Robustly Reusable Fuzzy Extractor from Standard Assumptions

Yunhua Wen[1] and Shengli Liu[1,2,3]

[1] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{happyle8, slliu}@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. A fuzzy extractor (FE) aims at deriving and reproducing (almost) uniform cryptographic keys from noisy non-uniform sources. To reproduce an identical key R from subsequent readings of a noisy source, it is necessary to eliminate the noises from those readings. To this end, a public helper string P, together with the key R, is produced from the first reading of the source during the initial enrollment phase.

In this paper, we consider computational fuzzy extractor. We formalize *robustly reusable fuzzy extractor* (rrFE) which considers *reusability* and *robustness* simultaneously in the Common Reference String (CRS) model. Reusability of rrFE deals with source reuse. It guarantees that the key R output by fuzzy extractor is pseudo-random even if the initial enrollment is applied to the same source several times, generating multiple public helper strings and keys (P_i, R_i). Robustness of rrFE deals with active probabilistic polynomial-time adversaries, who may manipulate the public helper string P_i to affect the reproduction of R_i. Any modification of P_i by the adversary will be detected by the robustness of rrFE.

- We show how to construct an rrFE from a Symmetric Key Encapsulation Mechanism (SKEM), a Secure Sketch (SS), an Extractor (Ext), and a Lossy Algebraic Filter (LAF). We characterize the key-shift security notion of SKEM and the homomorphic properties of SS, Ext and LAF, which enable our construction of rrFE to achieve both reusability and robustness.
- We present an instantiation of SKEM from the DDH assumption. Combined with the LAF by Hofheinz (EuroCrypt 2013), homomorphic SS and Ext, we obtain the first rrFE based on standard assumptions.

Keywords: Fuzzy extractor, Reusability, Robustness, Standard assumptions

1 Introduction

Uniformly distributed keys are pivots of cryptographic primitives. However, it is not easy for us to create, memorize and safely store random keys. In practice, there are plenty of noisy sources, which possess high entropy and provide similar but not identical reading at each enrollment. Such sources include biometrics like fingerprint, iris, face and voice [9,17,19,20], Physical Unclonable Functions [21,23] and quantum sources [3,16]. How to make use of these noisy sources to derive uniform and reproducible keys for cryptographic applications is exactly the concern of Fuzzy Extractors [12].

Fuzzy extractor. A fuzzy extractor FE consists of a pair of algorithms $(\mathsf{Gen}, \mathsf{Rep})$. It works as follows. The generation algorithm Gen takes as input a reading w of some source and outputs a public helper string P and an extracted key R. The reproduction algorithm Rep takes as input the public helper string P and a reading w' of the same source (w' is a noisy version of w). It reproduces R if w and w' are close enough. The security of fuzzy extractor requires that R is statistically (or computationally) indistinguishable from a uniform one, even conditioned on the public helper string P.

With a fuzzy extractor FE, one may invoke Gen to generate a random key R and a public helper string P from a noisy source, then he stores the helper string P (publicly), and uses the key R in a cryptographic application. Note that it is not necessary for the user to store R. Whenever key R is needed again, he just re-reads the (noisy) source and invokes Rep to reproduce R with the help of P.

However, there are two limitations of FE, leading to two issues.

- The extracted key R is (pseudo)random under the assumption that no more than a single extraction is performed on the noisy source by Gen. In reality, biometric information, like fingerprint or iris, is unique and cannot be changed or created. One may hope that the same source is enrolled multiple times by Gen to generate different keys $R_1, R_2, \ldots, R_\rho$ for different applications. But no security guarantee can be provided for any R_i if $\rho \geq 2$.
- The security notion of FE only considers passive adversary and says nothing about active attacks. If the public helper string P is modified by an active adversary, then the reproduction algorithm Rep may generate a wrong key $\widetilde{R}$. In this case, one might not realize that $\widetilde{R}$ is a wrong one, and it may lead to unbearable economic loss.

The first issue can be resolved by reusable FE and the second by robust FE.

Reusable Fuzzy Extractor. Reusable Fuzzy Extractor aims to address the first issue. It allows of multiple extractions from the same source, i.e., apply Gen to correlated readings $w, w_1, \ldots, w_\rho$ of a source to obtain keys and public helper strings (P, R) $\{P_i, R_i\}_{i \in \{1,2,\ldots,\rho\}}$. Define $[\rho] := \{1, 2, \ldots, \rho\}$. *Reusability* of FE asks for pseudorandomness of R, even conditioned on $\{P_i, R_i\}_{i\in[\rho]}$ and P.

The concept of reusable FE was first proposed by Boyen [4], who presented two reusable FE constructions with outsider security and insider security respectively. Outsider security considers the pseudorandomness of R even if the adversary is able to adaptively choose δ_i and see P_i (but not R_i), where $(P_i, R_i) \leftarrow \mathsf{Gen}(w + \delta_i)$. It can be regarded as weak reusability in the sense that the adversary sees only $\{P_i\}_{i\in[\rho]}$. Insider security is stronger by allowing the adversary to obtain not only $\{P_i\}_{i\in[\rho]}$ but also $\tilde{R}_i \leftarrow \mathsf{Rep}(\tilde{P}_i, w+\tilde{\delta}_i)$ where $\tilde{P}_i$ and $\tilde{\delta}_i$ are chosen by the adversary. However, the construction for insider security in [4] relies on the random oracle model. Meanwhile, the perturbation δ_i in the reusable FE constructions [4] is very special and independent of w, no matter for outsider security or insider security. Apon et al. [2] adapted the FE proposed by Fuller et al. [14] to obtain a weakly reusable FE. They also gave a reusable FE based on the LWE assumption. Their security model is similar to [4] but has no special requirements on δ_i except that $\mathsf{dis}(\delta_i) \leq \mathsf{t}$. However, just like [14] their reusable FE can only tolerate a logarithmic fraction of errors. With the same security model, a reusable FE tolerating linear fraction of errors from the LWE assumption was proposed in [24].

Canetti et al. [6] constructed a reusable FE for Hamming distance. The security model of their reusable FE makes no assumption about how repeating readings are correlated, but their construction only tolerates sub-linear fraction of errors. Moreover, their construction of FE has to rely on a powerful tool named "digital locker". Up to now, digital locker can only be instantiated with a hash function modeled as random oracle or constructed from the non-standard strong vector DDH assumption. Following the line of constructing reusable FE from digital locker, Alamelou et al. [1] constructed a reusable FE for both the set difference metric and Hamming distance. Their construction tolerates linear fraction of errors but requires that noisy secrets distributions have enough entropy in each symbol of a large alphabet.

Recently, Wen et al.[26] proposed a reusable FE from the DDH assumption which can tolerate linear fraction of errors. But a strong requirement is imposed on the input distribution: any differences between two distinct inputs should not leak too much information of the source w.

As far as we know, the available works on reusable FE follow three lines according to the correlations among source readings w_i's. The first line considers arbitrary correlations among w_i's and has to rely on non-standard assumptions or random oracle. The second line imposes strong requirements on the source, i.e., any differences between two distinct inputs should not leak too much information of the source w_i. The third line considers δ_i $(= w_i - w)$ controlled by adversaries. See Figure 1. The related works are also summarized in Table 1.

Robust Fuzzy Extractor. Robust Fuzzy Extractor aims to address the second issue. *Robustness* of FE requires that any modification of P by an adversary will be detected. Boyen et al. [5] introduced the concept of robust FE, and proposed a general way of converting a FE to a robust one. In their approach, a hash function is employed and modeled as a random oracle. Dodis et al. [10] strengthened robustness to *post-application robustness*, which guarantees that the FE will detect any modification of P by adversary who also sees R. Later, robust FE was slightly improved in [18]. Nevertheless, it was shown in [13] that in the information theoretic setting, it is impossible to construct a robust FE if the entropy rate of W is less than half in the plain model. Cramer et al. [7] broke this barrier by building a robust FE in the Common Reference String (CRS) model. Recall that CRS can be hardwired or hardcoded into the system

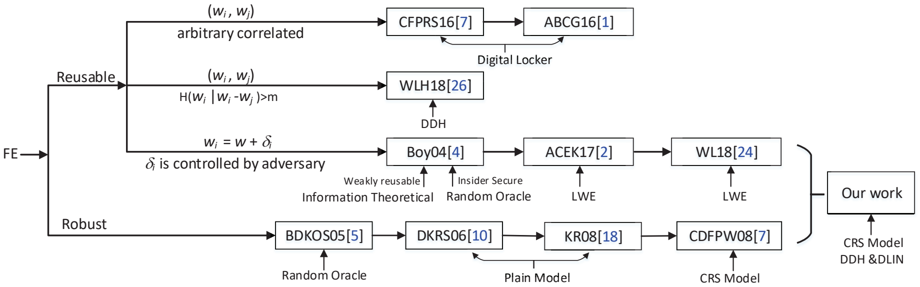

Fig. 1. Related works about reusable FE and robust FE. $\mathsf{H}(\mathsf{w_i}|\mathsf{w_i}-\mathsf{w_j})$ is the average min-entropy of $\mathsf{w_i}$ conditioned on $\mathsf{w_i}-\mathsf{w_j}$.

so that CRS can be observed but not modified by adversaries. See Figure 1 and Table 1 for related works of robust FE.

We stress that up to now there is no work ever considering robustness of reusable FE or reusability of robust FE in the standard model, since designing reusable FE or robust FE alone is already an uneasy task.

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 can correct linear fraction of errors. “–” represents the scheme is an information theoretical one.

FE Schemes	Robustness?	Reusabiliy?	Standard Assumption?	Linear Errors?
FMR13[14]	✗	✗	✔	✗
DRS04[12], Boy04[4]	✗	weak	–	✔
CFPRS16[6]	✗	✔	✗	✗
Boy04[4] ABCG16[1]	✗	✔	✗	✔
ACEK17[2]	✗	✔	✔	✗
BDKOS05[5]	✔	✗	✗	✔
DKRS06[10], KR08[18], CDFPW08[7]	✔	✗	–	✔
WL18[24],WLH18[26]	✗	✔	✔	✔
Ours	✔	✔	✔	✔

1.1 Our Contributions

We consider how to construct fuzzy extractors satisfying reusability and robustness simultaneously based on standard assumptions in the CRS model.

- We formalize *robustly reusable fuzzy extractor* (rrFE) whose security notions include both *reusability* and post-application *robustness* in the computational setting.

- We propose a general construction of rrFE from a Symmetric Key Encapsulation Mechanism (SKEM), a Secure Sketch (SS), an Extractor (Ext), and a Lossy Algebraic Filter (LAF) in the CRS model.
  - We characterize the required security notion of SKEM and the homomorphic properties of SS, Ext and LAF, which enable the construction of rrFE to achieve both reusability ad robustness.
  - SKEM is a primitive similar to Key Encapsulation Mechanism (KEM), but the encapsulation and decapsulation make use of the same secret key. We define Key-Shift (KS) security for SKEM, which says that the encapsulated key is pseudorandom, even if the adversary sees multiple encapsulations under shifted secret keys where the shifts are designated by the adversary. We present an instantiation of SKEM and prove its KS-security from the DDH assumption.
- We obtain the first rrFE tolerating linear fraction of errors based on standard assumptions by instantiating SKEM, LAF, SS and Ext. More precisely, SKEM is built from the DDH assumption and LAF by Hofheinz (EuroCrypt 2013) is based on the DLIN assumption.

Our construction is the first FE possessing both reusability and robustness. Meanwhile, our construction is able to tolerate a linear fraction of errors. However, we do not assume arbitrary correlations between different readings of w. Instead, we assume that the shifts between different readings are controlled by the adversary in the security model, just like [2]. Our work can be regarded as a step forward from the the third and fourth branches in Figure 1.

Table 1 compares our rrFE with the available reusable FE and robust FE.

1.2 Our Approach

Our work stems from the traditional *sketch-and-extract* paradigm [11] due to Dodis et al. First, we review the traditional *sketch-and-extract* paradigm [11]. Then we introduce a new primitive called Symmetric Key Encapsulation Mechanism (SKEM) and define for it a so-called *Key-Shift* security. We also recall the definition of Lossy Algebraic Filter (LAF) introduced by Hofheinz [15]. Equipped with SKEM and LAF, we show how to construct a *robustly reusable Fuzzy Extractor* (rrFE) from SS, Ext, SKEM and LAF. Finally, we describe the high level idea of why our construction of rrFE achieves both reusability and robustness.

The Sketch-and-Extract Paradigm. In [11], Dodis et al. proposed a paradigm of constructing FE from secure sketch and extractor.

Secure Sketch (SS) is used for removing noises from fuzzy inputs. An SS scheme consists of a pair of algorithms $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$. Algorithm $\mathsf{SS.Gen}$ on input w outputs a sketch s; algorithm $\mathsf{SS.Rec}$ on input s and w' recovers w as long as w and w' are close enough. For SS, it is required that W still has enough entropy conditioned on s.

An extractor Ext distills an almost uniform key R from the non-uniform random variable W of enough entropy, with the help of a random seed i_{ext}.

The sketch-and-extract construction of $\mathsf{FE} = (\mathsf{Gen}, \mathsf{Rep})$ [11] works as follows.

- $\mathsf{Gen}(w, i_{\mathsf{ext}})$: Set $P := (\mathsf{SS.Gen}(w), i_{\mathsf{ext}})$, $R := \mathsf{Ext}(w, i_{\mathsf{ext}})$. Output (P, R).
- $\mathsf{Rep}(w', P = (s, i_{\mathsf{ext}}))$: Recover $w := \mathsf{SS.Rec}(w', s)$ and output $R := \mathsf{Ext}(w, i_{\mathsf{ext}})$.

Symmetric Key Encapsulation Mechanism. For reusability, we introduce a technical tool called *symmetric key encapsulation mechanism* (SKEM). It is similar to Key Encapsulation Mechanism (KEM)[8], except that the encapsulation and decapsulation algorithms share the same secret key sk.

- Encapsulation algorithm $\mathsf{SKEM.Enc}$ takes as input the secret key sk, and outputs a ciphertext c and an encapsulated key $k \in \mathcal{K}$.
- Decapsulation algorithm $\mathsf{SKEM.Dec}$ recovers the key k, on input c and sk.

The requirement for SKEM is *key-shift* security. That is, $(c, k) \leftarrow \mathsf{SKEM.Enc}(sk)$ is computationally indistinguishable from (c, u), where u is uniformly chosen from $\mathcal{K}$, even if the adversary has an access to a key-shift encapsulation oracle $\mathsf{SKEM.Enc}(sk + \Delta_i)$, where Δ_i is chosen by the adversary adaptively.

Lossy Algebraic Filter. For robustness, we introduce a technical tool named lossy algebraic filter (LAF) by Hofheinz [15]. It is a family of functions indexed by a public key F_{pk} and a tag tag. A tag is lossy, injective or neither. A function from that family takes a vector $X = (X_i)_{i=1}^{\mathfrak{n}} \in \mathbb{Z}_p^{\mathfrak{n}}$ as input. If tag is an injective tag, then the function $\mathsf{LAF}_{F_{pk},\mathsf{tag}}(\cdot)$ is an injective function. If tag is lossy, then the function is *lossy* in the sense that the value only depends on a linear combination of $\sum_{i=1}^{\mathfrak{n}} u_i X_i \in \mathbb{Z}_p$ (instead of the whole X), where the coefficients $\{u_i\}_{i\in[\mathfrak{n}]}$ are independent of the lossy tag and depend only on the public key. In particular, evaluating the same input X under multiple lossy tags with respect to a common public key only reveals the same linear combination $\sum_{i=1}^{\mathfrak{n}} u_i X_i \in \mathbb{Z}_p$, thus leaking at most $\log p$ bits of information about X. It is required that there are many lossy tags and with a trapdoor one can efficiently sample a lossy tag. Additionally, LAF has two more properties named *evasiveness* and *indistinguishability*. Evasiveness demands that without the trapdoor, any PPT adversary can hardly find a new non-injective tag even given many lossy tags; indistinguishability demands that it is hard to distinguish lossy tags from random tags for all PPT adversaries.

Our Construction. Our rrFE stems from the basic "sketch-and-extract" FE [11], but an SKEM and an LAF are integrated to this basic FE to achieve reusability and robustness. The construction is shown in Fig. 2.

In our construction, the reading w of a source plays two roles, one is for extraction(reproduction) of R ($\widetilde{R}$), the other is for authentication (verification). We stress that $\mathsf{LAF}_{F_{pk},\mathsf{tag}}(w)$ can be regarded as a message authentication code (MAC)[4], where w is the authentication key, tag is the message, and the output of LAF is just the authenticator σ.

Below describes how the generation algorithm of our rrFE works.

[4] The traditional MAC does not apply in the scenario of robust fuzzy extractor: the adversary can arbitrarily modify the public helper string P, so the key of the MAC is modified accordingly. As a result, the message and the authentication key are not independent anymore.

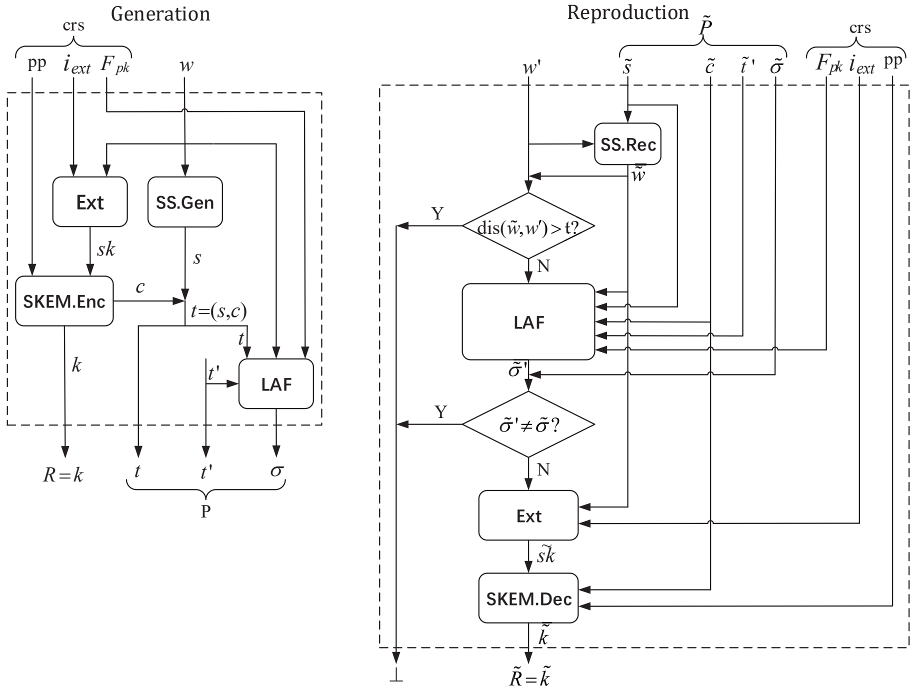

Fig. 2. Construction of robustly reusable fuzzy extractor.

- The common reference string crs consists of the public parameter pp of SKEM, the random seed i_{ext} of Ext, and the public key F_{pk} of LAF.
- The reading w of a source is fed not only to SS and Ext, but also to LAF. This results in a sketch s from $\mathsf{SS.Gen}$, a secret key sk from Ext, and an authenticator σ from LAF.
- We do not take the output sk of Ext as the final extracted key. Instead, the output sk of Ext serves as the secret key of $\mathsf{SKEM.Enc}$, which in turn outputs a ciphertext c and an encapsulated key k. This encapsulated key k is served as the final extracted key $R := k$.
- The evaluation of LAF on w under tag $\mathsf{tag} = (s, c, t')$ results in an authenticator σ, where t' is randomly chosen. The public helper string is set as $P := (s, c, t', \sigma)$.

Given the public helper string $\widetilde{P} = (\widetilde{s}, \widetilde{c}, \widetilde{t'}, \widetilde{\sigma})$ and a reading w', the reproduction algorithm of our rrFE will return the reproduced key $\widetilde{R} := \mathsf{SKEM.Dec}(\mathsf{Ext}(\widetilde{w}, i_{\mathsf{ext}}), \widetilde{c})$ only if the distance of $\widetilde{w} := \mathsf{SS.Rec}(w', \widetilde{s})$ and w' is no more than a predetermined threshold t and the computed authenticator $\widetilde{\sigma}' := \mathsf{LAF}_{F_{pk},(\widetilde{s},\widetilde{c},\widetilde{t'})}(\widetilde{w})$ is identical to the authenticator $\widetilde{\sigma}$ contained in $\widetilde{P}$.

Reusability. Reusability says that the extracted key R is pseudorandom even if the PPT adversary knows $P = (s, c, t, \sigma)$ and can adaptively asks the generation oracle with shift δ_i to get multiple $\{P_i = (s_i, c_i, t'_i, \sigma_i), R_i\}_{i\in[\rho]}$ where $(P_i, R_i) \leftarrow \mathsf{Gen}(w + \delta_i)$.

To achieve reusability, we require that the underlying building blocks SS, Ext and LAF are homomorphic and SKEM is key-shift secure. Recall that i_{ext} and F_{pk} are parts of crs so they are independent of each other and distributed as designed. The high level idea of proving reusability is as follows.

1. By the homomorphic property of SS, Ext and LAF, we have
   - $s_i := \mathsf{SS.Gen}(w_i) = \mathsf{SS.Gen}(w+\delta_i) = \mathsf{SS.Gen}(w)+\mathsf{SS.Gen}(\delta_i)= s + \mathsf{SS.Gen}(\delta_i)$;
   - $sk_i = \mathsf{Ext}(w_i, i_{\mathsf{ext}}) = \mathsf{Ext}(w+\delta_i, i_{\mathsf{ext}}) = \mathsf{Ext}(w, i_{\mathsf{ext}})+\mathsf{Ext}(\delta_i, i_{\mathsf{ext}})= sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$;
   - $\sigma_i := \mathsf{LAF}_{F_{pk},\mathsf{tag}_i}(w+\delta_i) = \mathsf{LAF}_{F_{pk},\mathsf{tag}_i}(w)+\mathsf{LAF}_{F_{pk},\mathsf{tag}_i}(\delta_i)= \sigma + \mathsf{LAF}_{F_{pk},\mathsf{tag}_i}(\delta_i)$.

   Observe that the knowledge of $\mathsf{SS.Gen}(w), \mathsf{Ext}(w)$ and $\{\mathsf{LAF}_{F_{pk},\mathsf{tag}_i}(w)\}_{i\in[\rho]}$ suffices for the challenger to simulate the whole view of the adversary in the reusability experiment.
2. By the indistinguishability property of LAF, $\{\mathsf{tag}_i\}_{i\in[\rho]}$ can be replaced with lossy tags. Now the challenger can use $\mathsf{SS.Gen}(w), \mathsf{Ext}(w)$ and $\mathcal{S} := \{\mathsf{LAF}_{F_{pk},\mathsf{tag}}(w)$ for all lossy tags$\}$ to simulate the view of the adversary.
3. By the lossiness of LAF, the information of W leaked by $\mathcal{S}$ is at most $\log p$ bits. By the security of SS, the information of W leaked by $\mathsf{SS.Gen}(w)$ is also bounded. Meanwhile, $\mathsf{SS.Gen}(w)$ and set $\mathcal{S}$ are independent of i_{ext} due to the independence between (W, F_{pk}) and i_{ext} (note that the lossy tag space is determined by F_{pk}). Consequently, $sk := \mathsf{Ext}(w, i_{ext})$ is almost uniform conditioned on $\mathsf{SS.Gen}(w)$ and $\mathcal{S}$.
4. Observe that $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(sk_i)$ can be regarded as encapsulations under shifted key $sk_i := sk + \mathsf{Ext}(\delta_i)$. With a uniform sk (conditioned on $\mathsf{SS.Gen}(w)$ and $\mathcal{S}$), the KS-security of SKEM makes sure that $R := k$ is pseudorandom given P and $\{P_i = (s_i, c_i, t'_i, \sigma_i), R_i = k_i\}_{i\in[\rho]}$, where $(c, k) \leftarrow \mathsf{SKEM.Enc}(sk)$.

Robustness. Robustness states that even if the PPT adversary can adaptively asks the generation oracle with shift δ_i to get $(P_i, R_i) \leftarrow \mathsf{Gen}(w + \delta_i)$, it is still hard to forge a fresh valid $\widetilde{P}$.

Following 1, 2 and 3 of the above analysis for reusability, the view of adversary in the robustness experiment can be simulated with the knowledge of $\mathsf{SS.Gen}(w)$ and $\mathcal{S}$. Note that the $\mathsf{SS.Gen}(w)$ and set $\mathcal{S}$ only leak bounded information of W. Consequently, even if the adversary sees $\{P_i, R_i\}_{i\in[\rho]}$, there is still enough entropy left in W. By the *evasiveness* of LAF, the forged tag $\widetilde{\mathsf{tag}} = (\widetilde{s}, \widetilde{c}, \widetilde{t'})$ contained in $\widetilde{P} = (\widetilde{s}, \widetilde{c}, \widetilde{t'}, \widetilde{\sigma})$ must be injective, hence $\mathsf{LAF}_{F_{pk},\widetilde{\mathsf{tag}}}(\cdot)$ is an injective function. Consequently, the entropy of W is intactly transferred to $\widetilde{\sigma}' := \mathsf{LAF}_{F_{pk},\widetilde{\mathsf{tag}}}(\widetilde{w})$ and the forged authenticator $\widetilde{\sigma}$ hits the value of $\widetilde{\sigma}'$ with negligible probability.

2 Preliminaries

Let λ be the security parameter. We write PPT short for probabilistic polynomial-time. Let $[\rho]$ denote set $\{1, 2 \cdots, \rho\}$. Let $\lceil x\rceil$ denote the smallest integer that is not smaller than x. If X is a distribution, $x \leftarrow X$ denotes sampling x according to distribution X; if X is a set, $x \leftarrow_{\$} X$ denotes choosing x from X uniformly. For a set X, let $|X|$ denote the size of X. Let $\overbrace{xxx}^{y}$ and $\underbrace{xxx}_{y}$ denote $y := xxx$. For a primitive XX and a security notion YY, by $\mathsf{Exp}^{\mathrm{YY}}_{\mathrm{XX},\mathcal{A}}(\cdot) \Rightarrow 1$, we mean that the security experiment outputs 1 after interacting with an adversary $\mathcal{A}$; by $\mathsf{Adv}^{\mathrm{YY}}_{\mathrm{XX},\mathcal{A}}(1^{\lambda})$, we denote the advantage of a PPT adversary $\mathcal{A}$ and define $\mathsf{Adv}^{\mathrm{YY}}_{\mathrm{XX}}(1^{\lambda}) := \max_{\mathrm{PPT}\mathcal{A}} \mathsf{Adv}^{\mathrm{YY}}_{\mathrm{XX},\mathcal{A}}(1^{\lambda})$. Our security proof will proceed by a sequence of games. By $a \overset{\mathsf{G}}{=} b$ we mean that a equals b or is computed as b in game G. By $\mathsf{G}^{\mathcal{A}} \Rightarrow b$, we mean that game G outputs b after interacting with $\mathcal{A}$.

2.1 Metric Spaces

A metric space is a set $\mathcal{M}$ with a distance function $\mathsf{dis}\colon \mathcal{M} \times \mathcal{M} \mapsto [0, \infty)$. We usually consider multi-dimensional metric spaces of form $\mathcal{M} = \mathcal{F}^n$ for some alphabet $\mathcal{F}$ (usually a finite filed $\mathbb{F}_p$) equipped with the Hamming distance. For any two element $w, w' \in \mathcal{M}$, the Hamming distance $\mathsf{dis}(w, w')$ is the number of coordinates in which they differ. For an element $w \in \mathcal{M}$, let $\mathsf{dis}(w) := \mathsf{dis}(w, 0)$.

2.2 Min-Entropy, Statistical Distance and Extractor

Definition 1 (Min-Entropy). *For a random variable X, the* min-entropy *of X is defined by $H_\infty(X) = -\log(\max_x \Pr[X = x])$. The* average min-entropy *of X given Y is defined by $\widetilde{H}_\infty(X|Y) = -\log[\mathbb{E}_{y \leftarrow Y}(\max_x \Pr[X = x|Y = y])]$.*

Obviously, for a deterministic function f and a randomized function g with the random coins R independent of X, we have that

$$\widetilde{H}_\infty(X \mid Y, f(Y)) = \widetilde{H}_\infty(X \mid Y). \tag{1}$$

$$\widetilde{H}_\infty(X \mid Y, g(Y, R)) = \widetilde{H}_\infty(X \mid Y). \tag{2}$$

Lemma 1. *[11] If Y takes at most 2^λ possible values, then $\widetilde{H}_\infty(X \mid Y) \geq \widetilde{H}_\infty(X) - \lambda$.*

Definition 2 (Statistical Distance). *For two random variables X and Y over a set $\mathcal{M}$, the* statistical distance *of X and Y is given by $\mathbf{SD}(X, Y) = \frac{1}{2}\sum_{w \in \mathcal{M}} |\Pr[X = w] - \Pr[Y = w]|$. If $\mathbf{SD}(X, Y) \leq \varepsilon$, X and Y are called* ε-statistically indistinguishable, *denoted by $X \overset{\varepsilon}{\approx} Y$.*

Lemma 2. *[22] Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be finite sets, X and Y be random variables over $\mathcal{M}_1$, and $f\colon \mathcal{M}_1 \mapsto \mathcal{M}_2$ be a function. Then $\mathbf{SD}(f(X), f(Y)) \leq \mathbf{SD}(X, Y)$.*

Definition 3 (Average-Case Strong Extractor [11]). *We call a function $\mathsf{Ext}\colon \mathcal{M} \times \mathcal{I} \mapsto \mathcal{SK}$ an average-case $(\mathcal{M}, m, \mathcal{SK}, \varepsilon)$-strong extractor with seed space $\mathcal{I}$, if for all pairs of random variables (X, Y) such that $X \in \mathcal{M}$ and $\widetilde{H}_\infty(X \mid Y) \geq m$, we have*

$$(\mathsf{Ext}(X, I), I, Y) \overset{\varepsilon}{\approx} (U, I, Y), \tag{3}$$

where I and U are uniformly distributed over $\mathcal{I}$ and $\mathcal{SK}$, respectively.

2.3 Secure Sketch

Definition 4 (Secure Sketch [11]). *An* $(m, \hat{m}, \mathfrak{t})$-secure sketch (SS) $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ *for metric space* $\mathcal{M}$ *with distance function* dis*, consists of a pair of PPT algorithms and satisfies correctness and security.*

- $\mathsf{SS.Gen}$ *on input* $w \in \mathcal{M}$*, outputs a sketch* s*.*
- $\mathsf{SS.Rec}$ *takes as input* $w' \in \mathcal{M}$ *and a sketch* s*, and outputs* $\tilde{w}$*.*

Correctness. $\forall w \in \mathcal{M}$*, if* $\mathsf{dis}(w, w') \leq \mathfrak{t}$*, then* $\mathsf{SS.Rec}(w', \mathsf{SS.Gen}(w)) = w$*.*
Security. *For any random variable* W *over* $\mathcal{M}$ *with min-entropy* m*, we have* $\widetilde{H}_\infty(W \mid \mathsf{SS.Gen}(W)) \geq \hat{m}$*.*

Lemma 3. *[5] Let* $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ *be an* $(m, \hat{m}, t)$*-SS for* $\mathcal{M}$*, if* W_0, W_1 *are two random variables over* $\mathcal{M}$ *satisfying* $\mathsf{dis}(W_0, W_1) \leq \mathfrak{t}$*, then for any variable* Y*, we have* $\widetilde{H}_\infty(W_1 \mid (\mathsf{SS.Gen}(W_0), Y)) \geq \widetilde{H}_\infty(W_0 \mid (\mathsf{SS.Gen}(W_0), Y))$*.*

2.4 Lossy Algebraic Filter

Our construction of robustly reusable fuzzy extractor relies on a technical tool, named lossy algebraic filter which is proposed by Hofheinz [15].

Definition 5 (Lossy Algebraic Filter). *An* $(l_{\mathsf{LAF}}, \mathfrak{n})$*-lossy algebraic filter* $\mathsf{LAF} = (\mathsf{FGen}, \mathsf{FEval}, \mathsf{FTag})$ *consists of three PPT algorithms.*

- **Key generation**. $\mathsf{FGen}(1^\lambda)$ outputs a public key F_{pk} together with a trapdoor F_{td}, i.e., $(F_{pk}, F_{td}) \leftarrow \mathsf{FGen}(1^\lambda)$. The public key F_{pk} contains an l_{LAF}-bit prime p and defines a tag space $\mathcal{T}_{\mathsf{tag}} = \{0,1\}^* \times \mathcal{T}'$, a lossy tag space $\mathcal{T}_{lossy} \subseteq \mathcal{T}_{\mathsf{tag}}$ and an injective tag space $\mathcal{T}_{inj} \subseteq \mathcal{T}_{\mathsf{tag}}$. A tag $\mathsf{tag} = (t, t') \in \mathcal{T}_{\mathsf{tag}}$ consists of a core tag $t' \in \mathcal{T}'$ and an auxiliary tag $t \in \{0,1\}^*$. F_{td} is a trapdoor that allows of sampling lossy tags.
- **Evaluation**. FEval takes as input the public key F_{pk}, a tag $\mathsf{tag} = (t, t')$, and $X = (X_i)_{i=1}^{\mathfrak{n}} \in \mathbb{Z}_p^{\mathfrak{n}}$, and outputs $\mathsf{LAF}_{F_{pk},\mathsf{tag}}(X)$, i.e., $\mathsf{LAF}_{F_{pk},\mathsf{tag}}(X) = \mathsf{FEval}(F_{pk}, \mathsf{tag}, X)$.
- **Lossy tag generation**. FTag takes as input the trapdoor F_{td} and an auxiliary tag t, and returns a core tag t', i.e., $t' \leftarrow \mathsf{FTag}(F_{td}, t)$, such that $\mathsf{tag} = (t, t')$ is a lossy tag.

We require the following:

- **Lossiness**. If $\mathsf{tag} \in \mathcal{T}_{inj}$, then the function $\mathsf{LAF}_{F_{pk},\mathsf{tag}}(\cdot)$ is injective. If $\mathsf{tag} \in \mathcal{T}_{lossy}$, then $\mathsf{LAF}_{F_{pk},\mathsf{tag}}(X)$ depends only on $\sum_{i=1}^{\mathfrak{n}} u_i X_i \mod p$ for $u_i \in \mathbb{Z}_p$ that only depends on F_{pk}.
- **Indistinguishability**. For all PPT adversaries, it is hard to distinguish lossy tags from random tags. Formally,

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF},\mathcal{A}}(1^\lambda) := \left| \Pr\left[\mathcal{A}(1^\lambda, F_{pk})^{\mathsf{FTag}(F_{td},\cdot)} = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, F_{pk})^{\mathcal{O}_{\mathcal{T}'}(\cdot)} = 1\right] \right|$$

is negligible for all PPT adversary $\mathcal{A}$, where $(F_{pk}, F_{td}) \leftarrow \mathsf{FTag}(1^\lambda)$ and $\mathcal{O}_{\mathcal{T}'}(\cdot)$ is the oracle that ignores its input and samples a random core tag t'.

- **Evasiveness**. For all PPT adversaries, without the trapdoor, non-injective tags are hard to find, even given multiple lossy tags. More precisely,

$$\mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF},\mathcal{A}}(1^\lambda) := \Pr\left[\mathsf{tag} \notin \mathcal{T}_{inj} \mid \mathsf{tag} \leftarrow \mathcal{A}(1^\lambda, F_{pk})^{\mathsf{FTag}(F_{td},\cdot)}\right]$$

  is negligible for all PPT admissible adversary $\mathcal{A}$ where $(F_{pk}, F_{td}) \leftarrow \mathsf{FGen}(1^\lambda)$. We call $\mathcal{A}$ is admissible if $\mathcal{A}$ never outputs a tag obtained from its oracle.

Remark 1. If $\mathsf{tag} = (t, t')$, we use $\mathsf{FEval}(F_{pk}, t, t', X)$ to denote $\mathsf{FEval}(F_{pk}, \mathsf{tag}, X)$.

Remark 2. Let us consider multiple, say $\mathfrak{m}$, evaluations of LAF of the same $X = (X_1, X_2, \ldots, X_\mathfrak{n})$ under a fixed public key F_{pk} but different tags (t_j, t'_j). According to the lossiness property of LAF, each evaluation of $\mathsf{FEval}(F_{pk}, t_j, t'_j, X)$ is completely determined by $\sum_{i=1}^{\mathfrak{n}} u_i X_i$ and (t_j, t'_j), so there exists a function f such that $\mathsf{FEval}(F_{pk}, t_j, t'_j, X) = f\left(\sum_{i=1}^{\mathfrak{n}} u_i X_i, (t_j, t'_j)\right)$. Suppose that F_{pk} is independent of X. As long as tags $\{(t_j, t'_j)\}_{j\in[\mathfrak{m}]}$ are independent of X or are (randomized) functions of $\sum_{i=1}^{\mathfrak{n}} u_i X_i$, we have

$$\widetilde{H}_\infty\left(X \,\middle|\, \{\mathsf{FEval}(F_{pk}, t_j, t'_j, X)\}_{j\in[\mathfrak{m}]}\right) = \widetilde{H}_\infty\left(X \,\middle|\, \left\{f\left(\sum_{i=1}^{\mathfrak{n}} u_i X_i, (t_j, t'_j)\right)\right\}_{j\in[\mathfrak{m}]}\right)$$
$$\geq \widetilde{H}_\infty\left(X \,\middle|\, \sum_{i=1}^{\mathfrak{n}} u_i X_i\right) \geq \widetilde{H}_\infty(X) - \log p, \tag{4}$$

where the last but one step is due to Eq. (2) and the last step is by Lemma 1.

2.5 Homomorphic Properties

We assume that the domains and codomains of Ext, SS and LAF are groups with operation "+" (we abuse "+" for different group operations for simplicity). Now we characterize homomorphic properties of Ext, SS and LAF.

Definition 6 (Homomorphic Average-Case Strong Extractor). *An average-case $(\mathcal{M}, m, \mathcal{SK}, \varepsilon)$-strong extractor* $\mathsf{Ext} : \mathcal{M} \times \mathcal{I} \to \mathcal{SK}$ *is homomorphic if for all $w_1, w_2 \in \mathcal{M}$, all $i_{\mathsf{ext}} \in \mathcal{I}$, we have* $\mathsf{Ext}(w_1 + w_2, i_{\mathsf{ext}}) = \mathsf{Ext}(w_1, i_{\mathsf{ext}}) + \mathsf{Ext}(w_2, i_{\mathsf{ext}})$.

It was shown in [11], universal hash functions are average-case strong extractors. In particular, $\mathsf{Ext}(x, i) \colon \mathbb{Z}_q^{l+1} \times \mathbb{Z}_q^l \to \mathbb{Z}_q$ defined by

$$\mathsf{Ext}(x, i) := x_0 + i_1 x_1 + \cdots + i_l x_l \tag{5}$$

is an average-case strong $(\mathbb{Z}_q^{l+1}, m, \mathbb{Z}_q, \varepsilon)$-extractor with $\log q \leq m + 2\log\varepsilon$, as shown in [22]. Obviously, it is homomorphic.

Definition 7 (Homomorphic Secure Sketch). *A* secure sketch *is homomorphic if for all $w_1, w_2 \in \mathcal{M}$,* $\mathsf{SS.Gen}(w_1 + w_2) = \mathsf{SS.Gen}(w_1) + \mathsf{SS.Gen}(w_2)$.

The syndrome-based secure sketch [12] is homomorphic (see the full version [25]).

Definition 8 (Homomorphic Lossy Algebraic Filter). *We call an $(l_{\mathsf{LAF}}, \mathfrak{n})$-LAF with domain $\mathbb{Z}_p^{\mathfrak{n}}$ is homomorphic if for all $(F_{pk}, F_{td}) \leftarrow \mathsf{FGen}(1^\lambda)$, all* $\mathsf{tag} \in \mathcal{T}_{\mathsf{tag}}$ *and all $w_1, w_2 \in \mathbb{Z}_p^{\mathfrak{n}}$, the following holds* $\mathsf{FEval}(F_{pk}, \mathsf{tag}, w_1 + w_2) = \mathsf{FEval}(F_{pk}, \mathsf{tag}, w_1) + \mathsf{FEval}(F_{pk}, \mathsf{tag}, w_2)$.

The LAF constructed from the DLIN assumption in [15] is homomorphic. See the full version [25] for the specific construction of homomorphic LAF.

2.6 Decisional Diffie-Hellman Assumption

Definition 9 (Decisional Diffie-Hellman Assumption). *The decisional Diffie-Hellman assumption holds w.r.t. a group generation algorithm $\mathcal{IG}$, if*

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{IG},\mathcal{A}}(1^\lambda) := |\Pr[\mathcal{A}((\mathbb{G},q,g),g^x,g^y,g^z)=1]-\Pr[\mathcal{A}((\mathbb{G},q,g),g^x,g^y,g^{xy})=1]|$$

is negligible for all PPT adversary $\mathcal{A}$, where $(\mathbb{G},q,g) \leftarrow \mathcal{IG}(1^\lambda)$, $\mathbb{G}$ is a cyclic group of order q with generator g and $x,y,z \leftarrow_\$ \mathbb{Z}_q$.

3 Symmetric Key Encapsulate Mechanism

3.1 Definition of SKEM

In this section, we propose a new primitive called *symmetric key encapsulate mechanism* (SKEM). It is one of the core technical tools in our rrFE.

Definition 10 (Symmetric Key Encapsulate Mechanism). *A symmetric key encapsulate mechanism* $\mathsf{SKEM} = (\mathsf{SKEM.Init}, \mathsf{SKEM.Enc}, \mathsf{SKEM.Dec})$ *consists of a triple of PPT algorithms.*

- $\mathsf{SKEM.Init}$ *takes as input the security parameter 1^λ and outputs public parameter* pp *which implicitly defines the secret key space $\mathcal{SK}$, encapsulated key space $\mathcal{K}$ and ciphertext space, i.e.,* $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$.
- $\mathsf{SKEM.Enc}$ *takes as input* pp *and the secret key sk, and outputs a ciphertext c and an encapsulated key $k \in \mathcal{K}$, i.e., $(c,k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$.*
- $\mathsf{SKEM.Dec}$ *takes as input* pp*, the secret key sk and a ciphertext c, and outputs $k \in \mathcal{K}$, i.e., $k \leftarrow \mathsf{SKEM.Dec}(\mathsf{pp}, sk, c)$.*

The correctness of SKEM is that for all $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$, $sk \in \mathcal{SK}$, $(c,k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, $k' \leftarrow \mathsf{SKEM.Dec}(\mathsf{pp}, sk, c)$, we have $k' = k$.

We require pseudorandomness of the encapsulated key under key-shift attack. Roughly speaking, the encapsulated key is pseudorandom even if the adversary observes multiple encapsulations under shifted secret key where the shift Δ_i is designated by the adversary adaptively. The formal definition is given below.

Definition 11 (KS-Security of SKEM). *A* SKEM $\mathsf{SKEM} = (\mathsf{SKEM.Init}, \mathsf{SKEM.Enc}, \mathsf{SKEM.Dec})$ *is Key-Shift* (KS) *secure if for all PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(1^\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(1) \Rightarrow 1] - \Pr[\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(0) \Rightarrow 1]|$$

is negligible. Here $\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(\beta)$, $\beta \in \{0,1\}$, is an experiment played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.

$\underline{\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(\beta):}$

- *$\mathcal{C}$ invokes $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$, samples $sk \leftarrow_\$ \mathcal{SK}$ and returns pp to $\mathcal{A}$.*
- *Challenge: Challenger $\mathcal{C}$ invokes $(c,k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$. If $\beta = 0$, it resets k with $k \leftarrow_\$ \mathcal{K}$. Finally it returns (c,k) to $\mathcal{A}$.*
- *During the whole experiment, $\mathcal{A}$ may adaptively make encapsulation oracle queries of the following form:*
  - *$\mathcal{A}$ submits a shift $\Delta_i \in \mathcal{SK}$ to challenger $\mathcal{C}$.*
  - *$\mathcal{C}$ invokes $(c_i,k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk + \Delta_i)$, and returns (c_i,k_i) to $\mathcal{A}$.*
- *As long as $\mathcal{A}$ outputs a guessing bit β', the experiment outputs β'.*

3.2 Construction of Symmetric Key Encapsulate Mechanism

We instantiate a KS-secure SKEM from the DDH assumption, and the construction is given in Fig. 3.

```
$\underline{\mathsf{SKEM.Init}(1^\lambda)}$:
    $(\mathbb{G}, q, g) \leftarrow \mathcal{IG}(1^\lambda)$.
    $\mathsf{pp} := (\mathbb{G}, q, g)$.
    $\mathcal{SK} := \mathbb{Z}_q$.
    $\mathcal{K} := \mathbb{G}$.
    Return $\mathsf{pp}$.
```

```
$\underline{\mathsf{SKEM.Enc}(\mathsf{pp}, sk)}$:   // $sk \in \mathcal{SK}$
    $r \leftarrow_{\$} \mathbb{Z}_q$.
    $c = g^r$.
    $k = c^{sk}$.
    Return $(c, k)$.
```

```
$\underline{\mathsf{SKEM.Dec}(\mathsf{pp}, sk, c)}$:
    $k = c^{sk}$.
    Return $k$.
```

Fig. 3. Construction of SKEM with KS-security from the DDH assumption.

Theorem 1. *If the DDH assumption holds with respect to $\mathcal{IG}$, then* SKEM *constructed in Fig. 3 is* KS*-secure. More precisely, for any PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{IG}}(1^\lambda).$$

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ who has advantage ϵ in the key-shift attack of SKEM in Fig. 3, then we can construct a PPT algorithm $\mathcal{B}$ with the same advantage ϵ in solving the DDH problem.

Given $(\mathbb{G}, q, g, g^x, g^y, g^d)$, where x, y are uniformly and independently chosen from $\mathbb{Z}_q$, algorithm $\mathcal{B}$ simulates an environment for $\mathcal{A}$ as follows.

- Algorithm $\mathcal{B}$ returns $\mathsf{pp} = (\mathbb{G}, q, g)$ to $\mathcal{A}$ and implicitly sets $sk := x$.
- Algorithm $\mathcal{B}$ returns (g^y, g^d) to $\mathcal{A}$.
- When adversary $\mathcal{A}$ makes an encapsulation query with $\Delta_i \in \mathbb{Z}_p$, algorithm $\mathcal{B}$ uniformly chooses $y_i \leftarrow \mathbb{Z}_q$ and sets $c_i := g^{y_i}$, $k_i := (g^x g^{\Delta_i})^{y_i}$ and returns (c_i, k_i) to $\mathcal{A}$.
- When adversary $\mathcal{A}$ returns a bit β', algorithm $\mathcal{B}$ returns β' to its own challenger.

Obviously, $\mathcal{B}$ simulates answers to the encapsulation queries for $\mathcal{A}$ perfectly. For the challenge,

– If $d = xy$, then $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(1)$ for $\mathcal{A}$.

– If $d = z$, where $z \leftarrow_{\$} \mathbb{Z}_q$, then $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(0)$ for $\mathcal{A}$.

Consequently,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{IG},\mathcal{B}}(1^\lambda) &= \Pr[\mathcal{B}((\mathbb{G}, q, g), g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{B}((\mathbb{G}, q, g), g^x, g^y, g^z) = 1] \\ &= |\Pr[\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(1) \Rightarrow 1] - \Pr[\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(0) \Rightarrow 1]| = \mathsf{Adv}^{\mathsf{ks}}_{\mathsf{SKEM},\mathcal{A}}(1^\lambda).\end{aligned}$$

This completes the proof of Theorem 1. ▌

4 Robustly Reusable Fuzzy Extractor

In this section, we define robustly reusable fuzzy extractor (rrFE) and present a construction of rrFE in the CRS model.

4.1 Definition of Robustly Reusable Fuzzy Extractor

First, we recall the definition of fuzzy extractor presented in [7].

Definition 12 (Fuzzy Extractor). *An* $(\mathcal{M}, m, \mathcal{R}, \mathsf{t}, \varepsilon)$-fuzzy extractor FE *for metric space* $\mathcal{M}$ *consists of three PPT algorithms* $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$*,*

- Init *on input security parameter* 1^λ *outputs common reference string* crs*, i.e.,* $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$*.*
- Gen *on input the common reference string* crs *and* $w \in \mathcal{M}$*, outputs a public helper string* P *and an extracted string* $R \in \mathcal{R}$*, i.e.,* $(P, R) \leftarrow \mathsf{Gen}(\mathsf{crs}, w)$*.*
- Rep *takes as input the common reference string* crs*, public helper string* P *and* $w' \in \mathcal{M}$*, and outputs an extracted string* R *or* $\perp$*, i.e.,* $R/\perp \leftarrow \mathsf{Rep}(\mathsf{crs}, P, w')$*.*

It satisfies the following properties.

Correctness. *If* $\mathsf{dis}(w, w') \leq \mathsf{t}$*, then for any* $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$*,* $(P, R) \leftarrow \mathsf{Gen}(\mathsf{crs}, w)$ *and* $R' \leftarrow \mathsf{Rep}(\mathsf{crs}, P, w')$*, it holds that* $R' = R$*.*

Privacy. *For any distribution* W *over metric space* $\mathcal{M}$ *with* $H_\infty(W) \geq m$*, any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{FE},\mathcal{A}}(1^\lambda) := |\Pr[\mathcal{A}(\mathsf{crs}, P, R) = 1] - \Pr[\mathcal{A}(\mathsf{crs}, P, U) = 1]| \leq \varepsilon,$$

where $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$*,* $(P, R) \leftarrow \mathsf{Gen}(\mathsf{crs}, W)$ *and* $U \leftarrow_{\$} \mathcal{R}$*.*

A fuzzy extractor is reusable if its privacy is retained even if the same noisy source is reused multiple times. We follow the definition of reusability of fuzzy extractor from [2] (which is called "strong reusability" in [2]).

Definition 13 (Reusable Fuzzy Extractor). *A fuzzy extractor* $\mathsf{rFE} = (\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ *is an* $(\mathcal{M}, m, \mathcal{R}, \mathsf{t}, \varepsilon_1)$*-reusable fuzzy extractor if it is a fuzzy extractor with* ε_1-reusability*. An* $(\mathcal{M}, m, \mathcal{R}, \mathsf{t}, \varepsilon_1)$*-fuzzy extractor is* ε_1-reusable*, if for any distribution* W *over metric space* $\mathcal{M}$ *with* $H_\infty(W) \geq m$*, for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(1^\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(1) \Rightarrow 1] - \Pr[\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(0) \Rightarrow 1]| \leq \varepsilon_1,$$

where $\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(\beta)$*,* $\beta \in \{0,1\}$*, describes the reusability experiment played between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*.*

$\underline{\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(\beta):}$ $\quad // \ \beta \in \{0,1\}$

1. *Challenger* $\mathcal{C}$ *invokes* $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$ *and returns* crs *to* $\mathcal{A}$*.*
2. $\mathcal{C}$ *samples* $w \leftarrow W$ *and invokes* $(P, R) \leftarrow \mathsf{Gen}(\mathsf{crs}, w)$*. If* $\beta = 1$*, return* (P, R) *to* $\mathcal{A}$*; otherwise, it chooses* $U \leftarrow_{\$} \mathcal{R}$ *and returns* (P, U) *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *may adaptively make queries of the following form:*
   - $\mathcal{A}$ *submits a shift* $\delta_i \in \mathcal{M}$ *satisfying* $\mathsf{dis}(\delta_i) \leq \mathsf{t}$ *to* $\mathcal{C}$*.*
   - $\mathcal{C}$ *invokes* $(P_i, R_i) \leftarrow \mathsf{Gen}(\mathsf{crs}, w + \delta_i)$*, and returns* (P_i, R_i) *to* $\mathcal{A}$*.*
4. *As long as* $\mathcal{A}$ *outputs a guessing bit* β'*, the experiment outputs* β'*.*

Robust fuzzy extractor guarantees that any modification of the public helper string by a PPT adversary will be detected. Now, combining the definition of reusability in [2] and robustness of fuzzy extractor in [7], we give the definition of robustly reusable fuzzy extractor.

Definition 14 (Robustness of Reusable Fuzzy Extractor). *Let* $\mathsf{rrFE} = (\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ *be an* $(\mathcal{M}, m, \mathcal{R}, \mathsf{t}, \varepsilon_1)$*-reusable fuzzy extractor. We say* rrFE *is* ε_2-robust *if for any distribution* W *over metric space* $\mathcal{M}$ *with* $H_\infty(W) \geq m$*, for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{rrFE},\mathcal{A}}(1^\lambda) := \Pr[\mathsf{Exp}^{\mathsf{rob}}_{\mathsf{rrFE},\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \varepsilon_2,$$

where $\mathsf{Exp}^{\mathsf{rob}}_{\mathsf{rrFE},\mathcal{A}}(1^\lambda)$ *describes the robustness experiment played between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*.*

$\underline{\mathsf{Exp}^{\mathsf{rob}}_{\mathsf{rrFE},\mathcal{A}}(1^\lambda):}$

1. *Challenger* $\mathcal{C}$ *invokes* $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$*, and returns* crs *to* $\mathcal{A}$*.*
2. $\mathcal{C}$ *samples* $w \leftarrow W$*, invokes* $(P, R) \leftarrow \mathsf{Gen}(\mathsf{crs}, w)$ *and returns* (P, R) *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *may adaptively make queries of the following form:*
   - $\mathcal{A}$ *submits a shift* $\delta_i \in \mathcal{M}$ *satisfying* $\mathsf{dis}(\delta_i) \leq \mathsf{t}$ *to challenger* $\mathcal{C}$*.*
   - $\mathcal{C}$ *invokes* $(P_i, R_i) \leftarrow \mathsf{Gen}(\mathsf{crs}, w + \delta_i)$*, and returns* (P_i, R_i) *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *submits its forgery* $(\widetilde{P}, \widetilde{\delta})$ *to* $\mathcal{C}$*.* $\mathcal{A}$ *wins if* $\mathsf{dis}(\widetilde{\delta}) \leq \mathsf{t}$*,* $\widetilde{P}$ *is fresh (i.e.,* $\widetilde{P}$ *is different from* P *and those* P_i*) and* $\mathsf{Rep}(\mathsf{crs}, \widetilde{P}, w + \widetilde{\delta}) \neq \perp$*. The experiment outputs* 1 *if* $\mathcal{A}$ *wins and* 0 *otherwise.*

Definition 15 (Robustly Reusable Fuzzy Extractor). *An* $(\mathcal{M}, m, \mathcal{R}, \mathsf{t}, \varepsilon_1, \varepsilon_2)$*-robustly reusable fuzzy extractor* (rrFE) *is an* $(\mathcal{M}, m, \mathcal{R}, \mathsf{t}, \varepsilon_1)$*-reusable fuzzy extractor with* ε_2-robustness*.*

Remark 3. In the robustness experiment, the adversary submits not only $\widetilde{P}$, but also the shift $\widetilde{\delta}$. In the previous works, such as [8], the authors considered two perturbation styles: 1)the shift is independent of W; 2) the shift can arbitrarily depend on W. In our definition, the shift is controlled by the adversary, and it just sits in the middle of the two styles. The reason we adopt such a definition is to make the perturbation style consistent with that in the reusability experiment.

4.2 Construction of Robustly Reusable Fuzzy Extractor

Figure 4 illustrates our construction of robustly reusable FE $\mathsf{rrFE} = (\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ for metric space $\mathcal{M}$, which makes use of the following building blocks:

- A key-shift secure symmetric key encapsulation mechanism $\mathsf{SKEM} = (\mathsf{SKEM.Init}, \mathsf{SKEM.Enc}, \mathsf{SKEM.Dec})$. Let its secret key space be $\mathcal{SK}$ and encapsulation key space be $\mathcal{K}$.
- A homomorphic average-case $(\mathcal{M}, \hat{m}, \mathcal{SK}, \varepsilon_{\mathsf{ext}})$-strong extractor Ext.
- A homomorphic $(m - \lceil \log p \rceil, \hat{m}, 2\mathsf{t})$-secure sketch $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ for metric space $\mathcal{M}$ with $\hat{m} - \lceil \log p \rceil \geq \omega(\log \lambda)$.

```
$\underline{\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)}$:
$(F_{pk}, F_{td}) \leftarrow \mathsf{FGen}(1^\lambda)$.
$i_{\mathsf{ext}} \leftarrow_{\$} \mathcal{I}$.
$\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$.
$\mathsf{crs} := (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$.
Return $\mathsf{crs}$.
```

```
$\underline{(R, P) \leftarrow \mathsf{Gen}(\mathsf{crs}, w)}$:
Parse $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$.
$s \leftarrow \mathsf{SS.Gen}(w)$.
$sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$.
$(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$.
$t := (s, c)$.
$t' \leftarrow_{\$} \mathcal{T}'$.
$\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$.
$P := (t = (s, c), t', \sigma)$
$R := k$.
Return $(P, R)$.
```

```
$\underline{R/\perp \leftarrow \mathsf{Rep}(\mathsf{crs}, \widetilde{P}, w')}$:
Parse $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$.
Parse $\widetilde{P} = (\widetilde{t} = (\widetilde{s}, \widetilde{c}), \widetilde{t'}, \widetilde{\sigma})$.
$\widetilde{w} \leftarrow \mathsf{SS.Rec}(w', \widetilde{s})$.
If $\mathsf{dis}(\widetilde{w}, w') > \mathfrak{t}$,
    Return $\perp$.
Else,
    $\widetilde{\sigma}' \leftarrow \mathsf{FEval}(F_{pk}, \widetilde{t}, \widetilde{t'}, \widetilde{w})$.
If $\widetilde{\sigma}' \neq \widetilde{\sigma}$,
    Return $\perp$.
Else,
    $\widetilde{sk} \leftarrow \mathsf{Ext}(\widetilde{w}, i_{\mathsf{ext}})$.
    $\widetilde{k} \leftarrow \mathsf{SKEM.Dec}(\mathsf{pp}, \widetilde{sk}, \widetilde{c})$.
    Return $\widetilde{k}$.
```

Fig. 4. Construction of robustly reusable fuzzy extractor rrFE.

– A homomorphic $(l_{\mathsf{LAF}}, \mathfrak{n})$-lossy algebraic filter $\mathsf{LAF} = (\mathsf{FGen}, \mathsf{FEval}, \mathsf{FTag})$ with domain $\mathbb{Z}_p^{\mathfrak{n}}$, $l_{\mathsf{LAF}} = \lceil \log p \rceil$, and tag space $\{0,1\}^* \times \mathcal{T}'$. We assume that any $w \in \mathcal{M}$ can be explained as an element in $\mathbb{Z}_p^{\mathfrak{n}}$.

The correctness of the fuzzy extractor follows from the correctness of the underlying SS and SKEM.

Theorem 2. *If the underlying* SKEM *is key-shift secure with secret key space* $\mathcal{SK}$ *and encapsulation key space* $\mathcal{K}$*,* Ext *is a homomorphic average-case* $(\mathcal{M}, \hat{m}, \mathcal{SK}, \varepsilon_{\mathsf{ext}})$*-strong extractor,* SS *is a homomorphic* $(m - \lceil \log p \rceil, \hat{m}, 2\mathfrak{t})$*-secure sketch for metric space* $\mathcal{M}$ *with* $\hat{m} - \lceil \log p \rceil \geq \omega(\log \lambda)$*, and* LAF *is a homomorphic* $(l_{\mathsf{LAF}}, \mathfrak{n})$*-lossy algebraic filter with domain* $\mathbb{Z}_p^{\mathfrak{n}}$ *and* $l_{\mathsf{LAF}} = \lceil \log p \rceil$*, and every element in* $\mathcal{M}$ *can be explained as an element in* $\mathbb{Z}_p^{\mathfrak{n}}$*, then the fuzzy extractor* rrFE *in Fig. 4 is an* $(\mathcal{M}, m, \mathcal{K}, \mathfrak{t}, \varepsilon_1, \varepsilon_2)$*-robustly reusable fuzzy extractor, where* $\varepsilon_1 = 2\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda) + 2\varepsilon_{\mathsf{ext}} + \mathsf{Adv}^{\mathsf{ks}}_{\mathsf{SKEM}}(1^\lambda)$ *and* $\varepsilon_2 = \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda) + \varepsilon_{\mathsf{ext}} + \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF}}(1^\lambda) + 2^{-\omega(\log \lambda)}$.

Proof. All we have to do is to show that rrFE is ε_1-reusable and ε_2-robust, which are proved in Theorem 3 and Theorem 4 respectively.

Theorem 3. *Given the building blocks specified in Theorem 2, the fuzzy extractor* rrFE *in Fig. 4 is* ε_1*-reusable, where* $\varepsilon_1 = 2\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda) + 2\varepsilon_{\mathsf{ext}} + \mathsf{Adv}^{\mathsf{ks}}_{\mathsf{SKEM}}(1^\lambda)$.

Proof. We will prove this theorem by a sequence of games. The changes from Game G_j to Game G_{j+1} are underlined.

Game G_0: It is exactly experiment $\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(1)$. More precisely,

1. Challenger $\mathcal{C}$ invokes $(F_{pk}, F_{td}) \leftarrow \mathsf{FGen}(1^\lambda)$ and $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$, samples a seed $i_{\mathsf{ext}} \leftarrow_{\$} \mathcal{I}$, sets $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$, and returns crs to $\mathcal{A}$.

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, sets $t := (s, c)$, samples $t' \leftarrow_{\$} \mathcal{T}'$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.
3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, challenger $\mathcal{C}$ invokes $s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)$, $sk_i \leftarrow \mathsf{Ext}(w + \delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, samples $t'_i \leftarrow_{\$} \mathcal{T}'$, invokes $\sigma_i \leftarrow \mathsf{FEval}(F_{pk}, t_i, t'_i, w + \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.
4. If $\mathcal{A}$ outputs a bit β', the game outputs β'.

Obviously,

$$\Pr[\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(1) \Rightarrow 1] = \Pr[\mathsf{G_0}^{\mathcal{A}} \Rightarrow 1]. \tag{6}$$

Game G_1: It is the same as G_0, except for conceptual changes of generating (P_i, R_i). More precisely,

3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, challenger $\mathcal{C}$ computes $\underline{s_i := s + \mathsf{SS.Gen}(\delta_i)}$, $\underline{sk_i := sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})}$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, samples $t'_i \leftarrow_{\$} \mathcal{T}'$, computes $\underline{\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)}$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 4. $\Pr[\mathsf{G_0}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G_1}^{\mathcal{A}} \Rightarrow 1]$.

Proof. By the homomorphic property of the deterministic secure sketch, we have:

$$s_i \overset{\mathsf{G}_0}{=} \mathsf{SS.Gen}(w + \delta_i) = \mathsf{SS.Gen}(w) + \mathsf{SS.Gen}(\delta_i) = s + \mathsf{SS.Gen}(\delta_i) \overset{\mathsf{G}_1}{=} s_i.$$

By the homomorphic property of Ext, we have:

$$sk_i \overset{\mathsf{G}_0}{=} \mathsf{Ext}(w + \delta_i, i_{\mathsf{ext}}) = \mathsf{Ext}(w, i_{\mathsf{ext}}) + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}}) = sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}}) \overset{\mathsf{G}_1}{=} sk_i.$$

Similarly, by the homomorphic property of LAF, we have:

$$\sigma_i \overset{\mathsf{G}_0}{=} \mathsf{FEval}(F_{pk}, t_i, t'_i, w + \delta_i) = \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i) \overset{\mathsf{G}_1}{=} \sigma_i.$$

Thus the changes are just conceptual, and Lemma 4 follows. ▮

Game G_2: It is the same as G_1, except that the core tags t', t'_i are not uniformly chosen any more. Now they are generated by FTag in G_2. More precisely,

2. Challenger $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, sets $t := (s, c)$, $\underline{\text{generates } t' \leftarrow \mathsf{FTag}(F_{td}, t)}$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.
3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, challenger $\mathcal{C}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, $\underline{\text{generates } t'_i \leftarrow \mathsf{FTag}(F_{td}, t_i)}$, computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 5. $|\Pr[\mathsf{G_1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1]| \le \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda)$.

Proof. Assume there exists a PPT adversary $\mathcal{A}$ such that $|\Pr[\mathsf{G_1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1]| = \epsilon$. We construct a PPT algorithm $\mathcal{B}$ who, given F_{pk}, can distinguish oracle $\mathsf{FTag}(F_{td}, \cdot)$ from oracle $\mathcal{O}_{\mathcal{T}'}(\cdot)$ with advantage ϵ. Algorithm $\mathcal{B}$ simulates an environment for $\mathcal{A}$ as follows:

- Given F_{pk}, algorithm $\mathcal{B}$ invokes $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$, samples a seed $i_{\mathsf{ext}} \leftarrow_\$ \mathcal{I}$, sets $\mathsf{crs} := (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$, and returns crs to $\mathcal{A}$.
- Algorithm $\mathcal{B}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$ and sets $t := (s, c)$.
  – $\mathcal{B}$ queries its own oracle with $t = (s, c)$, and the oracle replies $\mathcal{B}$ with t'. After receiving t' from its oracle, $\mathcal{B}$ invokes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (t = (s, c), t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.
- Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \le \mathsf{t}$, algorithm $\mathcal{B}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$ and sets $t_i = (s_i, c_i)$.
  – $\mathcal{B}$ queries its oracle with $t_i := (s_i, c_i)$, and the oracle replies $\mathcal{B}$ with t'_i. After receiving t'_i from its oracle, $\mathcal{B}$ computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}\ (F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.
- When $\mathcal{A}$ outputs a bit β', algorithm $\mathcal{B}$ returns β'.

Observe that if the oracle to which $\mathcal{B}$ has access is $\mathsf{FTag}(F_{td}, \cdot)$, then $\mathcal{B}$ perfectly simulates $\mathsf{G_2}$ for $\mathcal{A}$; otherwise it perfectly simulates $\mathsf{G_1}$ for $\mathcal{A}$. Thus

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF},\mathcal{B}}(1^\lambda) = |\Pr[\mathsf{G_1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1]|.$$

This completes the proof of Lemma 5. ▌

Game $\mathsf{G_3}$: It is the same as $\mathsf{G_2}$, except that sk is changed to a uniform one. More precisely,

2. Challenger $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $\underline{\text{samples } \widehat{sk} \leftarrow_\$ \mathcal{SK}}$, computes $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, \underline{\widehat{sk}})$, sets $t := (s, c)$, generates $t' \leftarrow \mathsf{FTag}\ (F_{td}, t)$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.
3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \le \mathsf{t}$, challenger $\mathcal{C}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := \underline{\widehat{sk}} + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, generates $t'_i \leftarrow \mathsf{FTag}(F_{td}, t_i)$, computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 6. $|\Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_3}^{\mathcal{A}} \Rightarrow 1]| \le \varepsilon_{\mathsf{ext}}$.

Proof. Assume that $\mathcal{A}$ makes ρ queries to the challenger. The only difference between $\mathsf{G_2}$ and $\mathsf{G_3}$ is that $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$ in $\mathsf{G_2}$ is changed to $\widehat{sk} \leftarrow_\$ \mathcal{SK}$ in

G_3. We will show that the views of adversary $\mathcal{A}$ in G_2 and G_3 are statistically indistinguishable.

Since F_{pk}, F_{td} and pp are independent of W, we have

$$\widetilde{H}_\infty(W \mid (F_{pk}, F_{td}, \mathsf{pp})) = \widetilde{H}_\infty(W) \geq m. \tag{7}$$

Define $\mathcal{S} := \{\sigma \mid \sigma = \mathsf{FEval}(F_{pk}, t, t', W) \wedge \mathsf{tag} = (t, t') \in \mathcal{T}_{lossy}\}$, which collects all function values w.r.t. the same W and the same F_{pk} but under all possible lossy tags. By the lossiness of LAF, $\mathcal{S}$ only reveals $\log p$ bits information of W (see remark 2). According to Lemma 1 and Eq. (7), we have

$$\widetilde{H}_\infty(W \mid (F_{pk}, F_{td}, \mathsf{pp}, \mathcal{S})) \geq \widetilde{H}_\infty(W \mid (F_{pk}, F_{td}, \mathsf{pp})) - \log p \geq m - \log p. \tag{8}$$

Since SS is a $(m - \log p, \hat{m}, 2\mathsf{t})$-secure sketch, we have

$$\widetilde{H}_\infty(W \mid (s = \mathsf{SS.Gen}(W), F_{pk}, F_{td}, \mathsf{pp}, \mathcal{S})) \geq \hat{m}.$$

Define $\mathsf{AuxiliaryInput} := (s = \mathsf{SS.Gen}(W), F_{pk}, F_{td}, \mathsf{pp}, \mathcal{S})$. Obviously $\mathsf{AuxiliaryInput}$ is independent of i_{ext}. According to Eq. (3), the average-case $(\mathcal{M}, \hat{m}, \mathcal{SK}, \varepsilon_{\mathsf{ext}})$-strong extractor Ext implies

$$\Big(sk, i_{\mathsf{ext}}, \underbrace{(s = \mathsf{SS.Gen}(W), F_{pk}, F_{td}, \mathsf{pp}, \mathcal{S})}_{\mathsf{AuxiliaryInput}} \Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\widehat{sk}, i_{\mathsf{ext}}, \underbrace{(s = \mathsf{SS.Gen}(W), F_{pk}, F_{td}, \mathsf{pp}, \mathcal{S})}_{\mathsf{AuxiliaryInput}} \Big), \tag{9}$$

where $sk := \mathsf{Ext}(W, i_{\mathsf{ext}})$ and $\widehat{sk} \leftarrow_{\$} \mathcal{SK}$. Since $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$, Eq. (9) implies

$$\Big(\underbrace{sk, s = \mathsf{SS.Gen}(W), \mathsf{crs}, F_{td}, \mathcal{S}}_{\varOmega} \Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\widehat{sk}, s = \mathsf{SS.Gen}(W), \mathsf{crs}, F_{td}, \mathcal{S}}_{\varXi} \Big). \tag{10}$$

Let w be a specific value taken by random variable W.

Recall that $P = (s, c, t', \sigma)$ and $R = k$, where $s \leftarrow \mathsf{SS.Gen}(w)$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, $t := (s, c)$, $t' \leftarrow \mathsf{FTag}(F_{td}, t)$, $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$. Obviously, (P, R) can be regarded as an output of some randomized function on input $\varOmega$.

Define $\widehat{P} := (s, \widehat{c}, \widehat{t'}, \widehat{\sigma})$ and $\widehat{R} := \widehat{k}$, where $s \leftarrow \mathsf{SS.Gen}(w)$, $(\widehat{c}, \widehat{k}) \leftarrow \mathsf{SKEM.Enc}$ $(\mathsf{pp}, \widehat{sk})$, $\widehat{t} = (s, \widehat{c})$, $\widehat{t'} \leftarrow \mathsf{FTag}(F_{td}, \widehat{t})$, $\widehat{\sigma} \leftarrow \mathsf{FEval}(F_{pk}, \widehat{t}, \widehat{t'}, w)$. In other words, $(\widehat{P}, \widehat{R})$ is the helper string and the extracted string generated with the random key $\widehat{sk}$. Then $(\widehat{P}, \widehat{R})$ can be regarded as an output of the same randomized function on input $\varXi$ as that for (P, R).

According to Lemma 2, Formula (10) implies

$$\Big(\underbrace{\overbrace{sk := \mathsf{Ext}(W, i_{\mathsf{ext}}), s = \mathsf{SS.Gen}(W), \mathsf{crs}, F_{td}, \mathcal{S}}^{\varOmega}, P, R}_{\varOmega_0} \Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\overbrace{\widehat{sk} \leftarrow_{\$} \mathcal{SK}, s = \mathsf{SS.Gen}(W), \mathsf{crs}, F_{td}, \mathcal{S}}^{\varXi}, \widehat{P}, \widehat{R}}_{\varXi_0} \Big),$$

in short,
$$\Big(\underbrace{\varOmega, P, R}_{\varOmega_0}\Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\varXi, \widehat{P}, \widehat{R}}_{\varXi_0}\Big). \tag{11}$$

Before $\mathcal{A}$ submits its first query δ_1 in G_2, its view is described by $\langle \mathsf{crs}, P, R\rangle$. Obviously, δ_1 can be computed by some randomized function of $\langle \mathsf{crs}, P, R\rangle$ (the function is determined by $\mathcal{A}$'s strategy). Naturally, it can be regarded as an output of some randomized function on input $\varOmega_0$.

Similarly, the first query $\widehat{\delta_1}$ of $\mathcal{A}$ in G_3 is determined by the same randomized function of its view $\langle \mathsf{crs}, \widehat{P}, \widehat{R}\rangle$, hence it can also be regarded as an output of the same randomized function of $\varXi_0$.

By Lemma 2 again, Formula (11) implies
$$\Big(\underbrace{\varOmega, P, R, \delta_1}_{\varOmega'_0}\Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\varXi, \widehat{P}, \widehat{R}, \widehat{\delta_1}}_{\varXi'_0}\Big). \tag{12}$$

Recall that $P_1 := (s_1, c_1, t'_1, \sigma_1)$, $R_1 := k_1$, where $s_1 = s + \mathsf{SS.Gen}(\delta_1)$, $sk_1 = sk + \mathsf{Ext}(\delta_1, i_{\mathsf{ext}})$, $(c_1, k_1) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_1)$, $t_1 = (s_1, c_1)$, $t'_1 \leftarrow \mathsf{FTag}(F_{td}, t_1)$ $\sigma_1 = \mathsf{FEval}(F_{pk}, t_1, t'_1, w) + \mathsf{FEval}(F_{pk}, t_1, t'_1, \delta_1)$. Note that (t_1, t'_1) is a lossy tag, hence $\mathsf{FEval}(F_{pk}, t_1, t'_1, w) \in \mathcal{S}$. Obviously, P_1 and R_1 can be determined by some randomized function of $\varOmega'_0$.

Define $\widehat{P_1} := (\widehat{s_1}, \widehat{c_1}, \widehat{t'_1}, \widehat{\sigma_1})$, $\widehat{R_1} := \widehat{k_1}$, where $\widehat{s_1} = s + \mathsf{SS.Gen}(\widehat{\delta_1})$, $\widehat{sk_1} = \widehat{sk} + \mathsf{Ext}(\widehat{\delta_1}, i_{\mathsf{ext}})$, $(\widehat{c_1}, \widehat{k_1}) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, \widehat{sk_1})$, $\widehat{t_1} = (\widehat{s_1}, \widehat{c_1})$, $\widehat{t'_1} \leftarrow \mathsf{FTag}(F_{td}, \widehat{t_1})$ $\widehat{\sigma_1} = \mathsf{FEval}(F_{pk}, \widehat{t_1}, \widehat{t'_1}, w) + \mathsf{FEval}(F_{pk}, \widehat{t_1}, \widehat{t'_1}, \widehat{\delta_1})$. Similarly, $\widehat{P_1}$ and $\widehat{R_1}$ can be determined by the same randomized function of of $\varXi'_0$.

Applying Lemma 2 once more, Formula (12) implies
$$\Big(\underbrace{\overbrace{\varOmega, P, R, \delta_1}^{\varOmega'_0}, P_1, R_1}_{\varOmega_1}\Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\overbrace{\varXi, \widehat{P}, \widehat{R}, \widehat{\delta_1}}^{\varXi'_0}, \widehat{P_1}, \widehat{R_1}}_{\varXi_1}\Big). \tag{13}$$

By induction on $i \in [\rho]$, we have that
$$\Big(\underbrace{\varOmega, P, R, \{\delta_i, P_i, R_i\}_{i\in[\rho]}}_{\varOmega_\rho}\Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\varXi, \widehat{P}, \widehat{R}, \{\widehat{\delta_i}, \widehat{P_i}, \widehat{R_i}\}_{i\in[\rho]}}_{\varXi_\rho}\Big). \tag{14}$$

More precisely,
$$\begin{gathered}\Big(\underbrace{\overbrace{sk := \mathsf{Ext}(W, i_{\mathsf{ext}}), s = \mathsf{SS.Gen}(W), \mathsf{crs}, F_{td}, \mathcal{S}}^{\varOmega}, P, R, \{\delta_i, P_i, R_i\}_{i\in[\rho]}}_{\varOmega_\rho}\Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \\ \Big(\underbrace{\overbrace{sk := \mathsf{Ext}(W, i_{\mathsf{ext}}), s = \mathsf{SS.Gen}(W), \mathsf{crs}, F_{td}, \mathcal{S}}^{\varXi}, \widehat{P}, \widehat{R}, \{\widehat{\delta_i}, \widehat{P_i}, \widehat{R_i}\}_{i\in[\rho]}}_{\varXi_\rho}\Big).\end{gathered} \tag{15}$$

(15) implies
$$\Big(\underbrace{\mathsf{crs}, P, R, \{\delta_i, P_i, R_i\}_{i\in[\rho]}}_{\varOmega^*_\rho}\Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\mathsf{crs}, \widehat{P}, \widehat{R}, \{\widehat{\delta_i}, \widehat{P_i}, \widehat{R_i}\}_{i\in[\rho]}}_{\varXi^*_\rho}\Big). \tag{16}$$

Observe that Ω^*_ρ is just the whole view of $\mathcal{A}$ in G_2, and Ξ^*_ρ is the whole view of $\mathcal{A}$ in G_3. The statistical distance of Ω^*_ρ and Ξ^*_ρ is smaller than $\varepsilon_{\mathsf{ext}}$. As a consequence, we have $|\Pr[{\mathsf{G}_2}^{\mathcal{A}} \Rightarrow 1] - \Pr[{\mathsf{G}_3}^{\mathcal{A}} \Rightarrow 1]| \leq \varepsilon_{\mathsf{ext}}$. ▮

Game G_4: It is the same as G_3, except that R is uniformly chosen from $\mathcal{K}$ instead of being output by SKEM. More precisely,

2. Challenger $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, samples $\widehat{sk} \leftarrow_\$ \mathcal{SK}$, computes $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, \widehat{sk})$, sets $t := (s, c)$, generates $t' \leftarrow \mathsf{FTag}(F_{td}, t)$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, $\underline{\text{samples } R \leftarrow_\$ \mathcal{K}}$, and returns (P, R) to $\mathcal{A}$.

Lemma 7. $|\Pr[{\mathsf{G}_3}^{\mathcal{A}} \Rightarrow 1] - \Pr[{\mathsf{G}_4}^{\mathcal{A}} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{ks}}_{\mathsf{SKEM}}(1^\lambda)$.

Proof. Assume there exists a PPT adversary $\mathcal{A}$ such that $|\Pr[{\mathsf{G}_3}^{\mathcal{A}} \Rightarrow 1] - \Pr[{\mathsf{G}_4}^{\mathcal{A}} \Rightarrow 1]| = \epsilon$. We construct a PPT algorithm $\mathcal{B}$ who can implement the key-shift attack with the same advantage ϵ. Algorithm $\mathcal{B}$ simulates an environment for $\mathcal{A}$ as follows:

- After receiving pp from its own challenger, algorithm $\mathcal{B}$ invokes $(F_{pk}, F_{td}) \leftarrow \mathsf{FGen}(1^\lambda)$, samples a seed $i_{\mathsf{ext}} \leftarrow_\$ \mathcal{I}$, sets $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$, and returns crs to $\mathcal{A}$.
- Algorithm $\mathcal{B}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, asks the *challenge* oracle of SKEM to get (c, k). Then $\mathcal{B}$ sets $t := (s, c)$, generates $t' \leftarrow \mathsf{FTag}(F_{td}, t)$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (t = (s, c), t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.
- Upon receiving a shift $\delta_i \in \mathcal{M}$ queried from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, algorithm $\mathcal{B}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $\Delta_i := \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, asks its *encapsulation* oracle with Δ_i to obtain (c_i, k_i), where $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk + \Delta_i)$. Then $\mathcal{B}$ sets $t_i := (s_i, c_i)$, generates $t'_i \leftarrow \mathsf{FTag}(F_{td}, t_i)$, computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}\ (F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.
- When $\mathcal{A}$ outputs a bit β', algorithm $\mathcal{B}$ outputs β' to its own challenger.

Note that if (c, k) is generated by $(c, k) \leftarrow \mathsf{SKEM.Enc}(pp, sk)$, then algorithm $\mathcal{B}$ perfectly simulates G_3 for $\mathcal{A}$; otherwise k is uniformly chosen from $\mathcal{K}$, then algorithm $\mathcal{B}$ perfectly simulates G_4 for $\mathcal{A}$. Hence $\mathcal{B}$ shares exactly the same advantage with $\mathcal{A}$. Thus $\mathsf{Adv}^{\mathsf{mrka}}_{\mathsf{SKEM},\mathcal{B}}(1^\lambda) = |\Pr[{\mathsf{G}_3}^{\mathcal{A}} \Rightarrow 1] - \Pr[{\mathsf{G}_4}^{\mathcal{A}} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{mrka}}_{\mathsf{SKEM}}(1^\lambda)$. This completes the proof of Lemma 7. ▮

Game G_5: It is the same as G_4, except that the generation of $\widehat{sk}$ is changed back to $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$. More precisely,

2. Challenger $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $\underline{sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})}$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, \underline{sk})$, sets $t := (s, c)$, generates $t' \leftarrow \mathsf{FTag}(F_{td}, t)$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, samples $R \leftarrow_\$ \mathcal{K}$, and returns (P, R) to $\mathcal{A}$.

3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \le \mathfrak{t}$, challenger $\mathcal{C}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := \underline{sk} + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, generates $t'_i \leftarrow \mathsf{FTag}(F_{td}, t_i)$, computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 8. $|\Pr[\mathsf{G_4}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_5}^{\mathcal{A}} \Rightarrow 1]| \le \varepsilon_{\mathsf{ext}}$.

Proof. The proof is similar to the proof of Lemma 6, since the changes from G_4 to G_5 is symmetric to that from G_2 to G_3. We omit the proof here. ∎

Game G_6: It is the same as G_5, except that the core tags are changed back to random tags. More precisely,

2. $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, sets $t := (s, c)$, $\underline{\text{samples } t' \leftarrow_{\$} \mathcal{T}'}$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, samples $R \leftarrow_{\$} \mathcal{K}$, and returns (P, R) to $\mathcal{A}$.
3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \le \mathfrak{t}$, $\mathcal{C}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, $\underline{\text{samples } t'_i \leftarrow_{\$} \mathcal{T}'}$, computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 9. $|\Pr[\mathsf{G_5}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_6}^{\mathcal{A}} \Rightarrow 1]| \le \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda)$.

Proof. The proof is similar to the proof of Lemma 5, since the changes from G_5 to G_6 is symmetric to that from G_1 to G_2. We omit the proof here. ∎

Game G_7: It is the same as G_6, except for conceptual changes of generating (P_i, R_i). More precisely,

3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \le \mathfrak{t}$, challenger $\mathcal{C}$ computes $\underline{s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)}$, $\underline{sk_i \leftarrow \mathsf{Ext}(w + \delta_i, i_{\mathsf{ext}})}$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, samples $t'_i \leftarrow_{\$} \mathcal{T}'$, computes $\underline{\sigma_i \leftarrow \mathsf{FEval}(F_{pk}, t_i, t'_i, w + \delta_i)}$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 10. $\Pr[\mathsf{G_6}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G_7}^{\mathcal{A}} \Rightarrow 1]$.

Proof. The proof is identical to the proof of Lemma 4, since the changes from G_6 to G_7 is symmetric to that from G_0 to G_1. We omit the proof here. ∎

Note that G_7 is identical to experiment $\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(0)$. Thus

$$\Pr[\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}}(0) \Rightarrow 1] = \Pr[\mathsf{G}_7 \Rightarrow 1]. \tag{17}$$

Taking all things together, by Eq. (6), Lemma 4-10 and Eq. (17), we have that

$$\mathsf{Adv}^{\mathsf{reu}}_{\mathsf{rFE},\mathcal{A}} \le 2\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda) + 2\varepsilon_{\mathsf{ext}} + \mathsf{Adv}^{\mathsf{ks}}_{\mathsf{SKEM}}(1^\lambda).$$

This completes the proof of Theorem 3. ∎

Theorem 4. *Given the building blocks specified in Theorem 2, the fuzzy extractor* rrFE *in Fig. 4 is* ε_2*-robust, where* $\varepsilon_2 = \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda) + \varepsilon_{\mathsf{ext}} + \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF}}(1^\lambda) + 2^{-\omega(\log \lambda)}$.

Proof. Similar to the proof of reusability, we will prove this theorem by a sequence of games again. The changes from Game G_j to adjacent Game G_{j+1} are underlined. Let win_j denote the event that adversary $\mathcal{A}$ wins in G_j. G_j outputs 1 if $\mathcal{A}$ wins and 0 otherwise. Obviously, $\Pr[\mathsf{win}_j] = \Pr[\mathsf{G}_j{}^{\mathcal{A}} \Rightarrow 1]$.

Game G_0: It is identical to the robustness experiment $\mathsf{Exp}^{\mathsf{rob}}_{\mathsf{rrFE},\mathcal{A}}(1^\lambda)$.

1. Challenger $\mathcal{C}$ invokes $(F_{pk}, F_{td}) \leftarrow \mathsf{FGen}(1^\lambda)$ and $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$, samples a seed $i_{\mathsf{ext}} \leftarrow_{\$} \mathcal{I}$, sets $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$, and returns crs to $\mathcal{A}$.
2. Challenger $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, sets $t := (s, c)$, samples $t' \leftarrow_{\$} \mathcal{T}'$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (t = (s, c), t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.
3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, challenger $\mathcal{C}$ computes $s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)$, $sk_i \leftarrow \mathsf{Ext}(w + \delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, samples $t'_i \leftarrow_{\$} \mathcal{T}'$, computes $\sigma_i \leftarrow \mathsf{FEval}(F_{pk}, t_i, t'_i, w + \delta_i)$, sets $P_i := (t_i = (s_i, c_i), t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.
4. $\mathcal{A}$ submits to $\mathcal{C}$ its forgery $(\widetilde{P}, \widetilde{\delta})$ with $\widetilde{P} = (\widetilde{t} = (\widetilde{s}, \widetilde{c}), \widetilde{t}', \widetilde{\sigma})$. $\mathcal{A}$ wins if $\mathsf{dis}(\widetilde{\delta}) \leq \mathsf{t}$, $\widetilde{P}$ is fresh and $\mathsf{Rep}(\mathsf{crs}, \widetilde{P}, w + \widetilde{\delta}) \neq \perp$. Recall that $\mathsf{Rep}(\mathsf{crs}, \widetilde{P}, w + \widetilde{\delta}) \neq \perp$ if and only if $\mathsf{dis}(\widetilde{w}, w + \widetilde{\delta}) \leq \mathsf{t}$ and $\widetilde{\sigma}' = \widetilde{\sigma}$ holds, where $\widetilde{w} \leftarrow \mathsf{SS.Rec}(w + \widetilde{\delta}, \widetilde{s})$ and $\widetilde{\sigma}' \leftarrow \mathsf{FEval}(F_{pk}, \widetilde{t}, \widetilde{t}', \widetilde{w})$. The game outputs 1 if $\mathcal{A}$ wins and 0 otherwise.

   Obviously,
$$\Pr[\mathsf{G}_0{}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{Exp}^{\mathsf{rob}}_{\mathsf{rrFE},\mathcal{A}}(1^\lambda) \Rightarrow 1]. \tag{18}$$

Game G_1: It is the same as G_0, except for conceptual changes of generating (P_i, R_i). More precisely,

3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, challenger $\mathcal{C}$ computes $\underline{s_i := s + \mathsf{SS.Gen}(\delta_i)}$, $\underline{sk_i := sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})}$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, samples $t'_i \leftarrow_{\$} \mathcal{T}'$, computes $\underline{\sigma_i \leftarrow \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)}$, sets $P_i := (t_i = (s_i, c_i), t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 11. $\Pr[\mathsf{G}_0{}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}_1{}^{\mathcal{A}} \Rightarrow 1]$.

Proof. The changes are just conceptual by the homomorphic properties of SS, Ext, LAF. Similar to the proof of Lemma 4, Lemma 11 follows. ▌

Game G_2: It is the same as G_1, except that the core tags t', t'_i are not uniformly chosen any more. Now they are generated by FTag in G_2. More precisely,

2. Challenger $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, sets $t := (s, c)$, $\underline{\text{generates } t' \leftarrow \mathsf{FTag}(F_{td}, t)}$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.

3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, challenger $\mathcal{C}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$, $\underline{\text{generates } t'_i \leftarrow \mathsf{FTag}(F_{td}, t_i)}$, computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 12. $|\Pr[\mathsf{G_1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^{\lambda})$.

Proof. The proof is similar to that of Lemma 5 (the difference is the output strategy of algorithm $\mathcal{B}$). Assume there exists a PPT adversary $\mathcal{A}$ such that $|\Pr[\mathsf{G_1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1]| = \epsilon$. We construct a PPT algorithm $\mathcal{B}$ who, given F_{pk}, can distinguish oracle $\mathsf{FTag}(F_{td}, \cdot)$ from oracle $\mathcal{O}_{\mathcal{T}'}(\cdot)$ with advantage ϵ. Algorithm $\mathcal{B}$ simulates an environment for $\mathcal{A}$ as follows:

- Given F_{pk}, algorithm $\mathcal{B}$ invokes $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^{\lambda})$, samples a seed $i_{\mathsf{ext}} \leftarrow_{\$} \mathcal{I}$, sets $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$, and returns crs to $\mathcal{A}$.
- Algorithm $\mathcal{B}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$, $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk)$, sets $t := (s, c)$ and queries its oracle with t to obtain t'. After receiving t' from its oracle, $\mathcal{B}$ computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (t = (s, c), t', \sigma)$, $R := k$, and gives (P, R) to $\mathcal{A}$.
- Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, algorithm $\mathcal{B}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := sk + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i := (s_i, c_i)$ and queries its oracle with t_i to obtain t'_i. After receiving t'_i from its oracle, $\mathcal{B}$ computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}\ (F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.
- When $\mathcal{A}$ submits its forgery $(\widetilde{P} = (\widetilde{s}, \widetilde{c}, \widetilde{t}', \widetilde{\sigma}), \widetilde{\delta})$, algorithm $\mathcal{B}$ checks whether $\mathcal{A}$ wins. $\mathcal{B}$ returns 1 if $\mathcal{A}$ wins; otherwise, it returns 0.

Recall that $\mathcal{A}$ wins means that conditions $\mathsf{dis}(\widetilde{\delta}) \leq \mathsf{t}$, $\widetilde{P}$ is fresh and $\mathsf{Rep}(\mathsf{crs}, \widetilde{P}, w + \widetilde{\delta}) \neq \perp$ are satisfied. These conditions can be efficiently checked by $\mathcal{B}$. Moreover, if the oracle to which $\mathcal{B}$ has access is $\mathsf{FTag}(F_{td}, \cdot)$, then $\mathcal{B}$ perfectly simulates $\mathsf{G_2}$ for $\mathcal{A}$; otherwise it perfectly simulates $\mathsf{G_1}$ for $\mathcal{A}$. Thus

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF},\mathcal{B}}(1^{\lambda}) = \big|\Pr[\mathsf{win}_1] - \Pr[\mathsf{win}_2]\big| = \big|\Pr[\mathsf{G_1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1]\big|.$$

This completes the proof of Lemma 12. ▮

Game $\mathsf{G_3}$: It is the same as $\mathsf{G_2}$, except that sk is changed to a uniform one. More precisely,

2. Challenger $\mathcal{C}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, $\underline{\text{samples } \widehat{sk} \leftarrow_{\$} \mathcal{SK}}$, computes $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, \underline{\widehat{sk}})$, sets $t := (s, c)$, generates $t' \leftarrow \mathsf{FTag}(F_{td}, t)$, computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$, $R := k$, and returns (P, R) to $\mathcal{A}$.
3. Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \leq \mathsf{t}$, challenger $\mathcal{C}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := \underline{\widehat{sk}} + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$, sets $t_i = (s_i, c_i)$, generates $t'_i \leftarrow \mathsf{FTag}(F_{td}, t_i)$, computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w) + \mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$, $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

Lemma 13. $\big|\Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_3}^{\mathcal{A}} \Rightarrow 1]\big| \le \varepsilon_{\mathsf{ext}}$.

Proof. The only difference between G_2 and G_3 is that $sk \leftarrow \mathsf{Ext}(w, i_{\mathsf{ext}})$ in G_2 is changed to $\widehat{sk} \leftarrow_{\$} \mathcal{SK}$ in G_3. The proof is exactly the same as that of Lemma 6.

Assume that $\mathcal{A}$ makes ρ queries to the challenger before submitting its forgery $(\widetilde{P}, \widetilde{\delta})$. Following similar arguments as those in the proof Lemma 6, we can show that the views of adversary $\mathcal{A}$ before submitting the forgery in G_2 and G_3 are statistically indistinguishable, i.e.,

$$\Big(\underbrace{\mathsf{crs}, P, R, \{\delta_i, P_i, R_i\}_{i\in[\rho]}}_{\Omega^*_\rho}\Big) \overset{\varepsilon_{\mathsf{ext}}}{\approx} \Big(\underbrace{\mathsf{crs}, \widehat{P}, \widehat{R}, \{\widehat{\delta_i}, \widehat{P}_i, \widehat{R_i}\}_{i\in[\rho]}}_{\Xi^*_\rho}\Big). \qquad (19)$$

Here Ω^*_ρ summerizes the view of $\mathcal{A}$ in G_2, and Ξ^*_ρ the view of $\mathcal{A}$ in G_3 before $\mathcal{A}$ submits its forgery. The statistical distance of Ω^*_ρ and Ξ^*_ρ is smaller than $\varepsilon_{\mathsf{ext}}$. As a consequence,

$$\big|\Pr[\mathsf{win}_2] - \Pr[\mathsf{win}_3]\big| = \big|\Pr[\mathsf{G_2}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G_3}^{\mathcal{A}} \Rightarrow 1]\big| \le \varepsilon_{\mathsf{ext}}. \quad ▮$$

Lemma 14. $\Pr[\mathsf{win}_3] \le \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF}}(1^\lambda) + 2^{-\omega(\log\lambda)}$.

Proof. Let bad denote the event that $\mathcal{A}$'s forgery $\widetilde{P} = (\widetilde{t}, \widetilde{t'}, \widetilde{\sigma})$ contains a non-injective tag, i.e., $(\widetilde{t}, \widetilde{t'}) \notin \mathcal{T}_{inj}$. We have

$$\Pr[\mathsf{win}_3] = \Pr[\mathsf{win}_3 \wedge \mathsf{bad}] + \Pr[\mathsf{win}_3 \wedge \neg\mathsf{bad}]. \qquad (20)$$

Thus it suffices to prove the following two claims.

Claim. $\Pr[\mathsf{win}_3 \wedge \mathsf{bad}] \le \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF}}(1^\lambda)$.

Proof. If there exists a PPT adversary $\mathcal{A}$ whose forgery makes $\mathsf{win}_3 \wedge \mathsf{bad}$ happen in G_3, we can construct a PPT algorithm $\mathcal{B}$ attacking on LAF's evasiveness. Given F_{pk} and a lossy tag generation oracle $\mathsf{FTag}(F_{td}, \cdot)$, $\mathcal{B}$ aims to output a new lossy tag. To this end, $\mathcal{B}$ simulates G_3 for $\mathcal{A}$ as follows:

- After receiving F_{pk} from its own challenger, $\mathcal{B}$ invokes $\mathsf{pp} \leftarrow \mathsf{SKEM.Init}(1^\lambda)$, samples a seed $i_{\mathsf{ext}} \leftarrow_{\$} \mathcal{I}$, sets $\mathsf{crs} = (F_{pk}, i_{\mathsf{ext}}, \mathsf{pp})$, and returns crs to $\mathcal{A}$.
- $\mathcal{B}$ samples $w \leftarrow W$, computes $s \leftarrow \mathsf{SS.Gen}(w)$, samples $\widehat{sk} \leftarrow_{\$} \mathcal{SK}$, computes $(c, k) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, \widehat{sk})$, and sets $t := (s, c)$.
  - $\mathcal{B}$ asks its own lossy tag generation oracle $\mathsf{FTag}(F_{td}, \cdot)$ with $t = (s, c)$ and obtains t' from the oracle. Obviously the oracle generates t' by $t' \leftarrow \mathsf{FTag}(F_{td}, t)$.

  $\mathcal{B}$ computes $\sigma \leftarrow \mathsf{FEval}(F_{pk}, t, t', w)$, sets $P := (s, c, t', \sigma)$ and $R := k$, and returns (P, R) to $\mathcal{A}$.
- Upon receiving a shift $\delta_i \in \mathcal{M}$ from $\mathcal{A}$ with $\mathsf{dis}(\delta_i) \le \mathsf{t}$, $\mathcal{B}$ computes $s_i := s + \mathsf{SS.Gen}(\delta_i)$, $sk_i := \widehat{sk} + \mathsf{Ext}(\delta_i, i_{\mathsf{ext}})$, $(c_i, k_i) \leftarrow \mathsf{SKEM.Enc}(\mathsf{pp}, sk_i)$ and sets $t_i := (s_i, c_i)$.

- $\mathcal{B}$ asks its own lossy tag generation oracle $\mathsf{FTag}(F_{td}, \cdot)$ with $t_i = (s_i, c_i)$ and obtains t'_i from the oracle. Obviously the oracle generates t'_i by $t'_i \leftarrow \mathsf{FTag}(F_{td}, t_i)$.

  $\mathcal{B}$ computes $\sigma_i := \mathsf{FEval}(F_{pk}, t_i, t'_i, w)+$ $\mathsf{FEval}(F_{pk}, t_i, t'_i, \delta_i)$, sets $P_i := (s_i, c_i, t'_i, \sigma_i)$ and $R_i := k_i$, and returns (P_i, R_i) to $\mathcal{A}$.

- When $\mathcal{A}$ outputs its forgery $\left(\widetilde{P} = (\widetilde{t} = (\widetilde{s}, \widetilde{c}), \widetilde{t'}, \widetilde{\sigma}),\ \widetilde{\delta}\right)$, $\mathcal{B}$ returns the tag $(\widetilde{t}, \widetilde{t'})$ to it own challenger.

Note that $\mathcal{B}$ perfectly simulates G_3 for $\mathcal{A}$, since its oracle generates lossy tags with $\mathsf{FTag}(F_{td}, t_i)$.

If event $\mathsf{win}_3 \wedge \mathsf{bad}$ occurs, the forged helper string $\widetilde{P}$ must be fresh, i.e., $\widetilde{P} \neq P$ and $\widetilde{P} \neq P_i$ for $i \in [\rho]$. Define freshT as the event that the forged tag $(\widetilde{t}, \widetilde{t'})$ is a fresh one, i.e., $(\widetilde{t}, \widetilde{t'}) \neq (t, t')$ and $(\widetilde{t}, \widetilde{t'}) \neq (t_i, t'_i)$ for all $i \in [\rho]$. Clearly,

$$\Pr[\mathsf{win}_3 \wedge \mathsf{bad}] = \Pr[\underbrace{\mathsf{win}_3 \wedge \mathsf{bad} \wedge \neg\mathsf{freshT}}_{\mathsf{Case\ 1}}] + \Pr[\underbrace{\mathsf{win}_3 \wedge \mathsf{bad} \wedge \mathsf{freshT}}_{\mathsf{Case\ 2}}]. \tag{21}$$

Case 1. In this case, freshT does not happen. Then we have $(\widetilde{t}, \widetilde{t'}) = (t, t')$ or $(\widetilde{t}, \widetilde{t'}) = (t_i, t'_i)$ for some $i \in [\rho]$. With loss of generality, we assume that $(\widetilde{t}, \widetilde{t'}) = (t_i, t'_i)$. Clearly $\widetilde{t} = t_i$ implies $\widetilde{s} = s_i$. Note that $\mathsf{dis}(\delta_i) \leq \mathsf{t}$ and $\mathsf{dis}(\widetilde{\delta}) \leq \mathsf{t}$, thus $\mathsf{dis}(w + \widetilde{\delta}, w + \delta_i) \leq \mathsf{dis}(w + \widetilde{\delta}, w) + \mathsf{dis}(w, w + \delta_i) \leq 2\mathsf{t}$. By the correctness of $(m - \lceil \log p \rceil, \hat{m}, 2\mathsf{t})$-secure sketch SS, we have $\widetilde{w} = w + \delta_i$, where $\widetilde{w} \leftarrow \mathsf{SS.Rec}(w + \widetilde{\delta}, s_i)$ and $s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)$. As a result,

$$\widetilde{\sigma}' = \mathsf{FEval}(\widetilde{t}, \widetilde{t'}, \widetilde{w}) = \mathsf{FEval}(t_i, t'_i, w + \delta_i) = \sigma_i.$$

If win_3 occurs, then $\widetilde{\sigma} = \widetilde{\sigma}'$ must hold. This implies $\widetilde{P} = (\widetilde{t}, \widetilde{t'}, \widetilde{\sigma}) = (t_i, t'_i, \sigma_i) = P_i$. This contradicts to the requirement of win_3 that $\widetilde{P}$ is fresh. Thus we have

$$\Pr[\mathsf{win}_3 \wedge \mathsf{bad} \wedge \neg\mathsf{freshT}] = 0. \tag{22}$$

Case 2. If both bad and freshT occur, then the forged tag $(\widetilde{t}, \widetilde{t'})$ is a fresh non-injective tag. Observe that $\mathcal{B}$ perfectly simulates G_3 for $\mathcal{A}$, then $\mathcal{B}$ succeeds in outputting a fresh non-injective tag, as long as $\mathsf{bad} \wedge \mathsf{freshT}$ occurs. Consequently,

$$\Pr[\mathsf{win}_3 \wedge \mathsf{bad} \wedge \mathsf{freshT}] \leq \Pr[\mathsf{bad} \wedge \mathsf{freshT}] = \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF},\mathcal{B}}(1^\lambda). \tag{23}$$

Combining (21), (22) and (23) together, we have

$$\Pr[\mathsf{win}_3 \wedge \mathsf{bad}] \leq \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF},\mathcal{B}}(1^\lambda). \quad \blacksquare$$

Claim. $\Pr[\mathsf{win}_3 | \neg\mathsf{bad}] \leq 2^{-\omega(\log \lambda)}$.

Proof. In G_3, adversary $\mathcal{A}$ interacts with the challenger and presents its forgery $(\widetilde{P}, \widetilde{\delta})$ at the end. Define $\mathcal{A}$'s view before it submits its forgery as

$$\boxed{\mathsf{view}} := (\mathsf{crs}, P, R, \{\delta_i, P_i, R_i\}_{i \in [\rho]}) = \big(\mathsf{crs}, (s, c, t', \sigma), k, \{\delta_i, (s_i, c_i, t'_i, \sigma_i), k_i\}_{i \in [\rho]}\big).$$

Given the forgery $(\widetilde{P} = (\widetilde{s}, \widetilde{c}, \widetilde{t}', \widetilde{\sigma}),\ \widetilde{\delta})$, $\mathcal{A}$ wins if $\mathsf{Rep}(\mathsf{crs}, \widetilde{P}, w + \widetilde{\delta}) \neq \perp$, $\widetilde{P}$ is fresh and $\mathsf{dis}(\widetilde{\delta}) \leq \mathsf{t}$. In the mean time, $\mathsf{Rep}(\mathsf{crs}, \widetilde{P}, w + \widetilde{\delta}) \neq \perp$ if and only if $\mathsf{dis}(\widetilde{w}, w + \widetilde{\delta}) \leq \mathsf{t}$ and $\widetilde{\sigma} = \widetilde{\sigma}'$ hold, where $\widetilde{w} \leftarrow \mathsf{SS.Rec}(w + \widetilde{\delta}, \widetilde{s})$ and $\widetilde{\sigma}' \leftarrow \mathsf{FEval}(F_{pk}, \tilde{t}, \tilde{t}', \widetilde{w})$. Therefore,

$$\begin{aligned} \Pr\left[\mathsf{win}_3 \wedge \neg \mathsf{bad}\right] &= \Pr\left[\begin{array}{c} \widetilde{P} \text{ is fresh } \wedge \ \mathsf{dis}(\widetilde{\delta}) \leq \mathsf{t} \ \wedge \\ \mathsf{dis}(\widetilde{w}, w + \widetilde{\delta}) \leq \mathsf{t} \ \wedge \ \widetilde{\sigma} = \widetilde{\sigma}' \ \wedge \ \neg \mathsf{bad} \end{array} \ \middle| \ \mathsf{G}_3 \right] \\ &\leq \Pr\left[\mathsf{dis}(\widetilde{w}, w + \widetilde{\delta}) \leq \mathsf{t} \ \wedge \ \widetilde{\sigma} = \widetilde{\sigma}' \ \wedge \ \neg \mathsf{bad} \ \middle| \ \mathsf{G}_3\right]. \end{aligned}$$

Now that bad does not occur, then the tag $\widetilde{\mathsf{tag}} = (\widetilde{t} = (\widetilde{s}, \widetilde{c}), \widetilde{t}')$ contained in $\widetilde{P}$ must be an injective tag. Thus $\mathsf{LAF}_{F_{pk},(t,t')}(\cdot)$ is injective and entropy preserving. This means $\widetilde{\sigma}' := \mathsf{FEval}(F_{pk}, \widetilde{t}, \widetilde{t}', \widetilde{W})$ has the same entropy as $\widetilde{W}$. Consequently, it will be hard for adversary $\mathcal{A}$ to forge a valid $\widetilde{\sigma}$ (i.e., $\widetilde{\sigma} = \widetilde{\sigma}'$) if $\widetilde{W}$ has enough min-entropy conditioned on $\mathcal{A}$'s view in G_3.

The outline of the proof is as follows.

- First, we prove that if $\mathsf{dis}(\widetilde{w}, w + \widetilde{\delta}) \leq \mathsf{t}$, then
$$\widetilde{H}_\infty(\widetilde{W} \mid \boxed{\mathsf{view}}) \geq \widetilde{H}_\infty(W \mid \boxed{\mathsf{view}}). \tag{24}$$
- Next, we show that
$$\widetilde{H}_\infty(W \mid \boxed{\mathsf{view}}) \geq \omega(\log \lambda). \tag{25}$$
- Formulas (24) and (25) give $\widetilde{H}_\infty(\widetilde{W} \mid \boxed{\mathsf{view}}) \geq \omega(\log \lambda)$.
  If the event bad does not happen, $(\widetilde{t}, \widetilde{t}')$ must be an injective tag, hence $\mathsf{LAF}_{F_{pk},(\widetilde{t},\widetilde{t}')}(\cdot)$ is an injective function, and $\widetilde{\sigma}' = \mathsf{FEval}(F_{pk}, \widetilde{t}, \widetilde{t}', \widetilde{W})$ preserves the entropy of $\widetilde{W}$. So we have
$$\Pr[\mathsf{win}_3 | \neg \mathsf{bad}] \leq \Pr\left[\mathsf{dis}(\widetilde{w}, w + \widetilde{\delta}) \leq \mathsf{t} \ \wedge \ \widetilde{\sigma} = \widetilde{\sigma}' \ \wedge \ \neg \mathsf{bad} \ \middle| \ \mathsf{G}_3\right] \leq 2^{-\omega(\log \lambda)}.$$

It remains to prove (24) and (25).

Proof of (24). Define the random variable $\widetilde{W} := \mathsf{SS.Rec}(\widetilde{s}, W + \widetilde{\delta})$, where W is the random variable in the robustness game. Let $w, \widetilde{w}$ denote the values taken by the random variables $W, \widetilde{W}$, respectively.
If $\mathcal{A}$ wins, then $\mathsf{dis}(w + \widetilde{\delta}, \widetilde{w}) \leq \mathsf{t}$. By Lemma 3, we have

$$\widetilde{H}_\infty\Big(\widetilde{W} \mid \big(\mathsf{SS.Gen}(W + \widetilde{\delta}), \boxed{\mathsf{view}}, \widetilde{\delta}\big)\Big) \geq \widetilde{H}_\infty\Big(W + \widetilde{\delta} \mid \big(\mathsf{SS.Gen}(W + \widetilde{\delta}), \boxed{\mathsf{view}}, \widetilde{\delta}\big)\Big). \tag{26}$$

Note that $\mathsf{SS.Gen}(W+\widetilde{\delta}) = \mathsf{SS.Gen}(W)+\mathsf{SS.Gen}(\widetilde{\delta})$. The sketch $s = \mathsf{SS.Gen}(W)$ belongs to $\boxed{\mathsf{view}}$, so $\mathsf{SS.Gen}(W+\widetilde{\delta})$ can be computed from $\boxed{\mathsf{view}}$ and $\widetilde{\delta}$. As a result, according to Eq. (1),

$$\widetilde{H}_\infty\Big(W+\widetilde{\delta} \mid \big(\mathsf{SS.Gen}(W+\widetilde{\delta}), \boxed{\mathsf{view}}, \widetilde{\delta}\big)\Big) = \widetilde{H}_\infty\Big(W+\widetilde{\delta} \mid \big(\boxed{\mathsf{view}}, \widetilde{\delta}\big)\Big). \quad (27)$$

Note that $\widetilde{\delta}$ is determined by $\mathcal{A}$ after seeing $\boxed{\mathsf{view}}$, therefore, it can be further eliminated from the condition because of Eq. (2), and we have

$$\widetilde{H}_\infty\Big(W \mid \big(\boxed{\mathsf{view}}, \widetilde{\delta}\big)\Big) = \widetilde{H}_\infty\Big(W \mid \boxed{\mathsf{view}}\Big). \quad (28)$$

With Eq. (27) and (28), we have

$$\widetilde{H}_\infty\Big(W+\widetilde{\delta} \mid \big(\mathsf{SS.Gen}(W+\widetilde{\delta}), \boxed{\mathsf{view}}, \widetilde{\delta}\big)\Big) = \widetilde{H}_\infty\Big(W \mid \boxed{\mathsf{view}}\Big). \quad (29)$$

Similarly, we have

$$\widetilde{H}_\infty\Big(\widetilde{W} \mid \big(\mathsf{SS.Gen}(W+\widetilde{\delta}), \boxed{\mathsf{view}}, \widetilde{\delta}\big)\Big) = \widetilde{H}_\infty\Big(\widetilde{W} \mid \boxed{\mathsf{view}}\Big). \quad (30)$$

Combining (26), (29) and (30), we have

$$\widetilde{H}_\infty(\widetilde{W} \mid \boxed{\mathsf{view}}) \geq \widetilde{H}_\infty(W \mid \boxed{\mathsf{view}}). \quad (31)$$

Proof of (25). The general idea of the proof is that we will, step by step, show that the view of adversary can be perfectly simulated by a simulator with $\mathcal{S}$ and s, where $\mathcal{S} := \{\sigma \mid \sigma = \mathsf{FEval}(F_{pk}, t, t', W) \wedge \mathsf{tag} = (t, t') \in \mathcal{T}_{lossy}\}$ and $s = \mathsf{SS.Gen}(W)$. By the lossiness of LAF, the information of W leaked by $\mathcal{S}$ is at most $\log p$ bits. By the fact that SS is a $(m - \lceil \log p \rceil, \hat{m}, 2\mathsf{t})$-secure sketch and the fact that $\hat{m} - \lceil \log p \rceil \geq \omega(\log \lambda)$, we have that $\widetilde{H}_\infty(W \mid \boxed{\mathsf{view}}) \geq \omega(\log \lambda)$. Details can be found in the full version [25]. ▌

Taking all things together, by Eq. (18) and Lemma 11-14, it follows that

$$\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{rrFE},\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF}}(1^\lambda) + \varepsilon_{\mathsf{ext}} + \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF}}(1^\lambda) + 2^{-\omega(\log \lambda)}. \quad ▌$$

Corollary 1. *If* SS *is instantiated by a syndrome-based secure sketch,* Ext *is instantiated as Eq. (5),* LAF *is instantiated with the scheme in [15], and* SKEM *is instantiated with the scheme shown in Fig. 3, then the construction in Fig. 4 results in a robustly reusable fuzzy extractor based on the DLIN assumption and the DDH assumption.*

Remark 4. Since there exist efficient linear error correcting codes which can correct linear fraction of errors, the syndrome-based secure sketch is able to correct linear fraction of errors as well, so is our robustly reusable fuzzy extractor.

Acknowledgements. We would like to thank the reviewers for their valuable comments. The authors are supported by the National Natural Science Foundation of China (NSFC No. 61672346).

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,Heidelberg (2017), https://doi.org/10.1007/978-3-319-60080-2_1
3. Bennett, C.H., DiVincenzo, D.P.: Quantum information and computation. Nature **404**(6775), 247–255 (2000)
4. 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
5. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.D.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) EUROCRYPT. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005), https://doi.org/10.1007/11426639_9
6. 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,Heidelberg (2016), https://doi.org/10.1007/978-3-662-49890-3_5
7. 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, Heidelberg (2008), https://doi.org/10.1007/978-3-540-78967-3_27
8. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1), 167–226 (2003), https://doi.org/10.1137/S0097539702403773
9. Daugman, J.: How iris recognition works. IEEE Trans. Circuits Syst. Video Techn. **14**(1), 21–30 (2004), https://doi.org/10.1109/TCSVT.2003.818350
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, Heidelberg (2006), https://doi.org/10.1007/11818175_14
11. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008), https://doi.org/10.1137/060651380
12. 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, Heidelberg (2004), https://doi.org/10.1007/978-3-540-24676-3_31
13. Dodis, Y., Wichs, D.: Non-malleable extractors and symmetric key cryptography from weak secrets. In: Mitzenmacher, M. (ed.) STOC 2009. pp. 601–610. ACM (2009), http://doi.acm.org/10.1145/1536414.1536496
14. Fuller, B., Meng, X., Reyzin, L.: Computational fuzzy extractors. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 174–193. Springer, Heidelberg (2013), https://doi.org/10.1007/978-3-642-42033-7_10
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, Heidelberg (2013), https://doi.org/10.1007/978-3-642-38348-9_31

16. Imamog, A., Awschalom, D.D., Burkard, G., DiVincenzo, D.P., Loss, D., Sherwin, M., Small, A., et al.: Quantum information processing using quantum dot spins and cavity qed. Physical Review Letters **83**(20), 4204 (1999)
17. Jain, A.K., Ross, A., Prabhakar, S.: An introduction to biometric recognition. IEEE Trans. Circuits Syst. Video Techn. **14**(1), 4–20 (2004), https://doi.org/10.1109/TCSVT.2003.818349
18. 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, Heidelberg (2008), https://doi.org/10.1007/978-3-540-85855-3_11
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. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) CCS 2010. pp. 237–249. ACM (2010), http://doi.acm.org/10.1145/1866307.1866335
22. Shoup, V.: A computational introduction to number theory and algebra. Cambridge University Press (2006)
23. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: DAC 2007. pp. 9–14. IEEE (2007), http://doi.acm.org/10.1145/1278480.1278484
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, Heidelberg (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. Designs Codes and Cryptography. (2018). https://doi.org/10.1007/s10623-018-0459-4