcal{S}$ replaces the sketch $\mathsf{SS}(w_i', \mathtt{sk}_i')$ by a random string over $\mathbb{Z}_q^m$. Below we show that the difference between $\mathbb{G}_1$ and $\mathbb{G}_2$ is negligible under the assumption that computational fuzzy extractor is IK-CPA secure.
Let $\mathcal{S}$ denote an attacker who is given two sketches $(\mathtt{pk}_0, \mathtt{pk}_1)$ (assuming a common public matrix $\mathbb{A}$ here, and $\mathtt{pk} \leftarrow \mathbb{A} \cdot X + \chi$), aims to break the IK-CPA security of the computational fuzzy extractor. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.
  - Setup. $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating N users and M servers. $\mathcal{S}$ honestly generates secret/public key pairs for M servers. $\mathcal{S}$ randomly selects indexes (i, j) and guesses the challenge will happen with regard to user i and server j. $\mathcal{S}$ sets the enrolled sketch of user i with respect to server j as $\mathtt{pk}_0$, and generates the enrolled secret/public key pair at random (which is a matrix pair over distribution $(\mathbb{Z}_q^{m\times k}, \mathbb{Z}_q^{n\times k})$). Additionally, $\mathcal{S}$ honestly generates biometrics, secret/public key pairs (w.r.t. M servers) and sketches for N-1 users. It is obvious that $\mathcal{S}$ can easily simulate all protocol executions except user i w.r.t server j. Below we mainly focus on the simulation between user i and server j. Note that $\mathbb{A}_3^\top = \underline{\mathbb{A}^\top}$.
  - Training. If $\mathcal{A}$ issues an Execute query, then $\mathcal{S}$ simulates the session execution as follows
    1. generate a ciphertext C_i on the identity of user i;
    2. choose nonces n_i, n_j and form a message $msg_i = (n_i, n_j)$;
    3. choose secret/public key pair $(\mathtt{sk}_i', \mathtt{pk}_i')$ and generate message-signature pair using secret key $\mathtt{sk}_i'$;
    4. compute "shift" $\Delta(\mathtt{sk}_i)$ from (enrolled public key) $\mathtt{pk}_i^j$ and $\mathtt{pk}_i'$;
    5. simulate a new sketch $\mathsf{SS}(w_i, \mathtt{sk}_i') = \mathtt{pk}_0 + \mathbb{A}^\top \cdot \Delta(\mathtt{sk}_i)$;
    6. sends the complete transcript to $\mathcal{A}$.
  - Challenge. Upon receiving challenge candidates $(ID_0, ID_1) \in \mathcal{U}_0'$ from $\mathcal{A}$, $\mathcal{S}$ follows the security game to select ID_b^* and server j (from honest set $\mathcal{U}_1'$). Then $\mathcal{S}$ executes the RFS-RUA protocol to generate the protocol transcript. After that, $\mathcal{S}$ generates the challenge message msg^* (msg^* derives from the "shift" between enrolled sketch and new sketch generated by ID_b^*) and sends it as the challenge message to its own oracle. After receiving the challenge ciphertext $\underline{C^*}$ from his own challenger, $\mathcal{S}$ replaces the sketch generated by ID_b in the third message of the transcript by $\underline{C^*}$. Eventually, $\mathcal{S}$ sends the complete transcript to $\mathcal{A}$.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break the IK-CPA security of the computational fuzzy extractors. Since at most N users and M servers in the system, we have

$$\left|\mathtt{Adv}_1^{\text{RFS-RUA}} - \mathtt{Adv}_2^{\text{RFS-RUA}}\right| \leq N \cdot M \cdot \mathtt{Adv}_{\mathcal{S}}^{\text{IK-CPA}}(\lambda). \tag{9}$$

– $\mathbb{G}_3$ This game is identical to game $\mathbb{G}_2$ except that at challenge stage, $\mathcal{S}$ replace the real verification key of digital signature σ_i by a random key over $\mathbb{Z}_q^{n\times k}$. Below we show that the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible under the assumption that the decisional $\mathrm{SIS}_{q,n,m,d}$ is hard.
Let $\mathcal{S}$ denote a decisional SIS problem distinguisher, who is given a pair $(\mathbb{A}^*, t^*)$, aims to decide it whether from a $\mathrm{SIS}_{q,n,m,d}$ distribution or from a random distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.
  - Setup: $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating N users and M servers with the corresponding identities and biometrics. $\mathcal{S}$ generates enrolled public/secret key pairs $(\mathtt{pk}_i^j, \mathtt{sk}_i^j)$ and sketches $\mathsf{SS}(w_i, \mathtt{sk}_i^j)$ for N users and M servers. Eventually, $\mathcal{S}$ sends all identities, public keys and sketches to $\mathcal{A}$. In particular, $\mathcal{S}$ sets $\mathbb{A}_3^\top = \underline{\mathbb{A}^{*\top}}$ and honestly generates other public parameters (such as matrixes $\mathbb{A}_1, \mathbb{A}_2$) for the system. Note that $\mathtt{pk}_i^j = \mathbb{A}^{*\top} \cdot \mathtt{sk}_i^j$.
  - Training: $\mathcal{S}$ honestly simulates the session execution (using user's secret keys and biometrics) according to the protocol specification.
  - Challenge: Upon receiving challenge candidates $(ID_0, ID_1) \in \mathcal{U}_0'$ from the $\mathcal{A}$, $\mathcal{S}$ follows the security game to select ID_b^* and server j. $\mathcal{S}$ simulates the the transcript as follows.
    1. generate a ciphertext C_i on the identity of user i;
    2. choose $s' \in_R \mathbb{Z}_q^n$ and compute a new sketch $\mathsf{SS}(w_b, \mathtt{sk}_b') \leftarrow \mathbb{A}^* \cdot s' + w_b$;
    3. choose nonces n_i, n_j and form a message $msg_{(i,j)} = (n_i, n_j)$;
    4. generate a message-signature pair $(msg_{(i,j)}, \sigma_b^*)$ (by design, $\sigma_b^* = (z_b^*, c_b^*)$) using the same method described in the security proof of [25]; Note that the corresponding public (verification) key is $\mathtt{pk}_b^{*'} = \underline{U^*} + \mathbb{A}_3^\top \cdot \Delta(\mathtt{sk})$, where $\Delta(\mathtt{sk})$ derives from public sketches.
    5. return $(C_i, msg_{(i,j)}, \underline{\sigma_b^*}, \mathsf{SS}(w_b, \mathtt{sk}_b'))$ as the complete transcript.

  Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. Notice that the enrolled public key $\mathtt{pk}_b$ is replaced by the given instance $\underline{U^*}$ in the Challenge stage. Moreover, the simulated message-signature pair $(msg_{(i,j)}, \sigma_b^*)$ can be successfully verified under verification key $\mathtt{pk}_b^{*'}$ (by programming random oracle such that $\mathtt{H}(\mathbb{A}^* \cdot z_b^* - \mathtt{pk}_b^{*'} \cdot c_b^*, msg_{(i,j)}) = c_b^*$). As for the matrix pair $(\mathbb{A}^*, U^*)$, according to the hybrid argument, distinguishing the real public key from a uniform distribution $(\mathbb{A}^*, U^*) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times k}$ is *as hard as* the $\mathrm{SIS}_{q,n,m,d}$ decisional problem (*with a loss of factor k in the advantage*). Therefore, the two verification keys $\mathtt{pk}_b^{*'}$ $(b = [0,1])$ are statistically indistinguishable w.r.t. high-density SIS or computational indistinguishable w.r.t. low-density SIS. Since at most N users and M servers in the system, we have

$$\left|\mathtt{Adv}_2^{\text{RFS-RUA}} - \mathtt{Adv}_3^{\text{RFS-RUA}}\right| \le N \cdot M \cdot k \cdot \mathtt{Adv}_{\mathcal{S}}^{\mathrm{SIS}_{q,n,m,d}}(\lambda). \tag{10}$$

It is easy to see that in game $\mathbb{G}_3$, $\mathcal{A}$ has no advantage, i.e.,

$$\mathtt{Adv}_3^{\text{RFS-RUA}} = 0. \tag{11}$$

Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\text{RFS-RUA}}(\lambda) \le M \cdot \mathbb{K} \cdot \mathtt{Adv}_{\mathcal{S}}^{\text{IND-CPA}}(\lambda) + N \cdot M \cdot (\mathtt{Adv}_{\mathcal{S}}^{\text{IK-CPA}}(\lambda) + k \cdot \mathtt{Adv}_{\mathcal{S}}^{\mathrm{SIS}_{q,n,m,d}}(\lambda)).$$

6 Conclusion

In this work, we have proposed a lattice-based construction for remote user authentication from RFS. We have introduced a reusability model that allows fuzzy signatures to be reusable, and provided the reusability proof of RFS. In addition, the RFS based remote user authentication had the privacy guarantee with respect to the eavesdroppers. Eventually, we have proved the security of the proposed construction in the random oracle model under our defined security models (user authenticity and user privacy). As our future work, we leave the construction of RFS based on efficient ring-SIS or ring-LWE [18, 28].

References

1. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.
2. D. Apon, C. Cho, K. Eldefrawy, and J. Katz. Efficient, reusable fuzzy extractors from LWE. In *International Conference on Cyber Security Cryptography and Machine Learning*, pages 1–18, 2017.
3. S. Bai and S. D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014.
4. M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, et al. Privacy-preserving fingercode authentication. In *Proceedings of the 12th ACM workshop on Multimedia and security*, pages 231–240, 2010.
5. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, pages 566–582, 2001.
6. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT*, pages 259–274, 2000.
7. M. Bellare, D. Cash, and R. Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT*, pages 486–503, 2011.
8. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, pages 139–155, 2000.
9. X. Boyen. Reusable cryptographic fuzzy extractors. In *ACM CCS*, pages 82–91, 2004.
10. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. D. Smith. Secure remote authentication using biometric data. In *EUROCRYPT*, volume 3494, pages 147–163, 2005.
11. R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. D. Smith. Reusable fuzzy extractors for low-entropy distributions. In *EUROCRYPT*, pages 117–146, 2016.
12. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540, 2004.
13. N. Döttling and J. Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In *EUROCRYPT*, pages 18–34, 2013.
14. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO*, pages 40–56. 2013.
15. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
16. B. Fuller, X. Meng, and L. Reyzin. Computational fuzzy extractors. In *ASIACRYPT*, pages 174–193, 2013.

17. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.
18. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, pages 530–547, 2012.
19. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of a pseudorandom generator from any one-way function. In *SIAM*, 1993.
20. A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Fingercode: a filterbank for fingerprint representation and matching. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 187–193, 1999.
21. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, pages 143–154, 1996.
22. A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *ACM CCS*, pages 28–36, 1999.
23. J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM*, 36(5):1231–1247, 2006.
24. N. Li, F. Guo, Y. Mu, W. Susilo, and S. Nepal. Fuzzy extractors for biometric identification. In *ICDCS*, pages 667–677, 2017.
25. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.
26. T. Matsuda, K. Takahashi, T. Murakami, and G. Hanaoka. Fuzzy signatures: relaxing requirements and a new construction. In *ACNS*, pages 97–116, 2016.
27. D. Micciancio. Lattice-based cryptography. In *Encyclopedia of Cryptography and Security*, pages 713–715. 2011.
28. C. Peikert. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.
29. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *JACM*, 56(6):34, 2009.
30. C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.
31. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
32. K. Takahashi, T. Matsuda, T. Murakami, G. Hanaoka, and M. Nishigaki. A signature scheme with a fuzzy private key. In *ACNS*, pages 105–126, 2015.
33. K. Takahashi, T. Matsuda, T. Murakami, G. Hanaoka, and M. Nishigaki. Signature schemes with a fuzzy private key. *IACR Cryptology ePrint Archive*, 2017:1188, 2017.
34. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.
35. Y. Wen and S. Liu. Robustly reusable fuzzy extractor from standard assumptions. In *ASIACRYPT*, pages 459–489, 2018.
36. Y. Wen, S. Liu, and S. Han. Reusable fuzzy extractor from the decisional diffie–hellman assumption. *Designs, Codes and Cryptography*, 86(11):2495–2512, 2018.
37. 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 *ISPEC*, pages 409–421, 2017.

A Privacy Concerns on [32, 26, 33]

We assume the $\mathsf{Enrollment}$ stage is also executed in a secure channel, which means the attackers cannot access user's enrolled public keys. According to the generic construction in [26] (also applicable to [32]), we assume an authorized user Alice wants to authenticate herself to an authentication server, and produces a transcript $\underline{(\mathtt{pk}'_A, \mathsf{SS}(w'_A, \mathtt{sk}'_A), \sigma'_A)}$ in one session, where the new public key is $\mathtt{pk}'_A = g^{\mathtt{sk}'_A}$. In another session, suppose a challenge identity ID_b^* generates a transcript $(\mathtt{pk}^*_{ID_b}, \mathsf{SS}(w^*_{ID_b}, \mathtt{sk}^*_{ID_b}), \sigma^*_{ID_b})$, where the new public key is $\mathtt{pk}^*_{ID_b} = g^{\mathtt{sk}^*_{ID_b}}$. Then attacker can *link* the challenge identity ID_b^* to user Alice. Specifically, an attacker will verify the signature $\sigma^*_{ID_b}$ using the new public key $\mathtt{pk}^*_{ID_b}$ first; then compute the "difference" $\Delta(\mathtt{sk}^*_{ID_b}) \leftarrow \mathsf{SS}(w^*_{ID_b}, \mathtt{sk}^*_{ID_b}) - \mathsf{SS}(w'_A, \mathtt{sk}'_A)$; eventually, attachers can verify the relationship between new public keys $\mathtt{pk}^*_{ID_b}$ and $\mathtt{pk}'_A$ via a "difference reconstruction" algorithm $M_{\mathtt{pk}'_A}(\mathtt{pk}'_A, \Delta(\mathtt{sk}^*_{ID_b})) \overset{?}{=} \mathtt{pk}^*_{ID_b}$. Notice that if algorithm $M_{\mathtt{pk}'_A}$ outputs 1, then attacker can conclude that the challenge user ID_b is user Alice. *The key point here is that the new public key of user Alice $\mathtt{pk}'_A$ in one session actually acts as an enrolled public key, and can be easily linked with the new public key $\mathtt{pk}'_{ID_b}$ in another session of the same user, assuming user Alice (with biometrics w'_A and $\mathsf{dist}(w'_A, w^*_{ID_b}) \leq t$) is successfully authenticated by an authentication server.*

Our Treatment. We modify the verification process, in which the verification of signature requires the enrolled public key of user Alice $\mathtt{pk}_A$. We follow the example above, user Alice first generates her new secret key $\mathtt{sk}'_A$ for signing a message, then computes a new sketch $\mathsf{SS}(w'_A, \mathtt{sk}'_A)$. The user Alice sends the transcript $\underline{(\mathsf{SS}(w'_A, \mathtt{sk}'_A), \sigma'_A)}$ to an authentication server. The authentication server first obtains the "difference" value $\Delta(\mathtt{sk}_A)$ from enrolled and new sketches, then computes her new public key $\mathtt{pk}'_A \leftarrow M_{\mathtt{pk}_A}(\mathtt{pk}_A, \Delta(\mathtt{sk}_A))$ and verifies the message-signature pair (m, σ'_A) using the corresponding new public key $\mathtt{pk}'_A$. Notice that the major difference between two transcripts (underline parts) is: *our treatment does not transmit new public key in the public channel.* In fact, the verification of message-signature pair is based on the designated verifier concept [21] such that the authentication server who holds the enrolled public key can verify it. We stress that the *publicly verifiable* of message-signature pair is the main privacy concern in [32, 26, 33].