Generic Constructions of Robustly Reusable Fuzzy Extractor

Yunhua Wen[1], Shengli Liu[1,2,3]✉, and Dawu Gu[1]

[1] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
`{happyle8, slliu, dwgu}@sjtu.edu.cn`
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Westone Cryptologic Research Center, Beijing 100070, China

Abstract. Robustly reusable Fuzzy Extractor (rrFE) considers reusability and robustness simultaneously. We present two approaches to the generic construction of rrFE. Both of approaches make use of a secure sketch and universal hash functions. The first approach also employs a special pseudo-random function (PRF), namely unique-input key-shift (ui-ks) secure PRF, and the second uses a key-shift secure auxiliary-input authenticated encryption (AIAE). The ui-ks security of PRF (resp. key-shift security of AIAE), together with the homomorphic properties of secure sketch and universal hash function, guarantees the reusability and robustness of rrFE. Meanwhile, we show two instantiations of the two approaches respectively. The first instantiation results in the first rrFE from the LWE assumption, while the second instantiation results in the first rrFE from the DDH assumption over non-pairing groups.

Keywords: Fuzzy Extractor; Reusability; Robustness

1 Introduction

In cryptographic applications, the underlying secret keys are required to be uniformly sampled and reproducible. Uniformity of the secret keys is necessary for the security of cryptographic algorithms, and reproducibility is responsible for the correctness of the algorithms. In reality, there exist many noisy random sources of high entropy, which are neither uniformly distributed nor reproducible. For instance, biometrics [19, 20] (like fingerprint, iris, voice, etc), physical unclonable functions [22, 23] in electronic devices, and quantum bits generated from quantum devices [4, 5]. In fact, the readings of the same source are rarely identical and noises are inevitably introduced in each reading. An interesting topic is research on converting such random sources into uniform and reproducible strings so that they can serve as secret keys for cryptographic systems. The topic was highlighted by Dodis et al. [11] who proposed the concept of Fuzzy Extractor.

Fuzzy extractor (FE) is able to turn a noisy variable of high entropy into a stable, uniformly distributed string. More precisely, it consists of two efficient algorithms $(\mathsf{Gen}, \mathsf{Rep})$. The generation algorithm Gen on input a reading w from a noisy source W outputs a public helper string P together with an extracted string R. The security of FE requires that R is (pseudo-)random if W has enough entropy. The reproduction algorithm Rep on input w' which is close to w will reproduce R with the help of the public helper string P.

Reusable Fuzzy Extractor. It should be noted that fuzzy extractor only allows one extraction from a noisy source. This feature limits the usability of fuzzy extractor. In fact, a user may like to use his/her fingerprint to generate several keys for different cryptographic applications. To this end, reusable fuzzy extractor was proposed by Boyen [6]. Generally, reusable fuzzy extractor guarantees the security of multiple keys extracted from a single noisy source. More precisely, $\mathsf{R}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q$ are all pseudorandom even conditioned on $(\mathsf{P}_1, \mathsf{P}_2, \cdots, \mathsf{P}_Q)$ where $(\mathsf{P}_j, \mathsf{R}_j) \leftarrow \mathsf{Gen}(w_j)$, $j \in \{1, \cdots, Q\}$ and w_j is the j-th reading of a noisy source.

In [6], Boyen proposed a reusable FE scheme based on the random oracle. In the security model, it assumes the exclusive OR of two readings of the same source reveals no information of the noisy source W. Wen et al. [26] constructed a reusable FE from the Decisional Diffie-Hellman (DDH) assumption, and the security model assumes that the difference of two readings of the same source does not reveal too much information of the random source.

Canetti et al. [8] constructed a reusable FE from a powerful tool named "digital locker". In the security model, no assumption is made on how multiple readings are correlated. However, existing instantiations of digital locker rely their security either on random oracles or non-standard assumptions. Their work was upgraded by Alamélou et.al. [1] to tolerate linear fraction of errors, but still rely on "digital locker".

Apon et al. [2] proposed a reusable FE from the learning with errors (LWE) assumption, but it can only tolerate logarithmic fraction of errors. Later, Wen et al. [24] constructed a new reusable FE from LWE assumption tolerating linear fraction of errors. In both works, it assumes that the differences between two readings of the same source are controlled by a probabilistic polynomial-time (PPT) adversary.

Robust Fuzzy Extractor. Fuzzy extractor does not consider active adversaries. If the public helper string P is modified by an active adversary, the correctness of fuzzy extractor might not be guaranteed. Boyen et al. [7] first highlighted this issue and introduced the concept of robust fuzzy extractor. Robustness of fuzzy extractor concerns the integrity of P, and requires that the reproduction algorithm of FE will output $\perp$ with overwhelming probability if P is modified.

Boyen et al. [7] proposed a generic way of transforming a fuzzy extractor to a robust one based on random oracles. Dodis et al. [10] showed that robustness of information-theoretic fuzzy extractor is not achievable in the *plain model* if the entropy rate of the source is less than $1/2$ and they constructed a fuzzy extractor with post-application robustness which applies to sources of entropy rate larger than 2/3. Later, Kanukurthi et al. [16] introduced an improved robust FE, which relaxes entropy rates of sources to be larger than 1/2. With the help of *common reference string* (CRS), Cramer et al. [9] proposed a robust FE and breaks the 1/2 entropy rate barrier in the CRS model.

Robustly Reusable Fuzzy Extractor. Most recently, Wen et al. [25] proposed the concept of robustly reusable Fuzzy Extractor (rrFE), which considers robustness and reusability simultaneously in the CRS model.

According to [25], the reusability of rrFE asks the pseudorandomness of R_j even conditioned on $(\mathsf{R}_1, \cdots, \mathsf{R}_{j-1}, \mathsf{R}_{j+1}, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$ where $(\mathsf{P}_j, \mathsf{R}_j) \leftarrow \mathsf{Gen}(w_j = w + \delta_j)$, $j \in [Q]$ ($[Q] := \{1, \cdots, Q\}$) and δ_j is controlled by the adversary. In formula, $(\mathsf{R}_1, \cdots, \mathsf{R}_j, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q) \approx_c (\mathsf{R}_1, \cdots, \mathsf{U}_j, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$, where U_j denotes a uniform distribution. In fact, a stronger version requires $(\mathsf{R}_1, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q) \approx_c (\mathsf{U}_1, \cdots, \mathsf{U}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$. The robustness of rrFE requires that for any PPT adversary, its forged public helper string and reading shift (P^*, δ^*) cannot pass the reproduction algorithm of rrFE except with negligible probability, even if the adversary sees $(\mathsf{P}_j, \mathsf{R}_j)_{j \in [Q]}$. Here $(\mathsf{P}_j, \mathsf{R}_j) \leftarrow \mathsf{Gen}(w_j = w + \delta_j)$. Moreover, $\{\delta_j\}_{j \in [Q]}, (\mathsf{P}^*, \delta^*) \notin \{(\mathsf{P}_j, \mathsf{R}_j)\}_{j \in [Q]}$ are adaptively chosen by the adversary.

In [25], the first robustly reusable fuzzy extractor was constructed based on the DDH and DLIN assumptions in the CRS model. We stress that the DLIN assumption is over pairing-friendly groups, since a core building block of their construction, namely homomorphic lossy algebraic filter (LAF) [15], has only one instantiation, which is over (symmetric) pairing-friendly groups. Though the construction is elegant, the instantiation of LAF introduces big public helper string, and complicated computations over symmetric pairing groups in rrFE.

Question. Is there any other approaches to rrFE? Is it possible to obtain a more efficient rrFE? Is it possible to construct a rrFE from the LWE assumption?

1.1 Our Contribution

We answer the above questions in the affirmative.

- We provide two generic constructions of rrFE. Namely,

$$\mathsf{SS} + \mathcal{H}_{\mathcal{I}} + \mathsf{ui\text{-}ks\text{-}PRF} \Rightarrow \mathsf{rrFE},$$
$$\mathsf{SS} + \mathcal{H}_{\mathcal{I}} + \mathsf{AIAE} \Rightarrow \mathsf{rrFE},$$

  where SS is a homomorphic Secure Sketch with linearity property, $\mathcal{H}_{\mathcal{I}}$ is a family of homomorphic universal hash functions, $\mathsf{ui\text{-}ks\text{-}PRF}$ is a pseudorandom function with unique-input key-shift security, and AIAE is an auxiliary-input authenticated encryption with key-shift security.
- Our construction is simple and can be instantiated with standard assumptions. Both SS and $\mathcal{H}_{\mathcal{I}}$ have information-theoretic instantiations, and $\mathsf{ui\text{-}ks\text{-}PRF}$ and AIAE have available instantiations from standard assumptions.

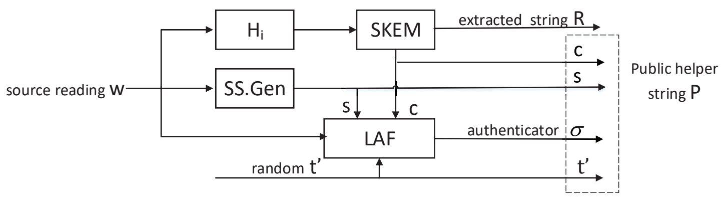

Fig. 1. The generation algorithm of the robustly reusable Fuzzy Extractor in [25].

- When instantiating $\mathsf{ui\text{-}ks\text{-}PRF}$ with the PRF constructed in [18], we obtain the first post-quantum rrFE from the LWE assumption.
- When instantiating AIAE with the AIAE scheme from [14], we obtain the first efficient rrFE from the DDH assumption over pairing-free groups.

1.2 Our Approaches

We provide two new approaches to rrFE. Following the security model in [25], the adversary controls the differences between any two different readings of the source W.

First, we recall the generation algorithm of rrFE in [25] in Fig. 1. The source reading w is served as inputs for three building blocks, i.e., universal hash function $\mathsf{H_i}$, secure sketch scheme $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ and lossy algebraic filter LAF. With $\mathsf{H_i}$, a string k is extracted from source w and used as a key in the symmetric KEM $\mathsf{SKEM} = (\mathsf{SKEM.Enc}, \mathsf{SKEM.Dec})$, which in turn encapsulates the final extracted string R; with $\mathsf{SS.Gen}$, a secure sketch s is generated from w to help eliminate noises in the reproduction algorithm of rrFE; with LAF, w is used as an authentication key to authenticate the ciphertext c generated by $\mathsf{SKEM.Gen}$, the secure sketch s and a random tag t'. The output σ of LAF can be regarded as an authenticator. The final public helper string is $\mathsf{P} = (\mathsf{c}, \mathsf{s}, \mathsf{t}', \sigma)$.

Differences between ours and [25]. Different from the rrFE in [25], we explore a different structure with different primitives. As for primitive, we use pseudorandom function (PRF) or auxiliary-input authentication encryption (AIAE), instead of LAF+SKEM. As for structure, rrFE in [25] uses LAF for authentication of $(\mathsf{c}, \mathsf{s}, \mathsf{t}')$ and SKEM for pseudo-randomness of R, while ours employs only a single primitive $\mathsf{ui\text{-}ks\text{-}PRF}$(or AIAE) to achieve both authentication and pseudo-randomness. Moreover, we do not use w directly as authentication key. Instead, we input w to $\mathsf{H_i}$ to obtain a key k for $\mathsf{ui\text{-}ks\text{-}PRF/AIAE}$. We expect $\mathsf{ui\text{-}ks\text{-}PRF/AIAE}$ to provide both pseudorandomness of R and authentication of the public helper string P. In fact, the security of the PRF $\mathsf{ui\text{-}ks\text{-}PRF/AIAE}$ helps us to obtain reusability and robustness of rrFE. See Fig. 2 and Fig. 3.

The First Approach. In the first approach, we resort to a special PRF, namely $\mathsf{ui\text{-}ks\text{-}PRF}$. Taking the output k from $\mathsf{H_i}$ as its key, and the output s from SS and a random t as its input, $\mathsf{ui\text{-}ks\text{-}PRF}$ outputs a string which is further divided into

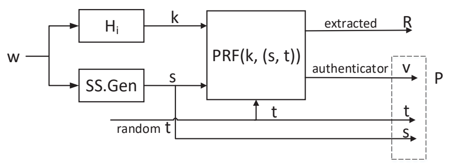

Fig. 2. Gen of the first approach.

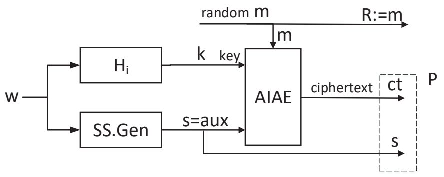

Fig. 3. Gen of the second approach.

R and v. Here R is the final extracted string, while v behaves as an authenticator. The public helper string is $\mathsf{P} = (\mathsf{s}, \mathsf{t}, \mathsf{v})$.

The reusability and robustness of rrFE can be reduced to the Unique-Input Key-Shift ($\mathsf{ui\text{-}ks}$) security of $\mathsf{ui\text{-}ks\text{-}PRF}$, with the help of the homomorphic properties of $\mathsf{H_i}$ and SS (and also the linearity property of SS). Informally, the security of the PRF $\mathsf{ui\text{-}ks\text{-}PRF}$ requires

$$(x_j, \mathsf{PRF}(\mathsf{k} + \Delta_j, x_j))_{j\in[Q]} \approx_c (x_j, \mathsf{U}_j)_{j\in[Q]}$$

for all PPT adversaries, where k, U_j are uniformly distributed, PRF is the evaluation algorithm of $\mathsf{ui\text{-}ks\text{-}PRF}$, inputs $\{x_j\}_{j\in[Q]}$ are distinct, and $\{x_j\}_{j\in[Q]}$ $\{\Delta_j\}_{j\in[Q]}$ are adaptively chosen by the adversary.

Now we outline the high-level idea of proving the reusability and robustness of our first rrFE in Fig. 2.

(i) Due to the homomorphic properties of SS and $\mathsf{H_i}$, it holds that $\mathsf{H_i}(w_j) = \mathsf{H_i}(w+\delta_j) = \mathsf{H_i}(w)+\mathsf{H_i}(\delta_j)$ and $\mathsf{SS.Gen}(w_j) = \mathsf{SS.Gen}(w+\delta_j) = \mathsf{SS.Gen}(w)+\mathsf{SS.Gen}(\delta_j)$. Then the view of the adversary can be considered as the randomized function of $\mathsf{k}(:= \mathsf{H_i}(w))$, $\mathsf{s}(:= \mathsf{SS.Gen}(w))$. The output s from SS only leaks limited amount of information of W. By the leftover hash lemma, the output k from universal hash function $\mathsf{H_i}$ is uniform and independent of s.

(ii) The key of $\mathsf{ui\text{-}ks}$ PRF is given by $\mathsf{k}_j = \mathsf{H_i}(w_j) = \mathsf{H_i}(w + \delta_j) = \mathsf{k} + \mathsf{H_i}(\delta_j)$, which can be regarded as a key shifted $\Delta_j := \mathsf{H_i}(\delta_j)$ from k. The inputs of $\mathsf{ui\text{-}ks}$ PRF are $(\mathsf{s}_j, \mathsf{t}_j)_{j\in[Q]}$, which are distinct with each other due to the randomness of t_j. Given that k is uniform and independent of s, key shift Δ_j is determined by δ_j, and all input $(\mathsf{s}_j, \mathsf{t}_j)$ are distinct, it is ready for us to implement the security reduction of rrFE to $\mathsf{ui\text{-}ks}$ security of $\mathsf{ui\text{-}ks}$ PRF. The security reduction is non-trivial (see Sect. 4 for details).

(iii) The $\mathsf{ui\text{-}ks}$ security of $\mathsf{ui\text{-}ks}$ PRF implies the pseudo-randomness of $(\mathsf{R}_j, \mathsf{v}_j)$ and $(\mathsf{R}^*, \mathsf{v}^*)$ for $\mathcal{A}$, which immediately implies reusability of rrFE. The robustness of rrFE follows as well, since the adversary cannot guess the correct authenticator v^* with non-negligible probability. The security reduction is non-trivial (see Sect. 4 for details).

The Second Approach. In the second approach, we use a special authenticated encryption scheme, namely auxiliary-input authenticated encryption (AIAE).

Taking $\mathsf{k} := \mathsf{H_i}(w)$ as its key and $\mathsf{s} := \mathsf{SS.Gen}(w)$ as its auxiliary input, AIAE encrypts a random string R and outputs a ciphertext ct. Then R serves as the final extracted string, while $\mathsf{P} = (\mathsf{s}, \mathsf{ct})$ as the public helper string.

As a symmetric encryption, the Key-Shift security AIAE asks both IND-RKA and (weak) INT-RKA security. The IND-RKA security requires

$$\begin{aligned}&(\mathsf{m}_{j,0}, \mathsf{m}_{j,1}, \mathsf{ct}_{j,0} \leftarrow \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{m}_{j,0}, \mathsf{aux}_j))_{j \in [Q]}\\ \approx_c\ &(\mathsf{m}_{j,0}, \mathsf{m}_{j,1}, \mathsf{ct}_{j,1} \leftarrow \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{m}_{j,1}, \mathsf{aux}_j))_{j \in [Q]},\end{aligned}$$

where $(\mathsf{m}_{j,0}, \mathsf{m}_{j,1}), \varDelta_j$ are adaptively chosen by PPT adversaries. It implies that

$$(\mathsf{R}_j, \mathsf{ct}_j)_{j \in [Q]} \approx_c \left(\mathsf{R}_j, \mathsf{ct}'_j\right)_{j \in [Q]},$$

where $\mathsf{R}_j, \mathsf{R}'_j$ are uniformly chosen and $\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{R}_j, \mathsf{aux}_j)$, $\mathsf{ct}'_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{R}'_j, \mathsf{aux}_j)$. The weak INT-RKA security requires that given $\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{m}_j, \mathsf{aux}_j)$ with $(\varDelta_j, \mathsf{m}_j, \mathsf{aux}_j)$ chosen by the adversary, it is hard for the PPT adversary to forge a new tuple $(\mathsf{aux}^*, \mathsf{ct}^*, \varDelta^*)$ such that $\mathsf{AIAE.Dec}(\mathsf{k} + \varDelta^*, \mathsf{ct}^*, \mathsf{aux}^*) \neq \perp$. Here a special rule is imposed: $\varDelta^* = \varDelta_j$ if $\mathsf{aux}^* = \mathsf{aux}_j$.

The reusability and robustness of rrFE can be reduced to the Key-Shift security of AIAE, thanks to the homomorphic properties of $\mathsf{H_i}$ and SS and the linearity property of SS.

(i)' With the same reason as in (i), $\mathsf{H_i}(w)$ outputs a uniform key k, which is independent of s.

(ii)' The key of AIAE is $\mathsf{k}_j = \mathsf{H_i}(w_j) = \mathsf{H_i}(w + \delta_j) = \mathsf{k} + \mathsf{H_i}(\delta_j)$, which can be regarded as a key shifted $\varDelta_j := \mathsf{H_i}(\delta_j)$ from k. The message of AIAE is a random string R, the auxiliary input is $\mathsf{s}_j := \mathsf{s} + \mathsf{SS.Gen}(\delta_j)$ and the corresponding ciphertext is $\mathsf{ct}_j := \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{R}, \mathsf{s}_j)$. Given that k is uniform and independent of s, key shift $\varDelta_j$ is determined by δ_j, it is ready for us to implement the reusability security reduction of rrFE to IND-RKA security of AIAE. The IND-RKA security of AIAE guarantees that $(\mathsf{R}_j, \mathsf{ct}_j)_{j \in [Q]} \approx_c \left(\mathsf{R}_j, \mathsf{ct}'_j\right)_{j \in [Q]}$, where $\mathsf{R}_j, \mathsf{R}'_j$ are uniformly chosen and $\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{R}_j, \mathsf{aux}_j)$, $\mathsf{ct}'_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k} + \varDelta_j, \mathsf{R}'_j, \mathsf{aux}_j)$. This suggests that the extracted string R_j's are pseudo-random, hence reusability of rrFE follows. The security reduction is non-trivial (see Sect. 5 for details).

(iii)' As for robustness, let $(\mathsf{P}^* = (\mathsf{s}^*, \mathsf{ct}^*), \delta^*)$ be the forged pair by a PPT adversary. If $\mathsf{aux}^* = \mathsf{s}^* = \mathsf{s}_j = \mathsf{aux}_j$, then the correctness of SS means $w^* = w_j$, hence the keys $\mathsf{k}^* = \mathsf{H_i}(w^*) = \mathsf{H_i}(w + \delta^*) = \mathsf{H_i}(w_j) = \mathsf{H_i}(w + \delta_j) = \mathsf{k}_j$, i.e., $\varDelta^* = \varDelta_j$. As a result, the special rule is satisfied. The secure sketch scheme SS is required to be linear so that there exists an efficient function g to compute $\widetilde{\delta^*} = g(\mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*)$ such that $\varDelta^* := \mathsf{H_i}(\widetilde{\delta^*})$. Now that k is uniform and independent of s, key shift $\varDelta_j, \varDelta^*$ are determined by $\delta_j, \delta^*, \mathsf{s}, \mathsf{s}^*$, and the special rule is satisfied. It is ready for us to implement the robustness security reduction of rrFE to INT-RKA security of AIAE. According to the weak INT-RKA security of AIAE, the probability that $\mathsf{AIAE.Dec}(\mathsf{k} + \varDelta^*, \mathsf{ct}^*, \mathsf{aux}^* = \mathsf{s}^*) \neq \perp$ is negligible. Hence robustness of rrFE follows. The security reduction is non-trivial (see Sect.5 for details.).

Table 1. Comparison with known FE schemes. "Robustness?" asks whether the scheme achieves robustness. "Reusability?" asks whether the scheme achieves reusability. "Standard Assumption ?" asks whether the scheme is based on standard assumptions. "Linear Errors?" asks whether the scheme tolerates linear fraction of errors. "Free of Pairing?" asks whether the scheme is free of pairing. "–" represents the scheme is an information theoretical one.

FE Schemes	Robustness?	Reusability?	Standard Assumption?	Linear Errors?	Free of Pairing?
DRS04[11]	✗	✗	–	✔	✔
FMR13[12]	✗	✗	✔	✗	✔
BDKOS05[7]	✔	✗	✗	✔	✔
DKRS06[10], KR08[16], CDFPW08[9]	✔	✗	–	✔	✔
CFPRS16[8]	✗	✔	✗	✗	✔
Boyen04[6] ABCCFGS18[1]	✗	✔	✗	✔	✔
ACEK17[2]	✗	✔	✔	✗	✔
WL18[24],WLH18[26]	✗	✔	✔	✔	✔
Wen-Liu18[25]	✔	✔	✔	✔	✗
Ours rrFE from ui-ks PRF	✔	✔	✔	✗	✔
Ours rrFE from AIAE	✔	✔	✔	✔	✔

1.3 Comparison

The instantiation of our first approach results in a rrFE based on the LWE assumption, and the instantiation of our second approach results in a rrFE based on the DDH assumption over non-pairing groups. In Table 1, we compare our instantiations with the related works.

The rrFE from the LWE assumption supports sub-linear fraction of errors, due to the parameter choice of ui-ks PRF, but it serves as the first post-quantum rrFE.

The rrFE from the DDH assumption supports linear fraction of errors, just like the Wen-Liu18 rrFE in [25]. The advantages of this rrFE over [25] are as follows.

- Our rrFE is free of pairing, since the underlying building block AIAE is built over non-pairing groups. However, Wen-Liu18 rrFE heavily relies on pairings since its building block LAF is built over symmetric pairing groups [4].
- The crs and the public helper string P of our rrFE are much shorter than that of Wen-Liu18 rrFE [25]. Recall that in the Wen-Liu18 rrFE [25], the reading w is directly input to the building block LAF as an authentication key. This makes the length of the public key (a part of crs), the length of the tag (a part of P) and the evaluation complexity of LAF closely related to the length of w ($|w|$). Our rrFE avoids this problem since our approach has a different frame structure.
- Our rrFE is more efficient than Wen-Liu18 [25]. Due to the complicated pairing operations and the number of pairings depending on $|w|$, Wen-Liu18

[4] As noted by Galbraith [13], the symmetric pairings (i.e., Type 1 pairings) are now essentially dead and it would be better in future to design protocols that do not require Type 1 pairings.

rrFE suffers from high computational complexities in the generation and reproduction algorithms. In contrast, our rrFE is much efficient since the underlying building block AIAE is built over a simple group.

Precise comparison between our DDH-based rrFE and the Wen-Liu18 rrFE is shown in Table 2.

Table 2. Efficiency comparison of our instantiation of rrFE from AIAE and the Wen-Liu18 rrFE in [25]. "Exp/Gen" and "Pairing/Gen" represent the numbers of exponentiations and pairings over groups per generation respectively. "Exp/Rep" and "Pairing/Rep" represent the numbers of exponentiations and pairings over groups per reproduction respectively. The Wen-Liu18 rrFE [25] relies on the DDH assumption over a group $\mathbb{G}$ and the DLIN assumption over a group $\hat{\mathbb{G}}$ of order p' which admits symmetric pairing $e : \hat{\mathbb{G}} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. Define $\mathsf{des}(\hat{\mathbb{G}}, \mathbb{G}_T, e)$ the description of the symmetric pairing group. H_{pk} describes the chameleon hash. $|R_{ch}|$ is the bit-length of the randomness used in chameleon hash function. Meanwhile, $\mathsf{H_i}$, $\mathsf{H_{i_2}}$ describe universal hash functions, and $\mathsf{H_{i_1}}$ a collision-resistant hash function. Define $|\mathsf{des}(\hat{\mathbb{G}}, \mathbb{G}_T, e)|, |H_{pk}|, |\mathsf{H_i}|, |\mathsf{H_{i_1}}|$ and $|\mathsf{H_{i_2}}|$ the bit-lengths of the descriptions respectively. Define $\mathfrak{n} := |w|/\log p'$ (it is necessary that $\mathfrak{n} \geq 2$), where $|w|$ is the bit-length of the source reading w. Define $|a\mathbb{G}|$ as the bit-length of a elements in group $\mathbb{G}$. $|s|$ is the bit-length of secure sketch. $\bar{\mathbb{N}}$ is a prime of $4\lambda + 1$, and $\mathbb{QR}_{\bar{\mathbb{N}}}$ is a subgroup of quadratic residues of $\mathbb{Z}_{\bar{\mathbb{N}}}^*$.

rrFE Schemes	Bit-length of crs	Bit-length of P	Exp/Gen	Exp/Rep	Pairing/Gen	Pairing/Rep	Assumptions
Wen-Liu18 [25]	$\lvert\mathsf{des}(\hat{\mathbb{G}}, \mathbb{G}_T, e)\rvert + \lvert(\lambda + \mathfrak{n})\hat{\mathbb{G}}\rvert$ $+\lvert\mathbb{G}\rvert + \lvert H_{pk}\rvert + \lvert\mathsf{H_i}\rvert = O(\lambda^2)$	$\lvert s\rvert + \lambda + \lvert R_{ch}\rvert$ $+\lvert(1 + \mathfrak{n} + \mathfrak{n}^2)\hat{\mathbb{G}}\rvert$	$\mathfrak{n}^2$ (over $\hat{\mathbb{G}}$) $+2$ (over $\mathbb{G}$)	$\mathfrak{n}^2$ (over $\hat{\mathbb{G}}$) $+1$ (over $\mathbb{G}$)	$4\mathfrak{n}^2$	$4\mathfrak{n}^2$	DDH (over $\mathbb{G}$)+DLIN (over sym. pairing $\hat{\mathbb{G}}$)
Our rrFE from AIAE	$20\lambda + 3 + \lvert\mathsf{H_i}\rvert + \lvert\mathsf{H_{i_1}}\rvert$ $+\lvert\mathsf{H_{i_2}}\rvert = O(\lambda)$	$\lvert s\rvert + 10\lambda + 2$	4 (over $\mathbb{QR}_{\bar{\mathbb{N}}}$)	2 (over $\mathbb{QR}_{\bar{\mathbb{N}}}$)	0	0	DDH (over $\mathbb{QR}_{\bar{\mathbb{N}}}$)

2 Preliminaries

For an integer, denote $\{1, 2, \cdots, n\}$ by $[n]$. For a set $\mathcal{X}$, let $x \leftarrow_\$ \mathcal{X}$ denote randomly choosing an element x from set $\mathcal{X}$. For a random variable X, let $x \leftarrow X$ denote sampling x according to X. For two random variables X and Y, let $H_\infty(X)$ denote the min-entropy of X, the conditional min-entropy is defined by $H_\infty(X|Y) = -\log(\mathbb{E}_{y \leftarrow Y}\left[2^{-H_\infty(X|Y=y)}\right])$, and the statistical distance between X and Y is defined by $\mathsf{SD}(X, Y) = \frac{1}{2}\sum_{x \in \mathcal{X}} |\Pr[X = x] - \Pr[Y = x]|$. Let $X \overset{c}{\approx}_\epsilon Y$ denote that for any PPT adversary, its advantage to distinguish X and Y is no more than ϵ, and $X \approx_c Y$ denote that distributions X and Y are computationally indistinguishable.

A family of functions $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ takes a key $k \in \mathcal{K}$ and input $x \in \mathcal{X}$, and returns an output $F(k, x) \in \mathcal{Y}$. Let $\mathsf{FF}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$ be the set of all families of functions $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. For sets $\mathcal{X}, \mathcal{Y}$, let $\mathsf{Fun}(\mathcal{X}, \mathcal{Y})$ be the set of all functions mapping $\mathcal{X}$ to $\mathcal{Y}$.

For a real number x, let $\lceil x \rfloor$ denote rounding x to the closest integer, and $\lfloor x \rfloor$ denote rounding x to the largest integer which does not exceed it. For a string x, let $|x|$ denote the bit length of x. For integers q, p, y where $q \geq p \geq 2$, we define the function $\lfloor y \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lfloor y \rceil_p = i$, where $i \cdot \lfloor q/p \rfloor$ is the largest multiple of $\lfloor q/p \rfloor$ that does not larger than y. For a vector $\mathbf{y} \in \mathbb{Z}_q^m$, we define $\lfloor \mathbf{y} \rceil_p$ as the vector in $\mathbb{Z}_p^m$ obtained by rounding each coordinate of the vector individually.

For a primitive XX and a security notion YY, by $\mathsf{Exp}^{\mathrm{YY}}_{\mathcal{A},\mathrm{XX}}(\lambda) \Rightarrow 1$, we mean that the security experiment outputs 1 after interacting with an adversary $\mathcal{A}$. By $\mathsf{Adv}^{\mathrm{YY}}_{\mathcal{A},\mathrm{XX}}(\lambda)$, we denote the advantage of a PPT adversary $\mathcal{A}$ and define $\mathsf{Adv}^{\mathrm{YY}}_{\mathrm{XX}}(\lambda) := \max_{\mathrm{PPT}\,\mathcal{A}} \mathsf{Adv}^{\mathrm{YY}}_{\mathcal{A},\mathrm{XX}}(\lambda)$.

2.1 Universal Hash Functions

Definition 1 (Universal Hash Functions). *A family of hash functions $\mathcal{H}_{\mathcal{I}} = \{\mathsf{H_i} : \mathcal{X} \to \mathcal{Y}\}_{\mathsf{i} \in \mathcal{I}}$ is universal, if for all distinct $x, x' \in \mathcal{X}$, it holds that*

$$\Pr[\mathsf{H_i} : \mathsf{H_i}(x) = \mathsf{H_i}(x')] \leq 1/|\mathcal{Y}|,$$

where i is uniformly chosen from $\mathcal{I}$.

Lemma 1 (Generalized Leftover Hash Lemma). *Let $\mathcal{H}_{\mathcal{I}} = \{\mathsf{H_i} : \mathcal{X} \to \mathcal{Y}\}$ be a family of universal hash functions. Then for any two random variables X, Z,*

$$\mathsf{SD}((\mathsf{H}_I(X), I, Z), (U, I, Z)) \leq \frac{1}{2}\sqrt{|\mathcal{Y}| \cdot 2^{-\tilde{H}_\infty(X|Z)}}$$

holds, where I and U are uniform distributions over $\mathcal{I}$ and $\mathcal{Y}$, respectively.

Definition 2 (Homomorphic Universal Hash Functions). *Let $\mathcal{H}_{\mathcal{I}} = \{\mathsf{H_i} : \mathcal{X} \to \mathcal{Y}\}_{\mathsf{i} \in \mathcal{I}}$ be a family of universal hash functions. $\mathcal{H}_{\mathcal{I}}$ is homomorphic if for all $i \in \mathcal{I}$,*

$$\mathsf{H_i}(x + x') = \mathsf{H_i}(x) + \mathsf{H_i}(x').$$

In Appendix B, we present a concrete construction of homomorphic universal hash functions.

2.2 Secure Sketch

Definition 3 (Secure Sketch). *An $(\mathcal{M}, m, \widetilde{m}, t)$-secure sketch (SS) consists of a pair of PPT algorithms $(\mathsf{SS.Gen}, \mathsf{SS.Rec})$ with the following specifications:*

- $\mathsf{SS.Gen}(\mathsf{w})$ *on input* $\mathsf{w} \in \mathcal{M}$ *outputs a sketch* $\mathsf{s} \in \mathcal{S}$.
- $\mathsf{SS.Rec}(\mathsf{w}', \mathsf{s})$ *on input* $\mathsf{w}' \in \mathcal{M}$ *and a sketch* s *outputs* $\widetilde{\mathsf{w}}$.

It also satisfies the following properties:

Correctness. *If* $\mathsf{dis}(\mathsf{w}, \mathsf{w}') \leq t$, *then* $\mathsf{w} = \mathsf{SS.Rec}(\mathsf{w}', \mathsf{SS.Gen}(\mathsf{w}))$.
Privacy. *For any distribution W over $\mathcal{M}$, if $H_\infty(W) \geq m$, then $\widetilde{H}_\infty(W|\mathsf{SS.Gen}(W)) \geq \widetilde{m}$.*

Definition 4 (Secure Sketch Linearity Property). *[9] Let* $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ *be an* $(\mathcal{M}, m, \tilde{m}, t)$*-secure sketch. For any* $\mathsf{w} \in \mathcal{M}$*,* $\tilde{\mathsf{s}} \in \mathcal{S}$ *and* δ *such that* $\mathsf{dis}(\delta) \le t$*, let* $\mathsf{s} := \mathsf{SS.Gen}(\mathsf{w})$*,* $\widetilde{\mathsf{w}} := \mathsf{SS.Rec}(\mathsf{w} + \delta, \tilde{\mathsf{s}})$ *and* $\widetilde{\delta} := \widetilde{\mathsf{w}} - \mathsf{w}$*, then* SS *is linear if there exists a deterministic and efficiently computable function* g *such that* $\widetilde{\delta} = g(\delta, \mathsf{s}, \tilde{\mathsf{s}})$*.*

Definition 5 (Homomorphic Secure Sketch). *Let* $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ *be an* $(\mathcal{M}, m, \tilde{m}, t)$*-secure sketch.* SS *is homomorphic if*

$$\mathsf{SS.Gen}(\mathsf{w} + \mathsf{w}') = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\mathsf{w}').$$

In this paper, we will employ a homomorphic secure sketch with linearity property. An instantiation of such SS is the syndrome-based secure sketch [9], which is recalled in Appendix A.

2.3 Pseudorandom Functions

Informally, a pseudorandom function (PRF) is an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that no PPT adversary can distinguish the function from a truly random function given only black-box access. We review the definition of pseudorandom functions (PRF) [18] which considers the PRF with public parameters pp.

Definition 6 (PRF). *An efficiently computable function* $F_{\mathsf{pp}} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is a secure* PRF *if for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda) \Rightarrow 1] - 1/2|$ *is negligible in* λ*, where the game* $\mathsf{Exp}^{\mathsf{prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda)$ *is defined in Fig. 4. Here* $\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$*,* $\mathcal{K}$ *is the key space,* $\mathcal{X}$ *is the domain, and* $\mathcal{Y}$ *is the range of the function.*

```
Procedure INITIALIZE:
pp ← PRF.Setup($1^\lambda$), $b \leftarrow_{\$} \{0,1\}$.
If $b = 0$, $f(\cdot) \leftarrow_{\$} \mathsf{Fun}(\mathcal{X},\mathcal{Y})$.
Else, $\mathsf{k} \leftarrow_{\$} \mathcal{K}$, set $f(\cdot) := F_{\mathsf{pp}}(k,\cdot)$.
Return pp.
```

```
Procedure QUE($x$):
Return $f(x)$.

Procedure FINALIZE($b^*$)
If $b^* = b$, Return 1.
Else, Return 0.
```

Fig. 4. The experiment for defining the game $\mathsf{Exp}^{\mathsf{prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda)$ for PRF, where $\mathsf{Fun}(\mathcal{X},\mathcal{Y})$ is the set of all functions mapping $\mathcal{X}$ to $\mathcal{Y}$.

Definition 7 ($\varPhi$-RKA-PRF). PRF $F_{\mathsf{pp}} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is* $\varPhi$-RKA*-secure w.r.t. a class of related-key deriving functions* $\varPhi = \{\phi : \mathcal{K} \to \mathcal{K}\}$*, if for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda) \Rightarrow 1] - 1/2|$ *is negligible in* λ*, where the game* $\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda)$ *is defined in Fig. 5.*

```
Procedure INITIALIZE:
$\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$, $b \leftarrow_{\$} \{0,1\}$.
$\mathsf{k} \leftarrow_{\$} \mathcal{K}$.
If $b = 0$, $f(\cdot,\cdot) \leftarrow_{\$} \mathsf{FF}(\mathcal{K},\mathcal{X},\mathcal{Y})$
Else, set $f(\cdot,\cdot) := F_{\mathsf{pp}}(\cdot,\cdot)$,
Return $\mathsf{pp}$.
```

```
Procedure RKQUE($\phi \in \Phi, x$):
Return $f(\phi(\mathsf{k}), x)$.

Procedure FINALIZE($b^*$)
If $b^* = b$, Return 1.
Else, Return 0.
```

Fig. 5. The experiment for defining the Φ-RKA game $\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}$ for PRF, where $\mathsf{FF}(\mathcal{K},\mathcal{X},\mathcal{Y})$ is the set of all functions $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$.

Remark 1. We will make use the fact that $\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda) \Rightarrow 1] - 1/2| = \frac{1}{2}|\Pr[\mathcal{A} \Rightarrow 1 \mid f(\cdot,\cdot) = F_{\mathsf{pp}}(\mathsf{k},\cdot)] - \Pr[\mathcal{A} \Rightarrow 1 \mid f(\cdot,\cdot) \text{ is random}]|$, where $\mathcal{A} \Rightarrow 1$ means that the adversary $\mathcal{A}$ returns 1 to FINALIZE.

Definition 8 (Unique-Input RKA Security). *An adversary is a unique-input adversary if the input queries $(\phi_1, x_1), \cdots, (\phi_Q, x_Q)$ are such that $x_i \neq x_j$ in the game $\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F}(\lambda)$. A PRF F_{pp} is* unique-input Φ-RKA-*secure if it is* Φ-RKA-*secure against unique-input adversaries.*

Define the shift function family $\Phi_\Delta := \{\phi_a : \mathcal{K} \to \mathcal{K} \mid \phi_a(\mathsf{k}) = \mathsf{k} + a\}_{a \in \mathcal{K}}$. The unique-input Φ_Δ-RKA-security of PRF is also named *unique-input key-shift* (ui-ks) security. In this paper, ui-ks security of PRF is sufficient for our construction in Section 4.

2.4 Auxiliary-Input Authenticated Encryption

We recall the definition of auxiliary-input authenticated encryption scheme [14].

Definition 9 (AIAE). *An auxiliary-input authenticated encryption scheme consists of three PPT algorithms:*

- $\mathsf{AIAE.Setup}(1^\lambda)$ *on input the security parameter λ outputs the system parameter* pp*, which is an implicit input to* $\mathsf{AIAE.Enc}$ *and* $\mathsf{AIAE.Dec}$*. The system parameter* pp *implicitly defines the key space $\mathcal{K}$, the message space $\mathcal{M}_{\mathsf{AIAE}}$ and the auxiliary input space $\mathcal{AUX}$.*
- $\mathsf{AIAE.Enc}(\mathsf{k}, \mathsf{m}, \mathsf{aux})$ *on input a key $\mathsf{k} \in \mathcal{K}$, a message $\mathsf{m} \in \mathcal{M}_{\mathsf{AIAE}}$ and an auxiliary input $\mathsf{aux} \in \mathcal{AUX}$ outputs a ciphertext* ct.
- $\mathsf{AIAE.Dec}(\mathsf{k}, \mathsf{ct}, \mathsf{aux})$ *on input a key* k*, a ciphertext* ct *and an auxiliary input* aux *outputs a message* m *or a rejection symbol $\perp$.*

Correctness. For all $\mathsf{pp} \leftarrow \mathsf{AIAE.Setup}(1^\lambda)$, all $k \in \mathcal{K}$, all $\mathsf{m} \in \mathcal{M}_{\mathsf{AIAE}}$ and all $\mathsf{ct} \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}, \mathsf{m}, \mathsf{aux})$, it holds that $\mathsf{m} = \mathsf{AIAE.Dec}(\mathsf{k}, \mathsf{ct}, \mathsf{aux})$.

Definition 10 (IND-Φ-RKA and Weak INT-Φ-RKA Securities for AIAE). *For a class of related-key deriving functions $\Phi = \{\phi : \mathcal{K} \to \mathcal{K}\}$, an* AIAE

scheme is IND-Φ-RKA *and* weak INT-Φ-RKA *secure, if for any PPT adversary $\mathcal{A}$, both* $\mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}(\lambda) \Rightarrow 1] - 1/2|$ *and* $\mathsf{Adv}^{\mathsf{int\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{int\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}(\lambda) \Rightarrow 1]$ *are negligible, where games* $\mathsf{Exp}^{\mathsf{ind\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{int\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}(\lambda)$ *are depicted in Fig. 6.*

If AIAE is IND-Φ_Δ-RKA and Weak INT-Φ_Δ-RKA secure, then it is also called a *Key-Shift secure* AIAE.

```
Procedure INITIALIZE:
pp ← AIAE.Setup(1^λ), k ←$ K.
b ←$ {0, 1}.
Return pp.

Procedure LR(m_0, m_1, aux, φ ∈ Φ):
If |m_0| ≠ |m_1|, Return ⊥.
ct ← AIAE.Enc(φ(k), m_b, aux).
Return ct.

Procedure FINALIZE(b*)
If b = b*, Return 1.
Else, Return 0.
```

```
Procedure INITIALIZE:
pp ← AIAE.Setup(1^λ), k ←$ K.
Q_enc = Q_aux = ∅.
Return pp.

Procedure ENC(m, aux, φ ∈ Φ):
ct ← AIAE.Enc(φ(k), m, aux).
Q_enc := Q_enc ∪ {(aux, φ, ct)}.
Q_aux := Q_aux ∪ {(aux, φ)}.
Return ct.

Procedure FINALIZE(aux*, φ* ∈ Φ, ct*)
If (aux*, φ* ∈ Φ, ct*) ∈ Q_enc, Return 0.
If there exsits (aux, φ) ∈ Q_aux, such that
aux* = aux but φ* ≠ φ, Return 0.
Return (AIAE.Dec(φ*(k), ct*, aux*) ≠ ⊥).
```

Fig. 6. Left: The experiment for defining the IND-Φ-RKA game $\mathsf{Exp}^{\mathsf{int\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}$ for AIAE. Right: The experiment for defining the weak INT-Φ-RKA game $\mathsf{Exp}^{\mathsf{ind\text{-}rka}}_{\mathcal{A},\mathsf{AIAE}}$ for AIAE.

3 Robustly Reusable Fuzzy Extractor

Definition 11 (Fuzzy Extractor). *An* $(\mathcal{M}, m, \mathcal{R}, t, \varepsilon)$-fuzzy extractor (FE) *consists of three PPT algorithms* $\mathsf{FE} = (\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ *with the following properties:*

- $\mathsf{Init}(1^\lambda)$ *on input the security parameter* λ*, outputs the common reference string* crs.
- $\mathsf{Gen}(\mathsf{crs}, \mathsf{w})$ *on input the common reference string* crs *and an element* $w \in \mathcal{M}$*, outputs a public helper string* P *and an extracted string* $\mathsf{R} \in \mathcal{R}$.
- $\mathsf{Rep}(\mathsf{crs}, \mathsf{w}', \mathsf{P})$ *on input the common reference string* crs*, an element* $w' \in \mathcal{M}$ *and the public helper string* P*, outputs an extracted string* R *or* $\perp$.
- **Correctness.** *If* $\mathsf{dis}(\mathsf{w}, \mathsf{w}') \le t$*, then for all* $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$ *and* $(\mathsf{P}, \mathsf{R}) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{w})$*, we have* $\mathsf{R} = \mathsf{Rep}(\mathsf{crs}, \mathsf{w}', \mathsf{P})$.
- **Security.** *For any distribution* W *over* $\mathcal{M}$ *such that* $H_\infty(W) \ge m$*,* R *is pseudorandom even conditioned on* P *and* crs*, where* $(\mathsf{P}, \mathsf{R}) \leftarrow \mathsf{Gen}(\mathsf{crs}, W)$ *and* $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$.

Definition 12 (Robustly Reusable Fuzzy Extractor). *A* $\mathsf{FE} = (\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ *is called an* $(\mathcal{M}, m, \mathcal{R}, t, \varepsilon_1, \varepsilon_2)$-robustly reusable Fuzzy Extractor (rrFE)*, if for any PPT adversary* $\mathcal{A}$ *and any distribution* W *over* $\mathcal{M}$ *such that* $H_\infty(W) \geq m$*, it holds that* $\mathsf{Adv}^{\mathsf{reu}}_{\mathcal{A},\mathsf{FE}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{reu}}_{\mathcal{A},\mathsf{FE}}(\lambda) \Rightarrow 1] - 1/2| \leq \varepsilon_1$ *and* $\mathsf{Adv}^{\mathsf{rob}}_{\mathcal{A},\mathsf{FE}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{rob}}_{\mathcal{A},\mathsf{FE}}(\lambda) \Rightarrow 1] \leq \varepsilon_2$*, where games* $\mathsf{Exp}^{\mathsf{reu}}_{\mathcal{A},\mathsf{FE}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{rob}}_{\mathcal{A},\mathsf{FE}}(\lambda)$ *are specified in Fig. 7.*

```
Procedure INITIALIZE:
crs ← Init(1^λ).
b ←$ {0,1}.
w ← W.
Return crs.

Procedure CHALLENGE(δ):
If dis(δ) > t, Return ⊥.
(P, R) ← Gen(crs, w + δ).
If b = 1, Return (P, R).
Else, U ←$ R, Return (P, U).

Procedure FINALIZE(b*):
If b = b*, Return 1.
Else, Return 0.
```

```
Procedure INITIALIZE:
crs ← Init(1^λ).
w ← W.
Q = ∅.
Return crs.

Procedure GENERATION(δ):
If dis(δ) > t, Return ⊥.
(P, R) ← Gen(crs, w + δ).
Q = Q ∪ {P}.
Return (P, R).

Procedure FINALIZE(P*, δ*):
If dis(δ*) > t, Return 0.
If P* ∈ Q, Return 0.
Return (Rep(crs, w + δ*, P*) ≠ ⊥).
```

Fig. 7. Left: The experiment for defining the reusability game $\mathsf{Exp}^{\mathsf{reu}}_{\mathcal{A},\mathsf{FE}}(\lambda)$ for a FE. Right: The experiment for defining the robustness game $\mathsf{Exp}^{\mathsf{rob}}_{\mathcal{A},\mathsf{FE}}(\lambda)$ for a FE.

Remark 2. The definition of reusability is not identical to but implies the reusability defined in [25]. In [25], a fuzzy extractor is reusable if for all PPT adversary it is hard to distinguish $(\mathsf{U}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$ from $(\mathsf{R}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$, where $\mathsf{U}_1 \leftarrow_\$ \mathcal{R}$, $(\mathsf{P}_i, \mathsf{R}_i) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{w} + \delta_i)$ and δ_i is chosen by the adversary. In our definition, a fuzzy extractor is reusable if for all PPT adversary, it is hard to distinguish the tuple $(\mathsf{U}_1, \mathsf{U}_2, \cdots, \mathsf{U}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$ from $(\mathsf{R}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$. In fact, we can show if $(\mathsf{U}_1, \mathsf{U}_2, \cdots, \mathsf{U}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q) \overset{c}{\approx}_\epsilon (\mathsf{R}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$, then $(\mathsf{U}_1, \mathsf{U}_2, \cdots, \mathsf{U}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q) \overset{c}{\approx}_\epsilon (\mathsf{U}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$, by a hybrid argument we get that $(\mathsf{R}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q) \overset{c}{\approx}_{2\epsilon} (\mathsf{U}_1, \mathsf{R}_2, \cdots, \mathsf{R}_Q, \mathsf{P}_1, \cdots, \mathsf{P}_Q)$. This means that if a fuzzy extractor is ϵ-reusable in our definition, then it is 2ϵ-reusable in [25].

4 Construction of rrFE from Unique-Input RKA-PRF

We introduce a generic construction of robustly reusable fuzzy extractor (rrFE) from a unique-input key-shift (Φ_Δ-RKA) secure PRF, a Secure Sketch and a family of universal hash functions, as shown in Fig. 8.

$\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$:	$(\mathsf{P},\mathsf{R}) \leftarrow \mathsf{Gen}(\mathsf{crs},\mathsf{w})$:	$\mathsf{R} \leftarrow \mathsf{Rep}(\mathsf{crs},\mathsf{P},\mathsf{w}')$:
$\mathsf{i} \leftarrow_\$ \mathcal{I}$ (i.e., $\mathsf{H_i} \leftarrow_\$ \mathcal{H}_\mathcal{I}$). $\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$. $\mathsf{crs} = (\mathsf{H_i},\mathsf{pp})$. Return crs.	$\mathsf{s} \leftarrow \mathsf{SS.Gen}(\mathsf{w})$. $\mathsf{k} \leftarrow \mathsf{H_i}(\mathsf{w})$. $\mathsf{t} \leftarrow_\$ \mathcal{T}$. $(\mathsf{r},\mathsf{v}) \leftarrow F_{\mathsf{pp}}(\mathsf{k},(\mathsf{s},\mathsf{t}))$. $\mathsf{P} := (\mathsf{s},\mathsf{t},\mathsf{v})$, $\mathsf{R} := \mathsf{r}$.	Parse $\mathsf{P} = (\mathsf{s},\mathsf{t},\mathsf{v})$. $\tilde{\mathsf{w}} \leftarrow \mathsf{SS.Rec}(\mathsf{w}',\mathsf{s})$. $\tilde{\mathsf{k}} \leftarrow \mathsf{H_i}(\tilde{\mathsf{w}})$. $(\tilde{\mathsf{r}},\tilde{\mathsf{v}}) \leftarrow F_{\mathsf{pp}}(\tilde{\mathsf{k}},(\mathsf{s},\mathsf{t}))$. If $\tilde{\mathsf{v}} = \mathsf{v}$, Return $\mathsf{R} := \tilde{\mathsf{r}}$. Else, Return $\perp$.

Fig. 8. Construction of $\mathsf{rrFE_{PRF}}$ from unique-input key-shift secure PRF.

Theorem 1. *The fuzzy extractor* $\mathsf{rrFE_{PRF}}$ *in Fig. 8 is an* $(\mathcal{M}, m, \mathcal{R}, t, \varepsilon_1, \varepsilon_2)$*-robustly reusable fuzzy extractor with* $\varepsilon_1 = 2\mathsf{Adv}^{\mathsf{rka}}_{\mathsf{PRF}}(\lambda) + 2^{-\omega(\log\lambda)}$ *and* $\varepsilon_2 = 2\mathsf{Adv}^{\mathsf{rka}}_{\mathsf{PRF}}(\lambda) + 2^{-\omega(\log\lambda)}$, *if the underlying building blocks satisfies the following properties.*

- $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ *is a homomorphic* $(\mathcal{M}, m, \widetilde{m}, 2t)$*-secure sketch with linearity property.*
- $\mathcal{H}_\mathcal{I} = \{\mathsf{H_i} : \mathcal{M} \rightarrow \mathcal{K}\}_{i\in\mathcal{I}}$ *is a family of homomorphic universal hash functions such that* $\widetilde{m} - \log|\mathcal{K}| \geq \omega(\log\lambda)$.
- $F_{\mathsf{pp}}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$ *is a unique-input key-shift secure PRF such that* $\mathcal{X} = \mathcal{U} \times \mathcal{T}$, $\mathcal{S} \subseteq \mathcal{U}$, $\mathcal{Y} = \mathcal{R} \times \mathcal{V}$, $\log|\mathcal{T}| \geq \omega(\log\lambda)$ *and* $\log|\mathcal{V}| \geq \omega(\log\lambda)$.

The correctness of $\mathsf{rrFE_{PRF}}$ follows from the correctness of the underlying SS, since w can be correctly recovered from the public helper string P if $\mathsf{dis}(\mathsf{w},\mathsf{w}') \leq t$. The reusability and robustness are shown in Lemma 2 and Lemma 3 respectively.

Lemma 2. *The construction of* rrFE *in Fig. 8 is* ε_1*-reusable with*

$$\varepsilon_1 = 2\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathsf{PRF}}(\lambda) + 2^{-\omega(\log\lambda)}.$$

Proof. We will prove the reusability of rrFE via a series of games, as shown in Fig. 9. Game $\mathbf{G}_j$ denotes a variant of reusability game played between a PPT adversary $\mathcal{A}$ and a challenger who provides Procedures Initialize and Challenge for $\mathcal{A}$. Denote by $\Pr[\mathbf{G}_j]$ the probability that $\mathcal{A}$ wins, i.e., Finalize returns 1, in game $\mathbf{G}_j$. Obviously, $\mathcal{A}$ wins iff $b = b^*$.

Game $\mathbf{G}_0$. $\mathbf{G}_0$ is just the reusability game. More precisely, in Procedures Initialize, the challenger chooses $b \leftarrow_\$ \{0,1\}$, samples $\mathsf{w} \leftarrow W$, and generates $\mathsf{crs} = (\mathsf{H_i},\mathsf{pp})$. Upon receiving the j-th Challenge query δ_j from $\mathcal{A}$, the challenger answers $\mathcal{A}$'s Challenge query as follows:

1. If $\mathsf{dis}(\delta_j) > t$, then return $\perp$.
2. Compute sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$ and hash value $\mathsf{k}_j = \mathsf{H_i}(\mathsf{w} + \delta_j)$.
3. Choose $\mathsf{t}_j \leftarrow_\$ \mathcal{T}$, compute $(\mathsf{r}_j,\mathsf{v}_j) \leftarrow F_{\mathsf{pp}}(\mathsf{k}_j,(\mathsf{s}_j,\mathsf{t}_j))$ and set $\mathsf{P}_j := (\mathsf{s}_j,\mathsf{t}_j,\mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$.
4. If $b = 1$, return $(\mathsf{P}_j,\mathsf{R}_j)$, else choose $\mathsf{U}_j \leftarrow_\$ \mathcal{R}$ and return $(\mathsf{P}_j,\mathsf{U}_j)$.

```
Procedure INITIALIZE:    // Games $\mathbf{G}_0$, $\mathbf{G}_1$, $\boxed{\mathbf{G}_2, \mathbf{G}_3}$
$\mathsf{i} \leftarrow_{\$} \mathcal{I}$ (i.e., $\mathsf{H}_\mathsf{i} \leftarrow_{\$} \mathcal{H}_\mathcal{I}$).
$\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
$\mathsf{crs} = (\mathsf{H}_\mathsf{i}, \mathsf{pp})$.
$b \leftarrow_{\$} \{0,1\}$.
$\mathsf{w} \leftarrow W$.
$\boxed{\mathsf{k} \leftarrow_{\$} \mathcal{K}.}$
Return $\mathsf{crs}$.

Procedure FINALIZE($b^*$):    // Games $\mathbf{G}_0$-$\mathbf{G}_3$
If $b = b^*$, Return 1.
Else, Return 0.
```

```
Procedure CHALLENGE($\delta_j$):
    // Games $\mathbf{G}_0$, $\boxed{\mathbf{G}_1, \boxed{\mathbf{G}_2, \mathbf{G}_3}}$
If $\mathsf{dis}(\delta_j) > t$, Return $\perp$.
$\mathsf{s}_j \leftarrow \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$.
$\boxed{\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j).}$
$\mathsf{k}_j \leftarrow \mathsf{H}_\mathsf{i}(\mathsf{w} + \delta_j)$.
$\boxed{\mathsf{k}_j = \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\delta_j).}$
$\boxed{\mathsf{k}_j = \mathsf{k} + \mathsf{H}_\mathsf{i}(\delta_j).}$
$\mathsf{t}_j \leftarrow_{\$} \mathcal{T}$.
$(\mathsf{r}_j, \mathsf{v}_j) \leftarrow F_{\mathsf{pp}}(\mathsf{k}_j, (\mathsf{s}_j, \mathsf{t}_j))$.
$(\mathsf{r}_j, \mathsf{v}_j) \leftarrow_{\$} \mathcal{R} \times \mathcal{V}$.
$\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$.
If $b = 1$, Return $(\mathsf{P}_j, \mathsf{R}_j)$.
Else, $\mathsf{U}_j \leftarrow_{\$} \mathcal{R}$, Return $(\mathsf{P}_j, \mathsf{U}_j)$.
```

Fig. 9. Game $\mathbf{G}_0$-$\mathbf{G}_3$ for the security proof of Lemma 2.

Clearly,

$$\mathsf{Adv}^{\mathsf{reu}}_{\mathcal{A},\mathsf{FE}}(\lambda) = |\Pr[\mathbf{G}_0] - 1/2|. \tag{1}$$

Game $\mathbf{G}_1$: $\mathbf{G}_1$ is identical to $\mathbf{G}_0$, except for some conceptual changes of the generations of secure sketch s_j and hash value k_j. More precisely, step 2 is changed to step $2'$.

$2'$. Compute $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and $\mathsf{k}_j = \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\delta_j)$.

By the homomorphic properties of secure sketch and hash function, we have that

$$\Pr[\mathbf{G}_0] = \Pr[\mathbf{G}_1]. \tag{2}$$

Game $\mathbf{G}_2$. $\mathbf{G}_2$ is the same as $\mathbf{G}_1$, except for two changes.

The first change is to add $\mathsf{k} \leftarrow_{\$} \mathcal{K}$ in INITIALIZE of $\mathbf{G}_2$. The second change is the generation of k_j in CHALLENGE. In $\mathbf{G}_2$, instead of computing $\mathsf{k}_j := \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\delta_j)$, the challenger computes $\mathsf{k}_j = \mathsf{k} + \mathsf{H}_\mathsf{i}(\delta_j)$. More precisely, step $2'$ is changed to step $2''$.

$2''$ Compute $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and $\mathsf{k}_j = \mathsf{k} + \mathsf{H}_\mathsf{i}(\delta_j)$.

Claim 1 $|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \le 2^{-\omega(\log \lambda)}$.

Proof. Recall that $H_\infty(W) \ge m$. Then by the privacy of secure sketch, it follows that $H_\infty(W|\mathsf{SS.Gen}(W)) \ge \tilde{m}$. According to the Leftover Hash Lemma (see Lemma 1), we have

$$\mathsf{SD}((\mathsf{H}_\mathsf{i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}), (U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))) \le \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\tilde{m}}}, \tag{3}$$

where $U \leftarrow_{\$} \mathcal{K}$. This implies that for all powerful (not necessarily PPT) algorithm $\mathcal{B}$, it is impossible for $\mathcal{B}$ to tell $(\mathsf{H}_\mathsf{i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$ from $(U, \mathsf{i}, \mathsf{s} =$

$\mathsf{SS.Gen(w)}))$ with probability more than $\frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\widetilde{m}}}$. In formula,

$$|\Pr[\mathcal{B}(U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen(w)}) \Rightarrow 1] - \Pr[\mathcal{B}(\mathsf{H_i(w)}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen(w)}) \Rightarrow 1]| \leq \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\widetilde{m}}}. \tag{4}$$

Now we show that

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \leq \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\widetilde{m}}} \leq 2^{-\omega(\log \lambda)}. \tag{5}$$

We prove (5) by constructing a powerful algorithm $\mathcal{B}$ who aims to distinguish $(\mathsf{H_i(w)}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen(w)})$ from $(U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen(w)})$. Given $(X, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen(w)})$, where X is either $\mathsf{H_i(w)}$ or a uniform U, $\mathcal{B}$ simulates $\mathbf{G}_1/\mathbf{G}_2$ for $\mathcal{A}$ as follows.

- To simulate Procedure INITIALIZE, $\mathcal{B}$ randomly chooses a bit $b \leftarrow_{\$} \{0, 1\}$, then determines $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp})$ for $\mathcal{A}$ by determining $\mathsf{H_i}$ with i and invoking $\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
- To answer $\mathcal{A}$'s query δ_j, $\mathcal{B}$ simulates Procedure CHALLENGE(δ_j) as follows.
  - If $\mathsf{dis}(\delta_j) > t$, return $\perp$.
  - $\mathsf{s}_j = \mathsf{s} + \mathsf{SS.Gen}(\delta_j)$.
  - $\mathsf{k}_j = X + \mathsf{H_i}(\delta_j)$.
  - $\mathsf{t}_j \leftarrow_{\$} \mathcal{T}$.
  - $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow F_{\mathsf{pp}}(\mathsf{k}_j, (\mathsf{s}_j, \mathsf{t}_j))$.
  - $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$.
  - If $b = 1$, return $(\mathsf{P}_j, \mathsf{R}_j)$. Else, $\mathsf{U}_j \leftarrow_{\$} \mathcal{R}$, return $(\mathsf{P}_j, \mathsf{U}_j)$.
- Finally $\mathcal{A}$ outputs a guessing bit b^*. If $b = b^*$ (i.e., $\mathcal{A}$ wins), then $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs 0.

If $X = \mathsf{H_i(w)}$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_1$ for $\mathcal{A}$; if $X = U$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$. Consequently,

$$\begin{aligned}&|\Pr[\mathcal{B}(\mathsf{H_i(w)}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen(w)}) \Rightarrow 1] - \Pr[\mathcal{B}((U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen(w)}) \Rightarrow 1]| \\ = &|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]|. \end{aligned}\tag{6}$$

Obviously, Eq. (5) follows from Eq. (4), Eq. (6) and the fact of $\widetilde{m} - \log |\mathcal{K}| \geq \omega(\log \lambda)$. The claim follows. □

Game $\mathbf{G}_3$. $\mathbf{G}_3$ is the same as $\mathbf{G}_2$, except that $(\mathsf{r}_j, \mathsf{v}_j)$ is randomly chosen in $\mathbf{G}_3$. More precisely, step 3 is replaced with 3′ in Procedure CHALLENGE(δ_j) of $\mathbf{G}_3$.

3′ $\mathsf{t}_j \leftarrow_{\$} \mathcal{T}$, $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow_{\$} \mathcal{R} \times \mathcal{V}$ and set $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$.

Claim 2 $|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_3]| \leq 2\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathsf{PRF}}(\lambda) + 2^{-\omega(\log \lambda)}$.

Proof. Suppose that $\mathcal{A}$ makes Q challenge queries. Let Bad denote the event that there exist $i, j \in [Q]$ such that $\mathsf{t}_i = \mathsf{t}_j$. Note that t_j is randomly chosen from $\mathcal{T}$, so $\Pr[\mathsf{Bad}] = Q(Q-1)/(2|\mathcal{T}|)$. Let $\overline{\mathsf{Bad}}$ denote the event that Bad does not happen. Then

$$\Pr[\mathbf{G}_2] = \Pr[\mathbf{G}_2 \wedge \mathsf{Bad}] + \Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}}],$$

$$\Pr[\mathbf{G}_3] = \Pr[\mathbf{G}_3 \wedge \mathsf{Bad}] + \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}}],$$

$$\begin{aligned}
|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2]| &\le |\Pr[\mathbf{G}_3 \wedge \mathsf{Bad}] - \Pr[\mathbf{G}_3 \wedge \mathsf{Bad}]| + |\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}}] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}}]| \\
&\le \Pr[\mathsf{Bad}] + |\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}}] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}}]| \\
&= \frac{Q(Q-1)}{2|\mathcal{T}|} + |\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}}] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}}]|. \qquad (7)
\end{aligned}$$

Next we show that

$$|\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}}] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}}]| \le 2\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathsf{PRF}}(\lambda). \qquad (8)$$

by constructing a PPT algorithm $\mathcal{A}'$ against the unique-input key-shift security of PRF F_{pp}. Recall that in the unique-input key-shift security game $\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}$, $\mathcal{A}'$ obtains the public parameter pp which is generated via $\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ by Procedure INITIALIZE. Meanwhile, $\mathcal{A}'$ is able to query (ϕ_Δ, x) and Procedure RKQUE will reply $\mathcal{A}'$ with the function value of $f(\mathsf{k}+\Delta, x)$. The aim of $\mathcal{A}'$ is to tell whether $f(\mathsf{k},\cdot)$ is $F_{\mathsf{pp}}(\mathsf{k},\cdot)$ or a random function. Now $\mathcal{A}'$ simulates $\mathbf{G}_2/\mathbf{G}_3$ for $\mathcal{A}$ as follows.

– To simulate Procedure INITIALIZE of $\mathbf{G}_2/\mathbf{G}_3$, $\mathcal{A}'$ samples $\mathsf{w} \leftarrow W$, chooses $b \leftarrow_\$ \{0,1\}$ and an index $\mathsf{i} \leftarrow_\$ \mathcal{I}$ (hence $\mathsf{H_i}$), then sends $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp})$ to $\mathcal{A}$.
– $\mathcal{A}'$ initializes a set $\mathcal{Q}_t = \emptyset$. To answer $\mathcal{A}$'s query δ_j, $\mathcal{A}'$ simulates Procedure CHALLENGE(δ_j) as follows.
  - If $\mathsf{dis}(\delta_j) > t$, return $\perp$.
  - Compute $\mathsf{s}_j := \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and sample $\mathsf{t}_j \leftarrow_\$ \mathcal{T}$. If $\mathsf{t}_j \in \mathcal{Q}_t$, i.e., Bad happens, then $\mathcal{A}'$ aborts the game. Otherwise, $\mathcal{Q}_t := \mathcal{Q}_t \cup \{\mathsf{t}_j\}$.
  - $\mathcal{A}'$ queries $(\phi_{\mathsf{H_i}(\delta_j)}, (\mathsf{s}_j, \mathsf{t}_j))$ to its Procedure RKQUE, then RKQUE returns $(\mathsf{r}_j, \mathsf{v}_j)$ to $\mathcal{A}'$.
  - $\mathcal{A}'$ defines $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$ and $\mathsf{R}_j := \mathsf{r}_j$.
  - If $b = 1$, return $(\mathsf{P}_j, \mathsf{R}_j)$. Else, $\mathsf{U}_j \leftarrow_\$ \mathcal{R}$, return $(\mathsf{P}_j, \mathsf{U}_j)$.
– Finally $\mathcal{A}$ outputs a guessing bit b^* for FINALIZE. If $b = b^*$ (i.e., $\mathcal{A}$ wins), then $\mathcal{A}'$ outputs 1, otherwise $\mathcal{A}'$ outputs 0.

If Bad does not happen, then $\mathcal{A}'$ is a unique-input adversary. There are two cases.

- If $f(\mathsf{k},\cdot) = F_{\mathsf{pp}}(\mathsf{k},\cdot)$, then RKQUE computes $(\mathsf{r}_j, \mathsf{v}_j)$ via $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow F_{\mathsf{pp}}(\mathsf{k} + \mathsf{H_i}(\delta_j), (\mathsf{s}_j, \mathsf{t}_j))$. In this case, $\mathcal{A}'$ can perfectly simulate $\mathbf{G}_2 \wedge \overline{\mathsf{Bad}}$ for $\mathcal{A}$.
- If $f(\mathsf{k},\cdot)$ is a random function and RKQUE takes the value of the random function $f(\mathsf{k} + \mathsf{H_i}(\delta_j), (\mathsf{s}_j, \mathsf{t}_j))$ as $(\mathsf{r}_j, \mathsf{v}_j)$. In this case, $\mathcal{A}'$ can perfectly simulates $\mathbf{G}_3 \wedge \overline{\mathsf{Bad}}$ for $\mathcal{A}$.

Now we consider the advantage of $\mathcal{A}'$

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathcal{A}',\mathsf{PRF}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A}',F_{\mathsf{pp}}}(\lambda) \Rightarrow 1] - \frac{1}{2}| \\
&= \frac{1}{2}|\Pr[\mathcal{A}' \Rightarrow 1 \mid f(\cdot,\cdot) = F_{\mathsf{pp}}(\mathsf{k},\cdot)] - \Pr[\mathcal{A}' \Rightarrow 1 \mid f(\cdot,\cdot) \text{ is random}]| \\
&= \frac{1}{2}|\Pr\left[\mathcal{A} \text{ wins} \wedge \overline{\mathsf{Bad}} \mid f(\cdot,\cdot) = F_{\mathsf{pp}}(\mathsf{k},\cdot)\right] - \Pr\left[\mathcal{A} \text{ wins} \wedge \overline{\mathsf{Bad}} \mid f(\cdot,\cdot) \text{ is random}\right]| \\
&= \frac{1}{2}|\Pr\left[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}}\right] - \Pr\left[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}}\right]|. \qquad (9)
\end{aligned}$$

The claim follows from Eq. (7), Eq. (8) and the fact that $\log(|\mathcal{T}|) \geq \omega(\log \lambda)$. □

Observe that in $\mathbf{G}_3$, $(\mathsf{P}_j, \mathsf{R}_j)$ is generated in the same way, no matter whether $b = 0$ or $b = 1$. Therefore,

$$\Pr[\mathbf{G}_3] = 1/2. \tag{10}$$

Taking Eq. (1), Eq. (2), Claim 1, Claim 2 and Eq. (10) together, Lemma 2 follows. ▮

```
Procedure INITIALIZE:
  // Games $\mathbf{G}_0$, $\mathbf{G}_1$, $\boxed{\mathbf{G}_2, \mathbf{G}_3}$
  $\mathsf{i} \leftarrow_\$ \mathcal{I}$ (i.e., $\mathsf{H_i} \leftarrow_\$ \mathcal{H}_\mathcal{I}$).
  $\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
  $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp})$.
  $\mathsf{w} \leftarrow W$.
  $\mathcal{Q} := \emptyset$.
  $\boxed{\mathsf{k} \leftarrow_\$ \mathcal{K}.}$
  Return $\mathsf{crs}$.
```

```
Procedure GENERATION($\delta_j$):
  // Games $\mathbf{G}_0$, $\mathbf{G}_1$, $\boxed{\mathbf{G}_2,\ \mathbf{G}_3}$
  If $\mathsf{dis}(\delta_j) > t$, return $\perp$.
  $\mathsf{s}_j \leftarrow \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$.
  $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$.
  $\mathsf{k}_j \leftarrow \mathsf{H_i}(\mathsf{w} + \delta_j)$.
  $\mathsf{k}_j = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\delta_j)$.
  $\boxed{\mathsf{k}_j = \mathsf{k} + \mathsf{H_i}(\delta_j).}$
  $\mathsf{t}_j \leftarrow_\$ \mathcal{T}$.
  $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow F_{\mathsf{pp}}(\mathsf{k}_j, (\mathsf{s}_j, \mathsf{t}_j))$.
  $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow_\$ \mathcal{R} \times \mathcal{V}$.
  $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$.
  $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{P}_j\}$.
  Return $(\mathsf{P}_j, \mathsf{R}_j)$.
```

```
Procedure FINALIZE($\mathsf{P}^*, \delta^*$):
  // Games $\mathbf{G}_0$, $\mathbf{G}_1$, $\boxed{\mathbf{G}_2,\ \mathbf{G}_3}$
  If $\mathsf{dis}(\delta^*) > t$, Return 0.
  If $\mathsf{P}^* \in \mathcal{Q}$, Return 0.
  Parse $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{t}^*, \mathsf{v}^*)$.
  $\tilde{\mathsf{w}} \leftarrow \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$.
  $\widetilde{\delta^*} = g(\mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*)$.
  $\tilde{\mathsf{k}} \leftarrow \mathsf{H_i}(\tilde{\mathsf{w}})$.
  $\tilde{\mathsf{k}} = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\widetilde{\delta^*})$.
  $\boxed{\tilde{\mathsf{k}} = \mathsf{k} + \mathsf{H_i}(\widetilde{\delta^*}).}$
  $(\tilde{\mathsf{r}}, \tilde{\mathsf{v}}) \leftarrow F_{\mathsf{pp}}(\tilde{\mathsf{k}}, (\mathsf{s}^*, \mathsf{t}^*))$.
  $(\tilde{\mathsf{r}}, \tilde{\mathsf{v}}) \leftarrow_\$ \mathcal{R} \times \mathcal{V}$.
  If $\tilde{\mathsf{v}} = \mathsf{v}^*$, Return 1.
  Else, Return 0.
```

Fig. 10. Game $\mathbf{G}_0$-$\mathbf{G}_3$ for the security proof of Lemma 3.

Lemma 3. *The construction in Fig. 8 is ε_2-robust, with*

$$\varepsilon_2 = 2\mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{rka\text{-}prf}}(\lambda) + 2^{-\omega(\log \lambda)}. \tag{11}$$

Proof. We prove the robustness of fuzzy extractor by a sequence of games as shown in Fig. 10. Denote by $\Pr[\mathbf{G}_j]$ the probability that $\mathcal{A}$ wins in $\mathbf{G}_j$.
Game $\mathbf{G}_0$: $\mathbf{G}_0$ is the robustness game played between the challenger and a PPT adversary $\mathcal{A}$. More precisely, the challenger generates $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp}_{\mathsf{AIAE}})$, samples $\mathsf{w} \leftarrow W$, sets $\mathcal{Q} = \emptyset$, and returns crs to $\mathcal{A}$. Upon receiving the j-th generation query δ_j from $\mathcal{A}$, the challenger answers $\mathcal{A}$'s GENERATION query δ_j as follows:

1. If $\mathsf{dis}(\delta_j) > t$, then return $\perp$.
2. Compute the sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$ and the hash value $\mathsf{k}_j = \mathsf{H_i}(\mathsf{w} + \delta_j)$.
3. Sample $\mathsf{t}_j \leftarrow_\$ \mathcal{T}$, compute $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow F_{\mathsf{pp}}(\mathsf{k}_j, (\mathsf{s}_j, \mathsf{t}_j))$, set $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$, $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{P}_j\}$, and return $(\mathsf{P}_j, \mathsf{R}_j)$ to $\mathcal{A}$.

In FINALIZE, upon receiving (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{dis}(\delta^*) \geq t$ or $\mathsf{P}^* \in \mathcal{Q}$, the challenger returns 0. Else, it parses $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{t}^*, \mathsf{v}^*)$, then computes $\tilde{\mathsf{w}} = \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$, $\tilde{\mathsf{k}} = \mathsf{H_i}(\tilde{\mathsf{w}})$ and $(\tilde{\mathsf{r}}, \tilde{\mathsf{v}}) \leftarrow F_{\mathsf{pp}}(\tilde{\mathsf{k}}, (\mathsf{s}^*, \mathsf{t}^*))$. If $\tilde{\mathsf{v}} = \mathsf{v}^*$, it returns 1, else, it returns 0.

We have that

$$\mathsf{Adv}^{\mathsf{rob}}_{\mathcal{A},\mathsf{FE}}(\lambda) = \Pr[\mathbf{G}_0]. \tag{12}$$

Game $\mathbf{G}_1$: $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except for the following changes.

– When answering a generation query δ_j from $\mathcal{A}$, step 2 in $\textsc{Generation}(\delta_j)$ is changed into step $2'$:
  $2'$. Compute $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and $\mathsf{k}_j = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\delta_j)$.
– In $\textsc{finalize}$, the generation of $\tilde{\mathsf{k}}$ is changed. Instead of computing $\tilde{\mathsf{k}} := \mathsf{H_i}(\tilde{\mathsf{w}})$ with $\tilde{\mathsf{w}} := \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$, now $\tilde{\mathsf{k}} := \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\widetilde{\delta^*})$ with $\widetilde{\delta^*} = g(\mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*)$, where g is defined in Definition 4.

By the linearity property of the secure sketch and the homomorphic properties of secure sketch and hash function, the changes are just conceptual. Hence

$$\Pr[\mathbf{G}_0] = \Pr[\mathbf{G}_1]. \tag{13}$$

Game $\mathbf{G}_2$: $\mathbf{G}_2$ is the same as $\mathbf{G}_1$, except for the generation of k_j and $\tilde{k}$. Instead of computing $\mathsf{k}_j := \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\delta_j)$, now the challenger computes $\mathsf{k}_j := \mathsf{k} + \mathsf{H_i}(\delta_j)$ in $\textsc{Generation}(\delta_j)$ of $\mathbf{G}_2$. Instead of computing $\tilde{\mathsf{k}} = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\widetilde{\delta^*})$, now the challenger computes $\tilde{\mathsf{k}} = \mathsf{k} + \mathsf{H_i}(\widetilde{\delta^*})$ in $\textsc{Finalize}(\mathsf{P}^*, \delta^*)$ of $\mathbf{G}_2$. Here k is randomly chosen (once and for all in $\textsc{Initialize}$). More precisely,

– In $\textsc{Initialize}$, add $\mathsf{k} \leftarrow_{\$} \mathcal{K}$.
– When answering the generation queries from $\mathcal{A}$, step $2'$ in $\textsc{Generation}(\delta_j)$ is changed into step $2''$.
  $2''$. Compute $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and $\mathsf{k}_j = \mathsf{k} + \mathsf{H_i}(\delta_j)$.
– In $\textsc{Finalize}(\mathsf{P}^*, \delta^*)$, the challenger computes $\tilde{\mathsf{k}} = \mathsf{k} + \mathsf{H_i}(\widetilde{\delta^*})$ instead of $\tilde{\mathsf{k}} = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\widetilde{\delta^*})$.

Claim 3 $|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \leq 2^{-\omega(\log \lambda)}$.

Proof. The proof is similar to that of Claim 1 in the reusability proof, we have that

$$\mathsf{SD}((\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}), (\mathsf{U}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))) \leq \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\tilde{m}}}, \tag{14}$$

where $\mathsf{U} \leftarrow_{\$} \mathcal{K}$. In other words, for all powerful (not necessarily PPT) algorithm $\mathcal{B}$, it holds that

$$|\Pr[\mathcal{B}(\mathsf{U}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1] - \Pr[\mathcal{B}((\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1]| \leq \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\tilde{m}}}. \tag{15}$$

We construct a powerful algorithm $\mathcal{B}$ who aims to distinguish $(\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$ from $(U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$. Suppose that the challenge of $\mathcal{B}$ is $(X, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})$, where X is either $\mathsf{H_i}(\mathsf{w})$ or a uniform U. Then $\mathcal{B}$ simulates $\mathbf{G}_1/\mathbf{G}_2$ for $\mathcal{A}$ as follows.

- To simulate Procedure INITIALIZE, $\mathcal{B}$ randomly chooses a bit $b \leftarrow_{\$} \{0,1\}$, then determines $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp})$ for $\mathcal{A}$ by determining $\mathsf{H_i}$ with i and invoking $\mathsf{pp} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
- To answer $\mathcal{A}$'s query δ_j, $\mathcal{B}$ simulates Procedure CHALLENGE(δ_j) as follows.
  - If $\mathsf{dis}(\delta_j) > t$, return $\perp$.
  - $\mathsf{s}_j = \mathsf{s} + \mathsf{SS.Gen}(\delta_j)$.
  - $\mathsf{k}_j = X + \mathsf{H_i}(\delta_j)$.
  - $\mathsf{t}_j \leftarrow_{\$} \mathcal{T}$.
  - $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow F_{\mathsf{pp}}(\mathsf{k}_j, (\mathsf{s}_j, \mathsf{t}_j))$.
  - $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$.
  - Return $(\mathsf{P}_j, \mathsf{R}_j)$.
- Finally $\mathcal{A}$ sends (P^*, δ^*) to FINALIZE. If $\mathsf{dis}(\delta^*) > t$ or $\mathsf{P}^* \in \mathcal{Q}$, $\mathcal{B}$ returns 0 to its own challenger. Else, $\mathcal{B}$ parses $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{t}^*, \mathsf{v}^*)$, and computes $\tilde{\mathsf{k}} = X + \mathsf{H_i}(\widetilde{\delta^*})$ and $(\tilde{\mathsf{r}}, \tilde{\mathsf{v}}) \leftarrow F_{\mathsf{pp}}(\tilde{\mathsf{k}}, (\mathsf{s}^*, \mathsf{t}^*))$. If $\tilde{\mathsf{v}} = \mathsf{v}^*$, $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

If $X = \mathsf{H_i}(\mathsf{w})$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_1$ for $\mathcal{A}$; if $X = U$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$. Consequently,

$$\begin{aligned} &|\Pr[\mathcal{B}(\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1] - \Pr[\mathcal{B}((U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1]| \\ =& |\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]|. \end{aligned} \tag{16}$$

Therefore, Claim 3 follows from Eq. (15), Eq. (16) and the fact of $\widetilde{m} - \log|\mathcal{K}| \geq \omega(\log\lambda)$. □

Game $\mathbf{G}_3$: $\mathbf{G}_3$ is the same as $\mathbf{G}_2$, except that $(\mathsf{r}_j, \mathsf{v}_j)$ in CHALLENGE(δ_j) and $(\tilde{\mathsf{r}}, \tilde{\mathsf{v}})$ in FINALIZE are randomly chosen in $\mathbf{G}_3$. More precisely,

- In CHALLENGE(δ_j), step 3 is replaced with $3'$.
  $3'$. $\mathsf{t}_j \leftarrow_{\$} \mathcal{T}$ and $(\mathsf{r}_j, \mathsf{v}_j) \leftarrow_{\$} \mathcal{R} \times \mathcal{V}$, set $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$, $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{P}_j\}$, and return $(\mathsf{P}_j, \mathsf{R}_j)$ to $\mathcal{A}$.
- In FINALIZE, upon receiving a (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{dis}(\delta^*) \geq t$ or $\mathsf{P}^* \in \mathcal{Q}$, the challenger returns 0. Else, it parses $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{t}^*, \mathsf{v}^*)$, and samples $(\tilde{\mathsf{r}}, \tilde{\mathsf{v}}) \leftarrow_{\$} \mathcal{R} \times \mathcal{V}$. If $\tilde{\mathsf{v}} = \mathsf{v}^*$, it returns 1, else, it returns 0.

Observe that in $\mathbf{G}_3$, $\tilde{\mathsf{v}}$ is randomly chosen from $\mathcal{V}$, the probability of $\tilde{\mathsf{v}} = \mathsf{v}^*$ is bounded by $1/|\mathcal{V}|$. Note that $\log|\mathcal{V}| \geq \omega(\log\lambda)$, so we have that

$$\Pr[\mathbf{G}_3] \leq 2^{-\omega(\log\lambda)}. \tag{17}$$

Claim 4 $|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_3]| \leq 2\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathsf{PRF}}(\lambda) + 2^{-\omega(\log\lambda)}$.

Proof. The proof is similar to that of Claim 2.

Let Q denote the number of generation queries by $\mathcal{A}$. Let Bad denote the event that there exist $i, j \in [Q]$ such that $\mathsf{t}_i = \mathsf{t}_j$. Let Bad' denote the event that

$\exists j \in [Q]$ such that $(\mathsf{s}^*, \mathsf{t}^*) = (\mathsf{s}_j, \mathsf{t}_j)$. Similar to Eq. (7), we have

$$\begin{aligned}
&|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_3]| \\
\le\ &|\Pr[\mathbf{G}_2 \wedge (\mathsf{Bad} \vee \mathsf{Bad}')] - \Pr[\mathbf{G}_3 \wedge (\mathsf{Bad} \vee \mathsf{Bad}')]| \\
&+|\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}'] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}']| \\
=\ &|\Pr[\mathbf{G}_2 \wedge \mathsf{Bad}] - \Pr[\mathbf{G}_3 \wedge \mathsf{Bad}]| \\
&+|\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}'] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}']| &(18)\\
\le\ &\frac{Q(Q-1)}{2|\mathcal{T}|} + |\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}'}] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}'} \wedge \overline{\mathsf{Bad}}]|. &(19)
\end{aligned}$$

Eq.(18) is due to

$$\Pr[\mathbf{G}_2 \wedge \mathsf{Bad}'] = \Pr[\mathbf{G}_3 \wedge \mathsf{Bad}'] = 0, \tag{20}$$

and Eq. (19) is due to $|\Pr[\mathbf{G}_2 \wedge \mathsf{Bad}] - \Pr[\mathbf{G}_3 \wedge \mathsf{Bad}]| \le \Pr[\mathsf{Bad}] = \frac{Q(Q-1)}{2|\mathcal{T}|}$.

Eq.(20) means that it is impossible for $\mathcal{A}$ to win if Bad' happens, say $(\mathsf{s}^*, \mathsf{t}^*) = (\mathsf{s}_j, \mathsf{t}_j)$. The reason is as follows. Recall that $(\mathsf{s}^*, \mathsf{t}^*)$ is from $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{t}^*, \mathsf{v}^*)$ and $(\mathsf{s}_j, \mathsf{t}_j)$ is from $\mathsf{P}_j = (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$. Note that $\mathsf{dis}(\delta^*) \le t$, $\mathsf{dis}(\delta_j) \le t$ and $\mathsf{s}^* = \mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$, so we have that $\mathsf{w} + \delta_j = \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$ by the correctness of $(\mathcal{M}, m, \widetilde{m}, 2t)$-secure sketch. Meanwhile, by the linearity property we have $\mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*) = \mathsf{w} + \widetilde{\delta}^*$, where $\widetilde{\delta}^* = g(\delta^*, \mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*)$. As a result, $\delta_j = \widetilde{\delta}^*$ and $\mathsf{k}_j = \phi_{\mathsf{H_i}(\delta_j)}(\mathsf{k}) = \mathsf{k} + \mathsf{H_i}(\delta_j) = \mathsf{k} + \mathsf{H_i}(\widetilde{\delta}^*) = \phi_{\mathsf{H_i}(\widetilde{\delta}^*)}(\mathsf{k}) = \tilde{\mathsf{k}}$. Now that $(\tilde{\mathsf{k}}, (\mathsf{s}^*, \mathsf{t}^*)) = (\mathsf{k}_j, \mathsf{s}_j, \mathsf{t}_j)$, hence $F_{\mathsf{pp}}(\tilde{\mathsf{k}}, (\mathsf{s}^*, \mathsf{t}^*)) = F_{\mathsf{pp}}(\mathsf{k}_j, (\mathsf{s}_j, \mathsf{t}_j))$, i.e., $(\tilde{\mathsf{r}}, \tilde{\mathsf{v}}) = (\mathsf{r}_j, \mathsf{v}_j)$. If $\mathsf{v}^* = \mathsf{v}_j$, then $\mathsf{P}^* = \mathsf{P}_j$; otherwise $\tilde{\mathsf{v}} \ne \mathsf{v}^*$. Either case results in the failure of $\mathcal{A}$ in $\mathbf{G}_2/\mathbf{G}_3$.

Next we will prove

$$|\Pr[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}'}] - \Pr[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}'}]| \le 2\mathsf{Adv}^{\mathsf{rka\text{-}prf}}_{\mathsf{PRF}}(\lambda) \tag{21}$$

by constructing a PPT algorithm $\mathcal{A}'$ against the unique-input key-shift security of PRF F_{pp}, just like the proof of Eq. (8).

Recall that in the unique-input key-shift security game $\mathsf{Exp}^{\mathsf{rka\text{-}prf}}_{\mathcal{A},F_{\mathsf{pp}}}(\lambda)$, $\mathcal{A}'$ obtains the public parameter pp from its own INITIALIZE. Meanwhile, $\mathcal{A}'$ is able to query (ϕ_Δ, x) to RKQUE and obtain the value of $f(\mathsf{k} + \Delta, x)$. The aim of $\mathcal{A}'$ is to tell whether $f(\mathsf{k}, \cdot)$ is $F_{\mathsf{pp}}(\mathsf{k}, \cdot)$ or a random function. Now $\mathcal{A}'$ simulates $\mathbf{G}_2/\mathbf{G}_3$ for $\mathcal{A}$ as follows.

– To simulate INITIALIZE of $\mathbf{G}_2/\mathbf{G}_3$, $\mathcal{A}'$ samples $\mathsf{w} \leftarrow W$, chooses $b \leftarrow_\$ \{0,1\}$ and an index $\mathsf{i} \leftarrow_\$ \mathcal{I}$, then sends $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp})$ for $\mathcal{A}$. And $\mathcal{A}'$ sets $\mathcal{Q} = \mathcal{Q}_t = \emptyset$.
– To answer $\mathcal{A}$'s query δ_j, $\mathcal{A}'$ simulates Procedure CHALLENGE(δ_j) as follows.
  - If $\mathsf{dis}(\delta_j) > t$, return $\perp$.
  - $\mathsf{s}_j := \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and $\mathsf{t}_j \leftarrow_\$ \mathcal{T}$. If $\mathsf{t}_j \in \mathcal{Q}_t$, i.e., Bad happens, then $\mathcal{A}'$ aborts the game. Otherwise, $\mathcal{Q}_t := \mathcal{Q}_t \cup \{\mathsf{t}_j\}$.
  - $\mathcal{A}'$ queries $(\phi_{\mathsf{H_i}(\delta_j)}, (\mathsf{s}_j, \mathsf{t}_j))$ to its Procedure RKQUE, then RKQUE returns $(\mathsf{r}_j, \mathsf{v}_j)$ to $\mathcal{A}'$.

  - $\mathcal{A}'$ defines $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{t}_j, \mathsf{v}_j)$, $\mathsf{R}_j := \mathsf{r}_j$ and $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{P}_j\}$.
  - Return $(\mathsf{P}_j, \mathsf{R}_j)$.
- Finally, $\mathcal{A}$ sends (P^*, δ^*) to FINALIZE.
  - If $\mathsf{dis}(\delta^*) \geq t$ or $\mathsf{P}^* \in \mathcal{Q}$, $\mathcal{A}'$ returns 0 to its own challenger.
  - If $\exists j \in [Q]$ such that $(\mathsf{s}^*, \mathsf{t}^*) = (\mathsf{s}_j, \mathsf{t}_j)$, $\mathcal{A}'$ returns 0 to its own challenger.
  - $\mathcal{A}'$ parses $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{t}^*, \mathsf{v}^*)$ and computes $\widetilde{\delta^*} = g(\mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*)$. Then $\mathcal{A}'$ queries $((\phi_{\mathsf{H_i}(\widetilde{\delta^*})}, \mathsf{s}^*, \mathsf{t}^*)$ to RKQUE and receives $\tilde{y} = f(\mathsf{k} + \mathsf{H_i}(\widetilde{\delta^*}), (\mathsf{s}^*, \mathsf{t}^*))$ from RKQUE. $\mathcal{A}'$ parses $\tilde{y} = (\tilde{\mathsf{r}}, \tilde{\mathsf{v}})$. If $\tilde{\mathsf{v}} = \mathsf{v}^*$, $\mathcal{A}'$ returns 1, else $\mathcal{A}'$ returns 0 to its own challenger.

Suppose that neither Bad nor Bad' happens. Then

- $\mathcal{A}'$ perfectly simulates $\mathbf{G}_2 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}'}$ for $\mathcal{A}$ if $f(k, \cdot) = F_{\mathsf{pp}}(k, \cdot)$;
- $\mathcal{A}'$ perfectly simulates $\mathbf{G}_3 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}'}$ for $\mathcal{A}$ if $f(k, \cdot)$ is a random function.

Then the advantage of $\mathcal{A}'$ is given by

$$
\begin{aligned}
\mathsf{Adv}^{\text{rka-prf}}_{\mathcal{A}',\mathsf{PRF}}(\lambda) &= |\Pr[\mathsf{Exp}^{\text{rka-prf}}_{\mathcal{A}',F_{\mathsf{pp}}}(\lambda) \Rightarrow 1] - \frac{1}{2}| \\
&= \frac{1}{2}|\Pr\left[\mathcal{A}' \Rightarrow 1 \mid f(\cdot,\cdot) = F_{\mathsf{pp}}(k,\cdot)\right] - \Pr\left[\mathcal{A}' \Rightarrow 1 \mid f(\cdot,\cdot) \text{ is random}\right]| \\
&= \frac{1}{2}\Big|\Pr\left[\mathcal{A} \text{ wins} \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}' \mid f(\cdot,\cdot) = F_{\mathsf{pp}}(k,\cdot)\right] \qquad (22) \\
&\qquad - \Pr\left[\mathcal{A} \text{ wins} \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}' \mid f(\cdot,\cdot) \text{ is random}\right]\Big| \\
&= \frac{1}{2}|\Pr\left[\mathbf{G}_2 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}'\right] - \Pr\left[\mathbf{G}_3 \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Bad}}'\right]|. \qquad (23)
\end{aligned}
$$

This completes the proof of Eq. (21).
The claim follows from Eq. (19) Eq. (21) and the fact that $\log(|\mathcal{T}|) \geq \omega(\log \lambda)$. □

Taking Eq. (12) Eq. (13), Claim 3, Claim 4 and Eq. (17) together, Lemma 3 follows. ▮

5 Construction of rrFE from AIAE

In this section we propose a generic construction of robustly reusable fuzzy extractor $\mathsf{rrFE} = (\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ from a key-shift secure AIAE, a secure sketch and a family of universal hash functions as shown in Fig. 11.

```
$\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$:
$\mathsf{i} \leftarrow_\$ \mathcal{I}$ (i.e., $\mathsf{H_i} \leftarrow_\$ \mathcal{H}_{\mathcal{I}}$).
$\mathsf{pp}_{\mathsf{AIAE}} \leftarrow \mathsf{AIAE.Setup}(1^\lambda)$.
$\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp}_{\mathsf{AIAE}})$.
Return $\mathsf{crs}$.
```

```
$(\mathsf{P}, \mathsf{R}) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{w})$:
$\mathsf{s} \leftarrow \mathsf{SS.Gen}(\mathsf{w})$.
$\mathsf{k} \leftarrow \mathsf{H_i}(\mathsf{w})$.
$\mathsf{m} \leftarrow_\$ \mathcal{M}_{\mathsf{AIAE}}$.
$\mathsf{ct} \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}, \mathsf{m}, \mathsf{s})$.
$\mathsf{P} := (\mathsf{s}, \mathsf{ct}), \mathsf{R} := \mathsf{m}$.
```

```
$\mathsf{R} \leftarrow \mathsf{Rep}(\mathsf{crs}, \mathsf{P}, \mathsf{w}')$:
Parse $\mathsf{P} = (\mathsf{s}, \mathsf{ct})$.
$\tilde{\mathsf{w}} \leftarrow \mathsf{SS.Rec}(\mathsf{w}', \mathsf{s})$.
$\tilde{\mathsf{k}} \leftarrow \mathsf{H_i}(\tilde{\mathsf{w}})$.
$\tilde{\mathsf{m}}/\perp \leftarrow \mathsf{AIAE.Dec}(\tilde{\mathsf{k}}, \mathsf{ct}, \mathsf{s})$.
Return $\tilde{\mathsf{m}}/\perp$.
```

Fig. 11. Construction of $\mathsf{rrFE}_{\mathsf{AIAE}}$ from key-shift secure AIAE.

Theorem 2. *The fuzzy extractor* $\mathsf{rrFE}_{\mathsf{AIAE}}$ *in Fig. 11 is an* $(\mathcal{M}, m, \mathcal{M}_{\mathsf{AIAE}}, t, \varepsilon_1, \varepsilon_2)$*-robustly reusable fuzzy extractor where* $\varepsilon_1 = \mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathsf{AIAE}}(\lambda) + 2^{-\omega(\log \lambda)}$ *and* $\varepsilon_2 = \mathsf{Adv}^{\mathsf{int\text{-}rka}}_{\mathsf{AIAE}}(\lambda) + 2^{-\omega(\log \lambda)}$*, if the building blocks satisfy the following properties.*

- $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ *is a homomorphic* $(\mathcal{M}, m, \tilde{m}, 2t)$*-secure sketch with linearity property.*
- $\mathcal{H}_{\mathcal{I}} = \{\mathsf{H_i} : \mathcal{M} \to \mathcal{K}\}_{i \in \mathcal{I}}$ *is a family of homomorphic universal hash functions such that* $\tilde{m} - \log |\mathcal{K}| \geq \omega(\log \lambda)$*.*
- AIAE *is key-shift secure (*IND-Φ_Δ-RKA *and* weak INT-Φ_Δ-RKA *secure) with key space* $\mathcal{K}$*, message space* $\mathcal{M}_{\mathsf{AIAE}}$ *and auxiliary input space* $\{0,1\}^*$*.*

The correctness follows from the correctness of the underlying SS and AIAE. More precisely, if $\mathsf{dis}(\mathsf{w}, \mathsf{w}') \leq t$, then by the correctness of secure sketch, w can be correctly recovered, so is the secret key $\mathsf{k}(= \mathsf{H_i}(\mathsf{w}))$. Then by the correctness of AIAE, the message m, i.e., R can be precisely reproduced. The reusability and robustness of $\mathsf{rrFE}_{\mathsf{AIAE}}$ are shown in Lemma 4 and Lemma 5 respectively.

```
Procedure INITIALIZE:   // Games $G_0$, $G_1$, $G_2$
$\mathsf{i} \leftarrow_\$ \mathcal{I}$ (i.e., $\mathsf{H_i} \leftarrow_\$ \mathcal{H}_\mathcal{I}$).
$\mathsf{pp}_{\mathsf{AIAE}} \leftarrow \mathsf{AIAE.Setup}(1^\lambda)$.
$\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp}_{\mathsf{AIAE}})$.
$b \leftarrow_\$ \{0,1\}$.
$\mathsf{w} \leftarrow W$.
$\mathsf{k} \leftarrow_\$ \mathcal{K}$.
Return $\mathsf{crs}$.

Procedure FINALIZE($b^*$):   // Games $G_0$-$G_2$
If $b = b^*$, Return 1.
Else, Return 0.
```

```
Procedure CHALLENGE($\delta_j$):
                              // Games $G_0$, $\boxed{G_1,\ G_2}$
If $\mathsf{dis}(\delta_j) > t$, Return $\perp$.
$\mathsf{s}_j \leftarrow \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$.
$\boxed{\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j).}$
$\mathsf{k}_j \leftarrow \mathsf{H_i}(\mathsf{w} + \delta_j)$.
$\boxed{\mathsf{k}_j = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\delta_j).}$
$\mathsf{k}_j = \mathsf{k} + \mathsf{H_i}(\delta_j)$.
$\mathsf{m}_j \leftarrow_\$ \mathcal{M}_{\mathsf{AIAE}}$.
$\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}_j, \mathsf{m}_j, \mathsf{s}_j)$.
$\mathsf{P}_j := (\mathsf{s}_j, \mathsf{ct}_j)$, $\mathsf{R} := \mathsf{m}_j$.
If $b = 1$, Return $(\mathsf{P}_j, \mathsf{R}_j)$.
Else, $\mathsf{U}_j \leftarrow_\$ \mathcal{M}_{\mathsf{AIAE}}$, Return $(\mathsf{P}_j, \mathsf{U}_j)$.
```

Fig. 12. Game $\mathbf{G}_0$, $\mathbf{G}_1$ and $\mathbf{G}_2$ for the security proof of Lemma 4.

Lemma 4. *The fuzzy extractor* $\mathsf{rrFE}_{\mathsf{AIAE}}$ *in Fig. 11 is* ε_1*-reusable with* $\varepsilon_1 = \mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathsf{AIAE}}(\lambda) + 2^{-\omega(\log \lambda)}$*.*

Proof. We will prove the reusability of our construction via a series of games as shown in Fig. 12. By $\Pr[\mathbf{G}_j]$ we denote the probability that $\mathcal{A}$ wins in game $\mathbf{G}_j$.

Game $\mathbf{G}_0$: $\mathbf{G}_0$ is the reusability game played between the challenger and a PPT adversary $\mathcal{A}$. More precisely, in Procedures INITIALIZE, the challenger chooses $b \leftarrow_\$ \{0,1\}$, samples $\mathsf{w} \leftarrow W$, generates $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp}_{\mathsf{AIAE}})$, and returns crs to $\mathcal{A}$. Upon receiving the j-th CHALLENGE query δ_j from $\mathcal{A}$, the challenger answers $\mathcal{A}$'s query as follows:

1. If $\mathsf{dis}(\delta_j) > t$, then return $\perp$.

2. Compute the sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}+\delta_j)$ and the hash value $\mathsf{k}_j = \mathsf{H_i}(\mathsf{w}+\delta_j)$.
3. Randomly choose a message $\mathsf{m}_j \leftarrow_{\$} \mathcal{M}_{\mathsf{AIAE}}$, compute $\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}_j, \mathsf{m}_j, \mathsf{s}_j)$, set $\mathsf{P}_j = (\mathsf{s}_j, \mathsf{ct}_j)$ and $\mathsf{R}_j = \mathsf{m}_j$.
4. If $b = 1$, return $(\mathsf{P}_j, \mathsf{R}_j)$, else randomly choose $\mathsf{U}_j \leftarrow_{\$} \mathcal{M}_{\mathsf{AIAE}}$ and return $(\mathsf{P}_j, \mathsf{U}_j)$.

We have that

$$\mathsf{Adv}^{\mathsf{reu}}_{\mathcal{A},\mathsf{FE}}(\lambda) = |\Pr[\mathbf{G}_0] - 1/2|. \tag{24}$$

Game $\mathbf{G}_1$: Game $\mathbf{G}_1$ is identical to $\mathbf{G}_0$, except the conceptual changes of the generations of the secure sketch and the hash value. More precisely, step 2 is changed to step 2′ in CHALLENGE(δ_j).

2′. compute the sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and the hash value $\mathsf{k}_j = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\delta_j)$.

By the homomorphic properties of secure sketch and hash function, we have

$$\Pr[\mathbf{G}_0] = \Pr[\mathbf{G}_1]. \tag{25}$$

Game $\mathbf{G}_2$: Game $\mathbf{G}_2$ is identical to $\mathbf{G}_1$, except that instead of computing $\mathsf{k}_j = \mathsf{H_i}(\mathsf{w}) + \mathsf{H_i}(\delta_j)$, the challenger randomly chooses an element k from $\mathcal{K}$ in INITIALIZE and computes $\mathsf{k}_j := \mathsf{k} + \mathsf{H_i}(\delta_j)$ in CHALLENGE(δ_j) of $\mathbf{G}_2$. More precisely,

– In INITIALIZE, add $\mathsf{k} \leftarrow_{\$} \mathcal{K}$.
– When answering the generation queries from $\mathcal{A}$, step 2′ in CHALLENGE(δ_j) is changed into step 2″.
  2″. Compute $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and $\mathsf{k}_j = \mathsf{k} + \mathsf{H_i}(\delta_j)$.

Claim 5 $|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \le 2^{-\omega(\log\lambda)}$.

Proof. This proof is essentially the same as the proof of Claim 1. We omit it here (See Appendix E.1 for details). □

Claim 6 $|\Pr[\mathbf{G}_2] - 1/2| \le \mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathsf{AIAE}}(\lambda)$.

Proof. We will reduce the IND-Φ_Δ-RKA security of AIAE to the altered reusability game as described in Game $\mathbf{G}_2$. To this end, we assume a PPT adversary $\mathcal{A}$ winning $\mathbf{G}_2$ and show how to construct a PPT IND-Φ_Δ-RKA adversary $\mathcal{B}$. On input $\mathsf{pp}_{\mathsf{AIAE}}$, $\mathcal{B}$ samples $\mathsf{w} \leftarrow W$ and $\mathsf{i} \leftarrow_{\$} \mathcal{I}$ (i.e., $\mathsf{H_i} \leftarrow_{\$} \mathcal{H}_{\mathcal{I}}$), sets $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp}_{\mathsf{AIAE}})$ and returns crs to $\mathcal{A}$. Upon receiving the i-th challenge query δ_j from $\mathcal{A}$, adversary $\mathcal{B}$ simulates CHALLENGE(δ_j) for $\mathcal{A}$ as follows:

1. If $\mathsf{dis}(\delta_j) > t$, then return $\perp$.
2. Compute the sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and the hash value $\Delta_j = \mathsf{H_i}(\delta_j)$.
3. Randomly choose two messages $(\mathsf{m}_{j0}, \mathsf{m}_{j1}) \leftarrow_{\$} \mathcal{M}_{\mathsf{AIAE}}$, and send $(\mathsf{m}_{j0}, \mathsf{m}_{j1}, \mathsf{aux}_j = \mathsf{s}_j, \phi_{\Delta_j})$ to its own challenger.

4. Upon receiving ct_j from its own challenger, set $\mathsf{P}_j = (\mathsf{s}_j, \mathsf{ct}_j)$, and return $(\mathsf{P}_j, \mathsf{m}_{j1})$.

Finally, $\mathcal{A}$ outputs a guessing bit b^*, then $\mathcal{B}$ forwards b^* to its own challenger. It is straightforward to see that $\mathcal{B}$ simulates game $\mathbf{G}_2$ perfectly. More precisely,

– If $\mathsf{ct}_j = \mathsf{AIAE.Enc}(\phi_{\Delta_j}, m_{j0}, \mathsf{aux})$, then $\mathcal{B}$ perfectly simulates for the case $b = 0$ in $\mathbf{G}_2$.
– If $\mathsf{ct}_j = \mathsf{AIAE.Enc}(\phi_{\Delta_j}, m_{j1}, \mathsf{aux})$, then $\mathcal{B}$ perfectly simulates for the case $b = 1$ in $\mathbf{G}_2$.

Clearly, $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins. This yields $|\Pr[\mathbf{G}_2] - 1/2| = \mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathcal{B},\mathsf{AIAE}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathsf{AIAE}}(\lambda)$. □

Taking Eq. (24), Eq. (25), Claim 5 and Claim 6 together, we have $\mathsf{Adv}^{\mathsf{reu}}_{\mathsf{FE}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathsf{AIAE}}(\lambda) + 2^{-\omega(\log\lambda)}$, and Lemma 4 follows. ▮

Procedure INITIALIZE:	**Procedure** GENERATION(δ_j):	**Procedure** FINALIZE(P^*, δ^*):
// Games $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$	// Games $\mathbf{G}_0$, $\boxed{\mathbf{G}_1,\ \mathbf{G}_2}$	// Games $\mathbf{G}_0$, $\boxed{\mathbf{G}_1,\ \mathbf{G}_2}$
$\mathsf{i} \leftarrow_\$ \mathcal{I}$ (i.e., $\mathsf{H}_\mathsf{i} \leftarrow_\$ \mathcal{H}_\mathcal{I}$).	If $\mathsf{dis}(\delta_j) > t$, Return $\perp$.	If $\mathsf{dis}(\delta^*) > t$, Return 0.
$\mathsf{pp}_{\mathsf{AIAE}} \leftarrow \mathsf{AIAE.Setup}(1^\lambda)$.	$\mathsf{s}_j \leftarrow \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$.	If $\mathsf{P}^* \in \mathcal{Q}$, Return 0.
$\mathsf{crs} = (\mathsf{H}_\mathsf{i}, \mathsf{pp}_{\mathsf{AIAE}})$.	$\boxed{\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j).}$	Parse $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{ct}^*)$.
$\mathsf{w} \leftarrow W$.	$\mathsf{k}_j \leftarrow \mathsf{H}_\mathsf{i}(\mathsf{w} + \delta_j)$.	$\tilde{\mathsf{w}} \leftarrow \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$.
$\mathcal{Q} := \emptyset$.	$\boxed{\mathsf{k}_j = \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\delta_j).}$	$\boxed{\widetilde{\delta^*} = g(\mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*).}$
$\mathsf{k} \leftarrow_\$ \mathcal{K}$.	$\mathsf{k}_j = \mathsf{k} + \mathsf{H}_\mathsf{i}(\delta_j)$.	$\tilde{\mathsf{k}} \leftarrow \mathsf{H}_\mathsf{i}(\tilde{\mathsf{w}})$.
Return crs.	$\mathsf{m}_j \leftarrow_\$ \mathcal{M}_{\mathsf{AIAE}}$.	$\boxed{\tilde{\mathsf{k}} = \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\widetilde{\delta^*}).}$
	$\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}_j, \mathsf{m}_j, \mathsf{s}_j)$.	$\tilde{\mathsf{k}} = \mathsf{k} + \mathsf{H}_\mathsf{i}(\widetilde{\delta^*})$.
	$\mathsf{P}_j := (\mathsf{s}_j, \mathsf{ct}_j), \mathsf{R} := \mathsf{m}_j$.	Return $(\mathsf{AIAE.Dec}(\tilde{\mathsf{k}}, \mathsf{ct}^*, \mathsf{s}^*) \neq \perp)$.
	$\mathcal{Q} = \mathcal{Q} \cup \{\mathsf{P}_j\}$.	
	Return $(\mathsf{P}_j, \mathsf{R}_j)$.	

Fig. 13. Game $\mathbf{G}_0$-$\mathbf{G}_2$ for the security proof of Lemma 5.

Lemma 5. *The fuzzy extractor* $\mathsf{rrFE}_{\mathsf{AIAE}}$ *in Fig. 11 is* ε_2*-robust with* $\varepsilon_2 = \mathsf{Adv}^{\mathsf{ind\text{-}rka}}_{\mathsf{AIAE}}(\lambda) + 2^{-\omega(\log\lambda)}$.

Proof. We prove the robustness of reusable fuzzy extractor by a sequence of games as shown in Fig. 13. By $\Pr[\mathbf{G}_j]$ we denote the probability that $\mathcal{A}$ wins in game $\mathbf{G}_j$.

Game $\mathbf{G}_0$: $\mathbf{G}_0$ is the original robustness game. More precisely, let $\mathsf{crs} = (\mathsf{H}_\mathsf{i}, \mathsf{pp}_{\mathsf{AIAE}})$, $\mathcal{Q} = \emptyset$ and $\mathsf{w} \leftarrow W$. Upon receiving the j-th GENERATION query δ_j from $\mathcal{A}$, the challenger answers $\mathcal{A}$'s query as follows:

1. If $\mathsf{dis}(\delta_j) > t$, then return $\perp$.
2. Compute the sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$ and the hash value $\mathsf{k}_j = \mathsf{H}_\mathsf{i}(\mathsf{w} + \delta_j)$.

3. Randomly choose a message $\mathsf{m}_j \leftarrow_\$ \mathcal{M}_{\mathsf{AIAE}}$, compute $\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}_j, \mathsf{m}_j, \mathsf{s}_j)$, set $\mathsf{P}_j = (\mathsf{s}_j, \mathsf{ct}_j)$, $\mathsf{R}_j = \mathsf{m}_j$ and $\mathcal{Q} = \mathcal{Q} \cup \{\mathsf{P}_j\}$, and return $(\mathsf{P}_j, \mathsf{R}_j)$ to $\mathcal{A}$.

In FINALIZE, upon receiving a (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{dis}(\delta^*) \geq t$ or $\mathsf{P}^* \in \mathcal{Q}$, the challenger returns 0. Else, it parses $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{ct}^*)$, then computes $\tilde{\mathsf{w}} = \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$ and $\tilde{\mathsf{k}} = \mathsf{H}_\mathsf{i}(\tilde{\mathsf{w}})$. If $\mathsf{AIAE.Dec}(\tilde{\mathsf{k}}, \mathsf{ct}^*, \mathsf{s}^*) = \perp$, then return 0, otherwise return 1. We have that

$$\mathsf{Adv}^{\mathsf{rob}}_{\mathcal{A},\mathsf{FE}}(\lambda) = \Pr[\mathbf{G}_0]. \tag{26}$$

Game $\mathbf{G}_1$: $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except for the following changes.

– When answering a generation query δ_j from $\mathcal{A}$, step 2 in GENERATION(δ_j) is changed into step $2'$:
  $2'$. Compute $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and $\mathsf{k}_j = \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\delta_j)$.
– In FINALIZE, the generation of $\tilde{\mathsf{k}}$ is changed. Instead of computing $\tilde{\mathsf{k}} := \mathsf{H}_\mathsf{i}(\tilde{\mathsf{w}})$ with $\tilde{\mathsf{w}} := \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$, now $\tilde{\mathsf{k}} := \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\widetilde{\delta^*})$ with $\widetilde{\delta^*} = g(\mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*)$, where g is defined in Definition 4.

By the linearity property of the secure sketch and the homomorphic properties of secure sketch and hash function, the changes are just conceptual. Hence

$$\Pr[\mathbf{G}_0] = \Pr[\mathbf{G}_1]. \tag{27}$$

Game $\mathbf{G}_2$: Game $\mathbf{G}_2$ is identical to $\mathbf{G}_1$, except that the challenger replaces $\mathsf{H}_\mathsf{i}(\mathsf{w})$ by a randomly choosen k from $\mathcal{K}$. More precisely,

– In INITIALIZE, challenger will additionally sample $\mathsf{k} \leftarrow_\$ \mathcal{K}$.
– When the challenger answers the generation queries, step $2'$ is changed into step $2''$:
  $2''$. compute the sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and the hash value $\mathsf{k}_j = \mathsf{k} + \mathsf{H}_\mathsf{i}(\delta_j)$.
– In FINALIZE, the challenger computes $\tilde{\mathsf{k}} = \mathsf{k} + \mathsf{H}_\mathsf{i}(\widetilde{\delta^*})$ instead of $\tilde{\mathsf{k}} = \mathsf{H}_\mathsf{i}(\mathsf{w}) + \mathsf{H}_\mathsf{i}(\widetilde{\delta^*})$.

Claim 7 $|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]| \leq 2^{-\omega(\log \lambda)}$.

Proof. This proof is similar to that of Claim 3. We omit it here (See Appendix E.2 for details).

Claim 8 $\Pr[\mathbf{G}_2] \leq \mathsf{Adv}^{\mathsf{int\text{-}rka}}_{\mathsf{AIAE}}(\lambda)$.

Proof. We will reduce the INT-Φ_Δ-RKA security of AIAE to the altered robustness game as described in Game $\mathbf{G}_2$. To this end, we assume a PPT adversary $\mathcal{A}$ winning $\mathbf{G}_2$ and show how to construct a PPT weak INT-Φ_Δ-RKA adversary $\mathcal{B}$. On input $\mathsf{pp}_{\mathsf{AIAE}}$, adversary $\mathcal{B}$ samples $\mathsf{w} \leftarrow W$ and $\mathsf{i} \leftarrow_\$ \mathcal{I}$ (i.e., $\mathsf{H}_\mathsf{i} \leftarrow_\$ \mathcal{H}_\mathcal{I}$), sets $\mathcal{Q} = \emptyset$ and $\mathsf{crs} = (\mathsf{H}_\mathsf{i}, \mathsf{pp}_{\mathsf{AIAE}})$, and returns crs to $\mathcal{A}$. Upon receiving the j-th GENERATION query δ_j from $\mathcal{A}$, adversary $\mathcal{B}$ answers $\mathcal{A}$'s query as follows:

1. If $\mathsf{dis}(\delta_j) > t$, then return $\perp$.
2. Compute the sketch $\mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta_j)$ and the hash value $\Delta_j = \mathsf{H_i}(\delta_j)$.
3. Randomly choose a messages $\mathsf{m}_j \leftarrow_\$ \mathcal{M}_{\mathsf{AIAE}}$, and send $(\mathsf{m}_j, \mathsf{aux}_j = \mathsf{s}_j, \phi_{\Delta_j})$ to its own challenger.
4. Upon receiving ct_j from its own challenger, set $\mathsf{P}_j = (\mathsf{s}_j, \mathsf{ct}_j)$, $\mathcal{Q} = \mathcal{Q} \cup \{\mathsf{P}_j\}$ and return $(\mathsf{P}_j, \mathsf{m}_j)$.

Finally $\mathcal{A}$ will output (P^*, δ^*) for FINALIZE. If $\mathsf{dis}(\delta^*) \geq t$ or $\mathsf{P}^* \in \mathcal{Q}$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ parses $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{ct}^*)$, then computes $\widetilde{\delta}^* = g(\mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*)$ and $\Delta^* = \mathsf{H_i}(\widetilde{\delta}^*)$. Finally, $\mathcal{B}$ takes $(\mathsf{aux}^* = \mathsf{s}^*, \phi_{\Delta^*}, \mathsf{ct}^*)$ as its own forgery and sends the forgery to its own challenger.

Note that $\mathcal{B}$ simulates game $\mathbf{G}_2$ perfectly. As long as the forgery satisfies the additional special rule required for the weak INT-Φ_Δ-RKA security, $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins.

We show that the forgery always satisfies the special rule, i.e., if $\mathsf{aux}^* = \mathsf{s}^* = \mathsf{s}_j = \mathsf{aux}_j$ for some $j \in [Q]$, then $\phi_{\Delta_j} = \phi_{\Delta^*}$.

Note that $\mathsf{dis}(\delta^*) \leq t$, $\mathsf{dis}(\delta_j) \leq t$ and $\mathsf{s}^* = \mathsf{s}_j = \mathsf{SS.Gen}(\mathsf{w} + \delta_j)$, so we have that $\mathsf{w} + \delta_j = \mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*)$ by the correctness of $(\mathcal{M}, m, \widetilde{m}, 2t)$-secure sketch. Meanwhile, by the linearity property we have $\mathsf{SS.Rec}(\mathsf{w} + \delta^*, \mathsf{s}^*) = \mathsf{w} + \widetilde{\delta}^*$, where $\widetilde{\delta}^* = g(\delta^*, \mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*)$. As a result, $\delta_j = \widetilde{\delta}^*$ and $\Delta_j = \mathsf{H_i}(\delta_j) = \mathsf{H_i}(\widetilde{\delta}^*) = \Delta^*$. Hence the key deriving function $\phi_{\Delta_j} = \phi_{\Delta^*}$, and the special rule is satisfied.

As a result $\Pr[\mathbf{G}_2] = \mathsf{Adv}^{\mathsf{int\text{-}rka}}_{\mathcal{B},\mathsf{AIAE}}(\lambda) \leq \mathsf{Adv}^{\mathsf{int\text{-}rka}}_{\mathsf{AIAE}}(\lambda)$. The claim follows. □

Taking Eq. (26), Eq. (27), Claim 7 and Claim 8 together, we have $\mathsf{Adv}^{\mathsf{reu}}_{\mathcal{A},\mathsf{FE}}(\lambda) \leq \mathsf{Adv}^{\mathsf{int\text{-}rka}}_{\mathsf{AIAE}}(\lambda) + 2^{-\omega(\log \lambda)}$. Lemma 5 follows. ▌ □

6 Instantiations

6.1 Instantiation of $\mathsf{rrFE}_{\mathsf{prf}}$

We recall the unique-input $\Phi_{\mathsf{ln\text{-}aff}}$-RKA-secure PRF for an affine class $\Phi_{\mathsf{ln\text{-}aff}}$ in [18]. For $\mathfrak{m}, p, q \in \mathbb{N}$ such that $p|q$, the public parameters $\mathsf{pp}_{\mathsf{PRF}}$ is a pair of matrices of the form $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}}$, where each row of $\mathbf{A}_0$ and $\mathbf{A}_1$ is sampled uniformly from $\{0,1\}^{\mathfrak{m}}$. The secret key $\mathbf{K}$ is a matrix in $\mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}}$. Pseudo-random function $F_{\mathsf{LWE}} : \mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}} \times \{0,1\}^l \to \mathbb{Z}_p^{\mathfrak{m}\times\mathfrak{m}}$ is defined as

$$F_{\mathsf{LWE}}(\mathbf{K}, x) := \left\lfloor \mathbf{K} \cdot \prod_{i=1}^{l} \mathbf{A}_{x_i} \right\rfloor_p . \qquad (28)$$

Its security is based on the LWE assumption (see Appendix D.2 for the definition).

Theorem 3 ([18]). *Let $q = O(\sqrt{\lambda}/\alpha)$, $\mathfrak{m} = \lceil \lambda \log q \rceil$, $l = \lambda^\epsilon / \log \lambda$, $0 < \epsilon < 1$, $p = 2^{\lambda^\epsilon - \omega(\log \lambda)}$, $\alpha = 2^{-\lambda^\epsilon}$, $c, B > 0$ such that the quantity $(2\mathfrak{m})^l cBp/q$ is negligible in the security parameter λ. Under the $(\mathbb{Z}_q, \lambda, \Psi_\alpha)$-LWE assumption, the PRF* *defined in Eq. (28) is $\varPhi_{\mathsf{ln\text{-}aff}}$-RKA-secure against unique-input adversaries for the class* $\varPhi_{\mathsf{ln\text{-}aff}} := \{\phi_{\mathbf{C},\mathbf{B}} : \phi_{\mathbf{C},\mathbf{B}}(\mathbf{K}) = \mathbf{CK} + \mathbf{B} \mid \mathbf{C} \in [-c, c]^{\mathfrak{m}\times\mathfrak{m}}, \mathbf{B} \in \mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}}\}$.

Obviously, $\varPhi_{\mathsf{ln\text{-}aff}}$ covers the key shift function set $\varPhi_{\varDelta} := \{\phi_{\varDelta} : \phi_{\varDelta}(\mathbf{K}) = \mathbf{K} + \varDelta \mid \varDelta \in \mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}}\}$. Hence, F_{LWE} is a unique-input key-shift secure PRF.

Let $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathsf{Sample}(\mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}})$ denote sampling two matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}}$, where each row of $\mathbf{A}_0$ and $\mathbf{A}_1$ is sampled uniformly from $\{0,1\}^{\mathfrak{m}}$. By instantiating the PRF F_{pp} in Fig. 8 with F_{LWE}, the SS with the syndrome-based secure sketch scheme in Appendix A and $\mathcal{H}_{\mathcal{I}} = \{\mathsf{H}_{\mathsf{i}} : \mathcal{M} \to \mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}}\}_{i\in\mathcal{I}}$ with the universal hash function in Appendix B, and by setting $\mathcal{T} = \{0,1\}^{\omega(\log\lambda)}$, $l = \omega(\log\lambda)$, $|s| = |t| = \frac{l}{2}$, $\mathcal{R} = \mathbb{Z}_p^{\mathfrak{m}\times(\mathfrak{m}-1)}$, $\mathcal{V} = \mathbb{Z}_p^{\mathfrak{m}}$, we get a concrete construction of $\mathsf{rrFE}_{\mathsf{prf}}$ from the $(\mathbb{Z}_q, \lambda, \varPsi_\alpha)$-LWE assumption.

```
$\underline{\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)}$:
$\mathsf{i} \leftarrow_{\$} \mathcal{I}$ (i.e., $\mathsf{H_i} \leftarrow_{\$} \mathcal{H}_{\mathcal{I}}$).
$\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathsf{Sample}(\mathbb{Z}_q^{\mathfrak{m}\times\mathfrak{m}})$.
$\mathsf{crs} = (\mathsf{H_i}, \mathbf{A}_0, \mathbf{A}_1)$.
Return $\mathsf{crs}$.
```

```
$\underline{(\mathsf{P}, \mathsf{R}) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{w})}$:
$\mathsf{s} \leftarrow \mathsf{SS.Gen}(\mathsf{w})$.
$\mathbf{K} \leftarrow \mathsf{H_i}(\mathsf{w})$.
$\mathsf{t} \leftarrow_{\$} \mathcal{T}$, $x = (\mathsf{s}, \mathsf{t})$.
$F_{\mathsf{LWE}}(\mathsf{k}, x) := \left\lfloor \mathbf{K} \cdot \prod_{i=1}^{l} \mathbf{A}_{x_i} \right\rfloor_p = (\mathsf{r}, \mathsf{v})$.
$\mathsf{P} := (\mathsf{s}, \mathsf{t}, \mathsf{v})$, $\mathsf{R} := \mathsf{r}$.
```

```
$\underline{\mathsf{R} \leftarrow \mathsf{Rep}(\mathsf{crs}, \mathsf{P}, \mathsf{w}')}$:
Parse $\mathsf{P} = (\mathsf{s}, \mathsf{t}, \mathsf{v})$.
$\tilde{\mathsf{w}} \leftarrow \mathsf{SS.Rec}(\mathsf{w}', \mathsf{s})$.
$\tilde{\mathbf{K}} \leftarrow \mathsf{H_i}(\tilde{\mathsf{w}})$, $x = (\mathsf{s}, \mathsf{t})$.
$F_{\mathsf{LWE}}(\tilde{\mathbf{K}}, x) = (\tilde{\mathsf{r}}, \tilde{\mathsf{v}})$.
If $\tilde{\mathsf{v}} = \mathsf{v}$, Return $\mathsf{R} := \tilde{\mathsf{r}}$.
Else, Return $\perp$.
```

Fig. 14. Instantiation of $\mathsf{rrFE}_{\mathsf{prf}}$ from F_{LWE}: $\mathsf{rrFE}_{\mathsf{prf}_{\mathsf{lwe}}}$.

Corollary 1. *Scheme* $\mathsf{rrFE}_{\mathsf{prf}_{\mathsf{lwe}}}$ *in Fig. 14 is a robustly reusable fuzzy extractor based on the LWE assumption.*

The computational complexities of Gen and Rep of $\mathsf{rrFE}_{\mathsf{prf}}$ are dominated by the computation of the underlying PRF. According to [3], the best known running time of F_{LWE} is $O(\mathfrak{m}\lambda^5)$ per output bit. There are totally $\mathfrak{m}^2 \log p$ output bits, so the complexity is $O(\lambda^{11})$.

The length of P is given by $|P| = l + \mathfrak{m}\log p$, while the length of R is $|R| = \mathfrak{m}(\mathfrak{m}-1)\log p$.

Note that $|s| = \omega(\log\lambda)$, and this limits the error tolerance of SS. As a result, $\mathsf{rrFE}_{\mathsf{prf}_{\mathsf{lwe}}}$ can only support sub-linear fraction of errors.

6.2 Instantiation of $\mathsf{rrFE}_{\mathsf{AIAE}}$

We recall the construction of AIAE from one-time (OT) secure AE and the DDH assumption in [14]. Let $(\bar{N}, N, p, q) \leftarrow \mathsf{GenN}(1^\lambda)$ be a group generation algorithm, where p, q are 2λ-bit safe primes such that $\bar{N} = 2pq + 1$ is also a prime and $N = pq$. Let $\mathcal{H}_{\mathcal{I}_1} = \{\mathsf{H}_{\mathsf{i}_1} : \{0,1\}^* \to \mathbb{Z}_N\}_{\mathsf{i}_1\in\mathcal{I}_1}$ and $\mathcal{H}_{\mathcal{I}_2} = \{\mathsf{H}_{\mathsf{i}_2} : \mathbb{QR}_{\bar{N}} \to \mathcal{K}_{\mathsf{AE}}\}_{\mathsf{i}_2\in\mathcal{I}_2}$ be two families of hash functions, where $\mathbb{QR}_{\bar{N}}$ is the subgroup of quadratic residues of $\mathbb{Z}_{\bar{N}}^*$. Let $\mathsf{AE} = (\mathsf{AE.Enc}, \mathsf{AE.Dec})$ be a OT-secure authenticated encryption scheme with key space $\mathcal{K}_{\mathsf{AE}}$ and message space $\mathcal{M}$. The scheme $\mathsf{AIAE} = (\mathsf{AIAE.Setup}, \mathsf{AIAE.Enc}, \mathsf{AIAE.Dec})$ in [14] is described as follows.

```
$\underline{\mathsf{pp}_{\mathsf{AIAE}} \leftarrow \mathsf{AIAE.Setup}(1^\lambda)}$:
$(\bar{N}, N, p, q) \leftarrow \mathsf{GenN}(1^\lambda)$.
$g_1, g_2 \leftarrow \mathbb{QR}_{\bar{N}}$.
$\mathsf{H}_{\mathsf{i}_1} \leftarrow_{\$} \mathcal{H}_{\mathcal{I}_1}, \mathsf{H}_{\mathsf{i}_2} \leftarrow_{\$} \mathcal{H}_{\mathcal{I}_2}$.
$\mathsf{pp}_{\mathsf{AIAE}} := (\bar{N}, N, p, q, g_1, g_2, \mathsf{H}_{\mathsf{i}_1}, \mathsf{H}_{\mathsf{i}_2})$.
Return $\mathsf{pp}_{\mathsf{AIAE}}$.
```

```
$\underline{(c_1, c_2, \chi) \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}, \mathsf{m}, \mathsf{aux})}$:
Parse $\mathsf{k} = (k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$.
$\alpha \leftarrow_{\$} \mathbb{Z}_N \setminus \{0\}$.
$(c_1, c_2) := (g_1^\alpha, g_2^\alpha) \in \mathbb{QR}_{\bar{N}}^2$.
$\beta := \mathsf{H}_{\mathsf{i}_1}(c_1, c_2, \mathsf{aux}) \in \mathbb{Z}_N$.
$\kappa := \mathsf{H}_{\mathsf{i}_2}(c_1^{k_1+k_3\beta} \cdot c_2^{k_2+k_4\beta}) \in \mathcal{K}_{\mathsf{AE}}$.
$\chi \leftarrow \mathsf{AE.Enc}(\kappa, \mathsf{m})$.
Return $(c_1, c_2, \chi)$.
```

```
$\underline{\mathsf{m}/\perp \mathsf{AIAE.Dec}(\mathsf{k}, (c_1, c_2, \chi), \mathsf{aux})}$:
Parse $\mathsf{k} = (k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$.
If $(c_1, c_2) \notin \mathbb{QR}_{\bar{N}}^2 \vee (c_1, c_2) = (1, 1)$,
Return $\perp$.
$\beta := \mathsf{H}_{\mathsf{i}_1}(c_1, c_2, \mathsf{aux}) \in \mathbb{Z}_N$.
$\kappa := \mathsf{H}_{\mathsf{i}_2}(c_1^{k_1+k_3\beta} \cdot c_2^{k_2+k_4\beta}) \in \mathcal{K}_{\mathsf{AE}}$.
$\mathsf{m}/\perp \leftarrow \mathsf{AE.Dec}(\kappa, \chi)$.
Return $\mathsf{m}/\perp$.
```

Fig. 15. Construction of DDH-based $\mathsf{AIAE}_{\mathsf{ddh}}$ from OT-secure AE.

Theorem 4. *[14] If the underlying* AE *is OT-secure, the DDH assumption holds w.r.t.* GenN *over* $\mathbb{QR}_{\bar{N}}$*,* $\mathcal{H}_{\mathcal{I}_1}$ *is collision resistant and* $\mathcal{H}_{\mathcal{I}_2}$ *is universal, then* $\mathsf{AIAE}_{\mathsf{ddh}}$ *in Fig. 15 is IND-*Φ_{raff}*-RKA and weak INT-*Φ_{raff}*-RKA secure, where* $\Phi_{\mathsf{raff}} := \{\phi_{a,\mathsf{b}} : (k_1, k_2, k_3, k_4) \in \mathbb{Z}_N^4 \mapsto (ak_1 + b_1, ak_2 + b_2, ak_3 + b_3, ak_4 + b_4) \in \mathbb{Z}_N^4 \mid a \in \mathbb{Z}_N^*, \mathsf{b} = (b_1, b_2, b_3, b_4) \in \mathbb{Z}_N^4\}$.

Clearly, the key deriving function set Φ_{raff} contains the key-shift function set $\Phi_\Delta := \{\phi_\Delta : (k_1, k_2, k_3, k_4) \in \mathbb{Z}_N^4 \mapsto (k_1+b_1, k_2+b_2, k_3+b_3, k_4+b_4) \in \mathbb{Z}_N^4 \mid \Delta = (b_1, b_2, b_3, b_4) \in \mathbb{Z}_N^4\}$. So the $\mathsf{AIAE}_{\mathsf{ddh}}$ in Fig. 15 is Key-Shift secure. In $\mathsf{AIAE}_{\mathsf{ddh}}$, the building block AE can be instantiated with OT-secure AE in Appendix C just like [17], where $\kappa = (\kappa_1, \kappa_2, \kappa_3) \in \{0,1\}^{3\lambda}$, $\mathcal{M}_{\mathsf{AE}} = \{0,1\}^\lambda$ and $\chi \in \{0,1\}^{2\lambda}$.

By instantiating the AIAE in Fig. 11 with $\mathsf{AIAE}_{\mathsf{ddh}}$, the SS with the syndrome-based secure sketch scheme in Appendix A and $\mathcal{H}_{\mathcal{I}} = \{\mathsf{H}_{\mathsf{i}} : \mathcal{M} \to \mathbb{Z}_N^4\}_{i \in \mathcal{I}}$ with the universal hash function in Appendix B, we get a concrete construction of $\mathsf{rrFE}_{\mathsf{AIAE}}$ from the DDH assumption (see Fig. 16).

Corollary 2. *Scheme* $\mathsf{rrFE}_{\mathsf{AIAE}_{\mathsf{ddh}}}$ *in Fig. 16 is a robustly reusable fuzzy extractor based on the DDH assumption.*

The computational complexities of Gen and Rep are dominated by the encryption and decryption algorithms of the underlying $\mathsf{AIAE}_{\mathsf{ddh}}$. Consequently, the complexity of Gen is dominated by four modular exponentiations while that of Rep by two modular exponentiations over $\mathbb{QR}_{\bar{N}}$.

```
$\underline{\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)}$:
$\mathsf{i} \leftarrow_{\$} \mathcal{I}$ (i.e., $\mathsf{H}_{\mathsf{i}} \leftarrow_{\$} \mathcal{H}_{\mathcal{I}}$).
$(\bar{N}, N, p, q) \leftarrow \mathsf{Gen}(1^\lambda)$.
$g_1, g_2 \leftarrow \mathbb{QR}_{\bar{N}}$.
$\mathsf{H}_{\mathsf{i}_1} \leftarrow_{\$} \mathcal{H}_{\mathcal{I}_1}, \mathsf{H}_{\mathsf{i}_2} \leftarrow_{\$} \mathcal{H}_{\mathcal{I}_2}$.
$\mathsf{pp}_{\mathsf{AIAE}} := (\bar{N}, N, p, q, g_1, g_2, \mathsf{H}_{\mathsf{i}_1}, \mathsf{H}_{\mathsf{i}_2})$.
$\mathsf{crs} = (\mathsf{H}_{\mathsf{i}}, \mathsf{pp}_{\mathsf{AIAE}})$.
Return $\mathsf{crs}$.
```

```
$\underline{(\mathsf{P}, \mathsf{R}) \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathsf{w})}$:
$\mathsf{s} \leftarrow \mathsf{SS.Gen}(\mathsf{w})$.
$\mathsf{k} \leftarrow \mathsf{H}_{\mathsf{i}}(\mathsf{w})$.
$\mathsf{m} \leftarrow_{\$} \mathcal{M}_{\mathsf{AIAE}}$.
Parse $\mathsf{k} = (k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$.
$\alpha \leftarrow_{\$} \mathbb{Z}_N \setminus \{0\}$.
$(c_1, c_2) := (g_1^w, g_2^w) \in \mathbb{QR}_{\bar{N}}^2$.
$t := \mathsf{H}_{\mathsf{i}_1}(c_1, c_2, \mathsf{s}) \in \mathbb{Z}_N$.
$\kappa := \mathsf{H}_{\mathsf{i}_2}(c_1^{k_1+k_3t} \cdot c_2^{k_2+k_4t}) \in \mathcal{K}_{\mathsf{AE}}$.
$\chi \leftarrow \mathsf{AE.Enc}(\kappa, \mathsf{m})$.
$\mathsf{P} := (\mathsf{s}, c_1, c_2, \chi), \mathsf{R} := \mathsf{m}$.
```

```
$\underline{\mathsf{R} \leftarrow \mathsf{Rep}(\mathsf{crs}, \mathsf{P}, \mathsf{w}')}$:
Parse $\mathsf{P} = (\mathsf{s}, c_1, c_2, \chi)$.
$\tilde{\mathsf{w}} \leftarrow \mathsf{SS.Rec}(\mathsf{w}', \mathsf{s})$.
$\tilde{\mathsf{k}} \leftarrow \mathsf{H}_{\mathsf{i}}(\tilde{\mathsf{w}})$.
Parse $\tilde{\mathsf{k}} = (\tilde{k}_1, \tilde{k}_2, \tilde{k}_3, \tilde{k}_4) \in (\mathbb{Z}_N)^4$.
If $(c_1, c_2) \notin \mathbb{QR}_{\bar{N}}^2 \vee (c_1, c_2) = (1, 1)$,
Return $\perp$.
$\beta := \mathsf{H}_{\mathsf{i}_1}(c_1, c_2, \mathsf{s}) \in \mathbb{Z}_N$.
$\kappa := \mathsf{H}_{\mathsf{i}_2}(c_1^{\tilde{k}_1+\tilde{k}_3\beta} \cdot c_2^{\tilde{k}_2+\tilde{k}_4\beta}) \in \mathcal{K}_{\mathsf{AE}}$.
$\tilde{\mathsf{m}}/\perp \leftarrow \mathsf{AE.Dec}(\kappa, \chi)$.
Return $\tilde{\mathsf{m}}/\perp$.
```

Fig. 16. Instantiation of $\mathsf{rrFE}_{\mathsf{AIAE}}$ from $\mathsf{AIAE}_{\mathsf{ddh}}$: $\mathsf{rrFE}_{\mathsf{AIAE}_{\mathsf{ddh}}}$.

Observe that the ciphertext ct of $\mathsf{AIAE}_{\mathsf{ddh}}$ in Fig. 15 is of size $(4\lambda+1)+(4\lambda+1)+2\lambda = 10\lambda+2$. So the public string P of $\mathsf{rrFE}_{\mathsf{AIAE}_{\mathsf{ddh}}}$ in Fig. 16 has $|s|+10\lambda+2$ bits, where $|s|$ depend on the maximal number of errors t. Note that $\mathsf{AIAE}_{\mathsf{ddh}}$ is very efficient, so this instantiation $\mathsf{rrFE}_{\mathsf{AIAE}_{\mathsf{ddh}}}$ in Fig. 16 is very efficient as well.

Since the syndrome-based secure sketch in Appendix A can correct linear fraction of errors and there is no further limits on the length of s, the resulting $\mathsf{rrFE}_{\mathsf{AIAE}_{\mathsf{ddh}}}$ in Fig. 16 can support linear fraction of errors.

Acknowledgements. We would like to thank the reviewers for their valuable comments. Yunhua Wen and Shengli Liu are supported by the National Natural Science Foundation of China (Grant No. 61672346). Dawu Gu is sponsored by the Natural Science Foundation of China (Grant No. 61472250) and Program of Shanghai Academic Research Leader (16XD1401300).

References

[1] Alamélou, Q., Berthier, P., Cachet, C., Cauchie, S., Fuller, B., Gaborit, P., Simhadri, S.: Pseudoentropic isometries: A new framework for fuzzy extractor reusability. In: Kim, J., Ahn, G., Kim, S., Kim, Y., López, J., Kim, T. (eds.) AsiaCCS 2018. pp. 673–684. ACM (2018), http://doi.acm.org/10.1145/3196494.3196530

[2] Apon, D., Cho, C., Eldefrawy, K., Katz, J.: Efficient, reusable fuzzy extractors from LWE. In: Dolev, S., Lodha, S. (eds.) CSCML 2017. LNCS, vol. 10332, pp. 1–18. Springer (2017), https://doi.org/10.1007/978-3-319-60080-2_1

[3] Banerjee, A., Peikert, C.: New and improved key-homomorphic pseudorandom functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 353–370. Springer (2014), https://doi.org/10.1007/978-3-662-44371-2_20

[4] Bennett, C.H., DiVincenzo, D.P.: Quantum information and computation. Nature **404**(6775), 247–255 (2000)

[5] Bennett, C.H., Shor, P.W.: Quantum information theory. IEEE Trans. Information Theory **44**(6), 2724–2742 (1998), https://doi.org/10.1109/18.720553

[6] Boyen, X.: Reusable cryptographic fuzzy extractors. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) CCS 2004. pp. 82–91. ACM (2004), http://doi.acm.org/10.1145/1030083.1030096

[7] Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.D.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer (2005), https://doi.org/10.1007/11426639_9

[8] Canetti, R., Fuller, B., Paneth, O., Reyzin, L., Smith, A.D.: Reusable fuzzy extractors for low-entropy distributions. In: Fischlin, M., Coron, J. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 117–146. Springer (2016), https://doi.org/10.1007/978-3-662-49890-3_5

[9] Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 471–488. Springer (2008), https://doi.org/10.1007/978-3-540-78967-3_27

[10] Dodis, Y., Katz, J., Reyzin, L., Smith, A.D.: Robust fuzzy extractors and authenticated key agreement from close secrets. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 232–250. Springer (2006), https://doi.org/10.1007/11818175_14

[11] Dodis, Y., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer (2004), https://doi.org/10.1007/978-3-540-24676-3_31

[12] Fuller, B., Meng, X., Reyzin, L.: Computational fuzzy extractors. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 174–193. Springer (2013), https://doi.org/10.1007/978-3-642-42033-7_10

[13] Galbraith, S.: New discrete logarithm records, and the death of type 1 pairings, https://ellipticnews.wordpress.com/2014/02/01/new-discrete-logarithm-records-and-the-death-of-type-1-pairings/

[14] Han, S., Liu, S., Lyu, L.: Efficient KDM-CCA secure public-key encryption for polynomial functions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 307–338 (2016), https://doi.org/10.1007/978-3-662-53890-6_11

[15] Hofheinz, D.: Circular chosen-ciphertext security with compact ciphertexts. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 520–536. Springer (2013), https://doi.org/10.1007/978-3-642-38348-9_31

[16] Kanukurthi, B., Reyzin, L.: An improved robust fuzzy extractor. In: Ostrovsky, R., Prisco, R.D., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 156–171. Springer (2008), https://doi.org/10.1007/978-3-540-85855-3_11

[17] Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer (2004), https://doi.org/10.1007/978-3-540-28628-8_26

[18] Lewi, K., Montgomery, H.W., Raghunathan, A.: Improved constructions of prfs secure against related-key attacks. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 44–61. Springer (2014), https://doi.org/10.1007/978-3-319-07536-5_4

[19] Li, S.Z., Jain, A.K. (eds.): Handbook of Face Recognition, 2nd Edition. Springer (2011), https://doi.org/10.1007/978-0-85729-932-1

[20] Marasco, E., Ross, A.: A survey on antispoofing schemes for fingerprint recognition systems. ACM Comput. Surv. **47**(2), 28:1–28:36 (2014), https://doi.org/10.1145/2617756

[21] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) STOC 2005. pp. 84–93. ACM (2005), http://doi.acm.org/10.1145/1060590.1060603

[22] Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: CCS 2010. pp. 237–249 (2010), http://doi.acm.org/10.1145/1866307.1866335

[23] Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th annual Design Automation Conference. pp. 9–14 (2007)

[24] Wen, Y., Liu, S.: Reusable fuzzy extractor from LWE. In: Susilo, W., Yang, G. (eds.) ACISP 2018. LNCS, vol. 10946, pp. 13–27. Springer (2018), https://doi.org/10.1007/978-3-319-93638-3_2

[25] Wen, Y., Liu, S.: Robustly reusable fuzzy extractor from standard assumptions. Cryptology ePrint Archive, Report 2018/818 (2018), https://eprint.iacr.org/2018/818

[26] Wen, Y., Liu, S., Han, S.: Reusable fuzzy extractor from the decisional diffie-hellman assumption. Des. Codes Cryptography **86**(11), 2495–2512 (2018), https://doi.org/10.1007/s10623-018-0459-4

Supplementary Materials

A Homomorphic Secure Sketch with Linear Property

Recall that an efficiently decodable $[n, k, 2t + 1]_{\mathbb{F}}$-linear error correcting code $\mathcal{C}$ can correct up to t errors and it is a linear subspace of $\mathbb{F}^n$ of dimension k. The parity-check matrix of $\mathcal{C}$ is an $(n-k) \times n$ matirx H whose rows generate the orthogonal space $\mathcal{C}^{\perp}$. For any $v \in \mathbb{F}^n$, the syndrome of v is defined by $\mathsf{syn}(v) := Hv$. Note that $v \in \mathcal{C} \iff \mathsf{syn}(v) = 0$. For any $c \in \mathcal{C}$, $\mathsf{syn}(c + e) = \mathsf{syn}(c) + \mathsf{syn}(e) = \mathsf{syn}(e)$.

A linear error-correcting code implies a syndrome-based secure sketch as shown below.

– $\mathsf{SS.Gen}(\mathsf{w}) := \mathsf{syn}(\mathsf{w}) = \mathsf{s}$.
– $\mathsf{SS.Rec}(\mathsf{w}', \mathsf{s}) := \mathsf{w}' - \mathsf{Decode}(\mathsf{syn}(\mathsf{w}') - \mathsf{s})$.

Lemma 6 ([11]). *Given an $[n, k, 2t+1]_{\mathbb{F}}$ linear error-correcting code over field $\mathbb{F}$, one can construct a $(m, m-n+k, t)$-secure sketch for $\mathbb{F}^n$. The secure sketch is deterministic and its output consists of $n-k$ elements of $\mathbb{F}$.*

Homomorphic Property. The syndrome-based secure sketch is homomorphic [25], since $\mathsf{SS.Gen}(\mathsf{w}+\delta) = \mathsf{syn}(\mathsf{w}+\delta) = H(\mathsf{w}+\delta) = H\mathsf{w}+H\delta = \mathsf{syn}(\mathsf{w})+\mathsf{syn}(\delta) = \mathsf{SS.Gen}(\mathsf{w}) + \mathsf{SS.Gen}(\delta)$.
Linearity Property. The syndrome-based secure sketch has linearity property [9], since

$$\begin{aligned}\tilde{\delta} &= \tilde{\mathsf{w}} - \mathsf{w} = \mathsf{SS.Rec}(\mathsf{w}', \tilde{\mathsf{s}}) - \mathsf{w} \\ &= \mathsf{w}' - \mathsf{Decode}(\mathsf{syn}(\mathsf{w}') - \tilde{\mathsf{s}}) - \mathsf{w} \\ &= \delta - \mathsf{Decode}(\mathsf{syn}(\mathsf{w} + \delta) - \tilde{\mathsf{s}}) \\ &= \delta - \mathsf{Decode}(\mathsf{s} + \mathsf{syn}(\delta) - \tilde{\mathsf{s}}).\end{aligned}$$

Define $g(\delta, \mathsf{s}, \tilde{\mathsf{s}}) := \delta - \mathsf{Decode}(\mathsf{s} + \mathsf{syn}(\delta) - \tilde{\mathsf{s}})$. Clearly, $g(\cdot,\cdot,\cdot)$ is an efficient deterministic function.

B Homomorphic Universal Hash Functions

Let q be a prime. For $\mathbf{w} \in \mathbb{Z}_q^l, \mathbf{A} \in \mathbb{Z}_q^{n \times l}$, define

$$\mathsf{H}_{\mathbf{A}}(\mathbf{w}) := \mathbf{A}\mathbf{w},$$

then $\mathcal{H} = \{\mathsf{H}_{\mathbf{A}} : \mathbb{Z}_q^l \to \mathbb{Z}_q^n \mid \mathbf{A} \in \mathbb{Z}_q^{n \times l}\}$ is a family of universal hash functions. Note that the above hash function is homomorphic, since

$$\mathsf{H}_{\mathbf{A}}(\mathbf{w} + \mathbf{w}') = \mathbf{A}(\mathbf{w} + \mathbf{w}') = \mathbf{A}\mathbf{w} + \mathbf{A}\mathbf{w}' = \mathsf{H}_{\mathbf{A}}(\mathbf{w}) + \mathsf{H}_{\mathbf{A}}(\mathbf{w}').$$

One can readily interpret a vector in $\mathbb{Z}_q^n$ as a matrix in $\mathbb{Z}_q^{\sqrt{n}\times\sqrt{n}}$, if $\sqrt{n}$ is an integer.

Let $\bar{N} = 2N + 1$ be a prime. Define $f : \mathbb{Z}_{\bar{N}}^n \to \mathbb{Z}_N^n$ with $f\left((x_1, \ldots, x_n)^\top\right) := (x_1 \mod N, \ldots, x_n \mod N)^\top$. One can easily get a family of hash functions $\mathcal{H}' = \{f \circ \mathsf{H}_{\mathbf{A}} : \mathbb{Z}_{\bar{N}}^l \to \mathbb{Z}_N^n \mid \mathbf{A} \in \mathbb{Z}_{\bar{N}}^{n \times l}\}$, where $f(\mathsf{H}_{\mathbf{A}}(\mathbf{w})) := f(\mathbf{A}\mathbf{w})$. It is easy to check that $\mathcal{H}'$ is almost universal.

C One-Time Secure Authenticated Encryption

Definition 13 (Authenticated Encryption). *An authenticated encryption scheme* AE *is associated with a message space* $\mathcal{M}_{\mathsf{AE}}$ *and a key space* $\mathcal{K}_{\mathsf{AE}}$*, and consists of a pair of algorithms:*

- $\mathsf{AE.Enc}(\mathsf{k}, \mathsf{m})$ *on input a key* $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$*, a message* $\mathsf{m} \in \mathcal{M}_{\mathsf{AE}}$ *outputs a ciphertext* ct*.*
- $\mathsf{AE.Dec}(\mathsf{k}, \mathsf{ct})$ *on input a key* k *and a ciphertext* ct *outputs a message* m *or a rejection symbol* $\perp$*.*

Correctness. For all $k \in \mathcal{K}_{\mathsf{AE}}$, all $\mathsf{m} \in \mathcal{M}_{\mathsf{AE}}$ and all $\mathsf{ct} \leftarrow \mathsf{AE.Enc}(\mathsf{k}, \mathsf{m})$, it holds that $\mathsf{m} = \mathsf{AE.Dec}(\mathsf{k}, \mathsf{ct})$.

Definition 14 (IND-OT and INT-OT Securities for AE). *An* AE *scheme is one-time secure if it is* IND-OT *and* INT-OT *secure. More precisely, for any PPT adversary* $\mathcal{A}$*, both* $\mathsf{Adv}^{\mathsf{ind\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda) \Rightarrow 1] - 1/2|$ *and* $\mathsf{Adv}^{\mathsf{int\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{int\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda) \Rightarrow 1]$ *are negligible, where games* $\mathsf{Exp}^{\mathsf{ind\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{int\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda)$ *are depicted in Fig. 17.*

```
Procedure INITIALIZE:
$\mathsf{k} \leftarrow_\$ \mathcal{K}_{\mathsf{AE}}$.
$b \leftarrow_\$ \{0,1\}$.
Return $\varepsilon$.

Procedure LR($\mathsf{m}_0, \mathsf{m}_1$):    // one query
If $|\mathsf{m}_0| \neq |\mathsf{m}_1|$, Return $\perp$.
$\mathsf{ct} \leftarrow \mathsf{AE.Enc}(\mathsf{k}, \mathsf{m}_b)$.
Return $\mathsf{ct}$.

Procedure FINALIZE($b^*$)
If $b = b^*$, Return 1.
Else, Return 0.
```

```
Procedure INITIALIZE:
$\mathsf{k} \leftarrow_\$ \mathcal{K}_{\mathsf{AE}}$.
Return $\varepsilon$.

Procedure ENC($\mathsf{m}$):    // one query
$\mathsf{ct} \leftarrow \mathsf{AE.Enc}(\mathsf{k}, \mathsf{m})$.
Return $\mathsf{ct}$.

Procedure FINALIZE($\mathsf{ct}^*$)
$\mathsf{ct}^* = \mathsf{ct}$, Return 0.
Return $(\mathsf{AE.Dec}(\mathsf{k}, \mathsf{ct}^*) \neq \perp)$.
```

Fig. 17. Security games for AE. Left: $\mathsf{Exp}^{\mathsf{ind\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda)$; Right: $\mathsf{Exp}^{\mathsf{int\text{-}ot}}_{\mathcal{A},\mathsf{AE}}(\lambda)$.

Now we present the one-time secure AE in [17].

Suppose that the key space is given by $\mathcal{K}_{\mathsf{AE}} := \{0,1\}^{3\lambda}$, the message space given by $\mathcal{M}_{\mathsf{AE}} = \{0,1\}^{\lambda}$. A key $\mathsf{k} = (k_0, k_1, k_2)$ is randomly sampled from $\{0,1\}^{3\lambda}$.

– $\mathsf{AE.Enc}(\mathsf{k}=(k_0,k_1,k_2),\mathsf{m}):\ e=k_0+\mathsf{m}, a=k_1\cdot e+k_2, \mathsf{ct}:=(e,a)$. Return ct.

– $\mathsf{AE.Dec}(\mathsf{k}=(k_0,k_1,k_2),\mathsf{ct}=(e,a)):$ If $a\neq k_1\cdot e+k_2$, return $\perp$, else return $\mathsf{m}:=e-k_1$.

The multiplication and addition are carried over $\mathbb{F}_{2^\lambda}$.

D Assumptions

D.1 Decisional Diffie-Hellman (DDH) Assumption

Let $(\bar{N},N,p,q)\leftarrow\mathsf{GenN}(1^\lambda)$ be a group generation algorithm, where p,q are 2λ-bit safe primes such that $\bar{N}=2pq+1$ is also a prime and $N=pq$.

Definition 15 (DDH Assumption). *The Decisional Diffie-Hellman (DDH) assumption holds over group $\mathbb{QR}_{\bar{N}}$ for GenN if for any PPT adversary $\mathcal{A}$, the following advantage is negligible in λ:*

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda):=|\Pr[\mathcal{A}(\bar{N},N,p,q,g_1,g_2,g_1^x,g_2^x)\Rightarrow 1]-\Pr[\mathcal{A}(\bar{N},N,p,q,g_1,g_2,g_1^x,g_2^y)\Rightarrow 1]|$$

where $(\bar{N},N,p,q)\leftarrow\mathsf{GenN}(1^\lambda)$, $g_1,g_2\leftarrow_{\$}\mathbb{QR}_{\bar{N}}$, $x,y\leftarrow_{\$}\mathbb{Z}_N\setminus\{0\}$.

D.2 Learning with Errors (LWE) Assumption

The learning with errors (LWE) problem was introduced by Regev [21].

Definition 16 (LWE Assumption). *Let integers $n=n(\lambda)$, $m=m(\lambda)$ and $q=q(\lambda)\geq 2$. Let $\chi(\lambda)$ be a distribution over $\mathbb{Z}_q$. The $(\mathbb{Z}_q,n,\chi)$-LWE problem is to distinguish the following two distributions,*

$$(\mathbf{A},\mathbf{A}\mathbf{s}+\mathbf{e})\quad \textit{and}\quad (\mathbf{A},\mathbf{u}),$$

where $\mathbf{A}\leftarrow_{\$}\mathbb{Z}_q^{m\times n}$, $\mathbf{s}\leftarrow_{\$}\mathbb{Z}_q^n$, $\mathbf{e}\leftarrow\chi^m$ and $\mathbf{u}\leftarrow_{\$}\mathbb{Z}_q^m$. The $(\mathbb{Z}_q,n,\chi)$-LWE assumption holds, if $(\mathbf{A},\mathbf{A}\mathbf{s}+\mathbf{e})\approx_c(\mathbf{A},\mathbf{u})$ for all PPT adversaries.

For an $\alpha\in(0,1)$ and a prime q, the distribution Ψ_α over $\mathbb{Z}_q$ is defined by $\lceil qY\rfloor(\mod q)$ where Y is a normal random variable with mean 0 and standard deviation $\alpha/2\pi$. Let $\mathsf{abs}(x)$ denote the absolute value of x. Let B be an error bound such that $\Pr[\mathsf{abs}(x)\leq B\mid x\leftarrow\Psi_\alpha]$ with overwhelming probability. Regev [21] showed that for noise distribution Ψ_α, if q is sufficiently large, the $(\mathbb{Z}_q,n,\Psi_\alpha)$-LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction.

E Omitted Proofs

E.1 Proof of Claim 5

Proof. Similar to the proof of Claim 1, the Leftover Hash Lemma implies that

$$\mathsf{SD}((\mathsf{H_i}(\mathsf{w}),\mathsf{i},\mathsf{s}=\mathsf{SS.Gen}(\mathsf{w}),(\mathsf{U},\mathsf{i},\mathsf{s}=\mathsf{SS.Gen}(\mathsf{w})))\leq\frac{1}{2}\sqrt{|\mathcal{K}|\cdot 2^{-\tilde{m}}},\qquad(29)$$

where $\mathsf{U} \leftarrow_{\$} \mathcal{K}$. In other words, for all powerful (not necessarily PPT) algorithm $\mathcal{B}$, it holds that

$$|\Pr[\mathcal{B}(U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1] - \Pr[\mathcal{B}((\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1]| \leq \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\tilde{m}}}. \tag{30}$$

We prove the claim by constructing a powerful algorithm $\mathcal{B}$ who aims to distinguish $(\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$ from $(\mathsf{U}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$. Given $(X, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$, where X is either $\mathsf{H_i}(\mathsf{w})$ or a uniform U, $\mathcal{B}$ simulates $\mathbf{G}_1/\mathbf{G}_2$ for $\mathcal{A}$ as follows.

– To simulate Procedure Initialize, $\mathcal{B}$ randomly chooses a bit $b \leftarrow_{\$} \{0,1\}$, then determines $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp_{AIAE}})$ for $\mathcal{A}$ by determining $\mathsf{H_i}$ with i and invoking $\mathsf{pp_{AIAE}} \leftarrow \mathsf{AIAE.Setup}(1^\lambda)$.
– To answer $\mathcal{A}$'s query δ_j, $\mathcal{B}$ simulates Procedure Challenge(δ_j) as follows.
  - If $\mathsf{dis}(\delta_j) > t$, return $\perp$.
  - $\mathsf{s}_j = \mathsf{s} + \mathsf{SS.Gen}(\delta_j)$.
  - $\mathsf{k}_j = X + \mathsf{H_i}(\delta_j)$.
  - $\mathsf{m}_j \leftarrow_{\$} \mathcal{M}_{\mathsf{AIAE}}$.
  - $\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}_j, \mathsf{m}_j, \mathsf{s}_j)$.
  - $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{ct}_j), \mathsf{R} := \mathsf{m}_j$.
  - If $b = 1$, return $(\mathsf{P}_j, \mathsf{R}_j)$. Else, $\mathsf{U}_j \leftarrow_{\$} \mathcal{M}_{\mathsf{AIAE}}$, return $(\mathsf{P}_j, \mathsf{U}_j)$.
– Finally $\mathcal{A}$ outputs a guessing bit of b^*. If $b = b^*$ (i.e., $\mathcal{A}$ wins), then $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs 0.

If $X = \mathsf{H_i}(\mathsf{w})$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_1$ for $\mathcal{A}$; if $X = \mathsf{U}$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$. Consequently,

$$\begin{aligned} &|\Pr[\mathcal{B}(\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1] - \Pr[\mathcal{B}((\mathsf{U}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1]| \\ =& |\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]|. \end{aligned} \tag{31}$$

Obviously, the claim follows from Eq. (30), Eq. (31) and the fact of $\widetilde{m} - \log|\mathcal{K}| \geq \omega(\log\lambda)$. □

E.2 Proof of Claim 7

Proof. Similar to that of Claim 3, the Leftover Hash Lemma implies that

$$\mathsf{SD}((\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}), (\mathsf{U}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))) \leq \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\tilde{m}}}, \tag{32}$$

where $\mathsf{U} \leftarrow_{\$} \mathcal{K}$. In other words, for all powerful (not necessarily PPT) algorithm $\mathcal{B}$, it holds that

$$|\Pr[\mathcal{B}(\mathsf{U}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1] - \Pr[\mathcal{B}((\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1]| \leq \frac{1}{2}\sqrt{|\mathcal{K}| \cdot 2^{-\tilde{m}}}. \tag{33}$$

We construct a powerful algorithm $\mathcal{B}$ who aims to distinguish $(\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$ from $(\mathsf{U}, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w}))$. Suppose that the challenge of $\mathcal{B}$ is $(X, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})$, where X is either $\mathsf{H_i}(\mathsf{w})$ or a uniform U. Then $\mathcal{B}$ simulates $\mathbf{G}_1/\mathbf{G}_2$ for $\mathcal{A}$ as follows.

– To simulate Procedure Initialize, $\mathcal{B}$ randomly chooses a bit $b \leftarrow_{\$} \{0,1\}$, then determines $\mathsf{crs} = (\mathsf{H_i}, \mathsf{pp_{AIAE}})$ for $\mathcal{A}$ by determining $\mathsf{H_i}$ with i and invoking $\mathsf{pp_{AIAE}} \leftarrow \mathsf{AIAE.Setup}(1^\lambda)$.
– To answer $\mathcal{A}$'s query δ_j, $\mathcal{B}$ simulates Procedure Challenge(δ_j) as follows.
  - If $\mathsf{dis}(\delta_j) > t$, return $\perp$.
  - $\mathsf{s}_j = \mathsf{s} + \mathsf{SS.Gen}(\delta_j)$.
  - $\mathsf{k}_j = X + \mathsf{H_i}(\delta_j)$.
  - $\mathsf{m}_j \leftarrow_{\$} \mathcal{M}_{\mathsf{AIAE}}$.
  - $\mathsf{ct}_j \leftarrow \mathsf{AIAE.Enc}(\mathsf{k}_j, \mathsf{m}_j, \mathsf{s}_j)$.
  - $\mathsf{P}_j := (\mathsf{s}_j, \mathsf{ct}_j), \mathsf{R} := \mathsf{m}_j$.
  - Return $(\mathsf{P}_j, \mathsf{R}_j)$.
– Finally $\mathcal{A}$ sends $(\mathsf{P}^* = (\mathsf{s}^*, \mathsf{ct}^*), \delta^*)$ to Finalize. If $\mathsf{dis}(\delta^*) > t$ or $\mathsf{P}^* \in \mathcal{Q}$, $\mathcal{B}$ returns 0 to its own challenger. Else, $\mathcal{B}$ parses $\mathsf{P}^* = (\mathsf{s}^*, \mathsf{ct}^*)$, and computes $\widetilde{\delta^*} = g(\mathsf{SS.Gen}(\mathsf{w}), \mathsf{s}^*, \delta^*)$ and $\tilde{\mathsf{k}} = X + \mathsf{H_i}(\widetilde{\delta^*})$. If $\mathsf{AIAE.Dec}(\tilde{\mathsf{k}}, \mathsf{ct}^*, \mathsf{s}^*) \neq \perp$, $\mathcal{B}$ returns 1, else returns 0.

If $X = \mathsf{H_i}(\mathsf{w})$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_1$ for $\mathcal{A}$; if $X = \mathsf{U}$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$. Consequently,

$$\begin{aligned} &|\Pr[\mathcal{B}(\mathsf{H_i}(\mathsf{w}), \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1] - \Pr[\mathcal{B}((U, \mathsf{i}, \mathsf{s} = \mathsf{SS.Gen}(\mathsf{w})) \Rightarrow 1]| \\ = &|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]|. \end{aligned} \tag{34}$$

Therefore, Claim 7 follows from Eq. (33), Eq. (34) and the fact of $\widetilde{m} - \log|\mathcal{K}| \geq \omega(\log\lambda)$. □