Revisiting Post-quantum Robustly Reusable Fuzzy Extractors

Somnath Panja and Reihaneh Safavi-Naini

University of Calgary, Canada

Abstract. Fuzzy extractors (FE) are cryptographic primitives that, for two samples w and w' of a randomness source, extract the same consistent, uniformly random key when the distance (in some metric space) between w and w' is less than a threshold. The FE's Gen algorithm takes w and generates a secret key R and a public value P that will be used by the Rep to recover R from w'. *Robustly reusable FEs* (rrFE) ensure security of R against an adversary that has access to multiple outputs of the Gen algorithm and its goal is to construct a P that is acceptable by Rep. rrFEs have found wide applications in biometric authentication and key establishment, in controlling access to digital wallets (including stored cryptocurrency private keys and digital identity credentials), and in device-to-device authentication and key establishment for IoT systems using Physically Unclonable Functions (PUFs).

In this paper, we revisit existing post-quantum rrFEs with proven security, analyze their models, and show that those following the established rrFE framework are either insecure (via concrete attacks) or require excessive entropy. We then propose an rrFE with proved security in standard model that satisfies all the desirable properties of rrFEs. We discuss our results and outline the open challenges.

Keywords: Fuzzy extractors · ring-LPN · post-quantum security.

1 Introduction

A *fuzzy extractor* (FE) is a cryptographic primitive that extracts a uniformly random key R from a sample w of a noisy entropy source modeled by a random variable W over a metric space. The key can later be recovered from a second sample w' drawn from the same distribution, provided that w' is *close* to w. Closeness is measured with respect to a distance function on the metric space, such as the Hamming distance, and is denoted by $\mathsf{dis}(w, w')$. Extracting randomness requires that W have sufficient entropy, where entropy is commonly measured using *min-entropy* (see section 2). *Information theoretically secure FEs* guarantee security against an adversary with unbounded computation [21,6,7,19,14] while *computationally secure FEs* assume an adversary with polynomially bounded (in the security parameter) computation [24,10,3,48,49].

FEs have been widely used for authentication and key establishment using biometric data [16,32,30,21,7,31,26,10,45] and Physical Unclonable Functions [46,40,43,27] with direct application to device authentication and key establishment. Other notable applications of FEs are biometric-based authentication in digital wallets to authorize transactions [47] or grant access to the wallet itself, and more complex blockchain applications, particularly in decentralized identity (DID) and self-sovereign identity [34].

An FE, in addition to an Init algorithm that initializes the system and generates the public parameter (before sampling the source), has a pair of algorithms: Gen takes a sample w of the source and outputs (R, P) where R is a uniformly random key, and P is the public *helper data*, and Rep that takes a sample w' that is close to w and the helper data P and reproduces R. Two important properties of FEs are *reusability* and *robustness*. Reusability guarantees security of the key when the adversary has access to the output of Gen on multiple samples and requires that giving (R_i, P_i), $i = 1 \cdots t$, $i \neq j$, and P_j to the adversary, does not allow them to distinguish R_j from uniformly random (see reusability in Definition 7). *Robustness* requires that any tempering with P be detectable by the Rep algorithm. An FE is *robustly reusable*, denoted by rrFE, if it is reusable and an adversary who sees $(R_i, P_i), i = 1 \cdots t-1$, cannot construct a P_t that is acceptable by the Rep algorithm (see Definition 7).

Entropy requirement. Information theoretically secure FEs have high entropy requirement. This is because helper data leaks information about the source sample and an rrFE requires entropy that is linear in the length of the key and the number of re-use, that is $m \in \Omega(|R|.t)$ where $|R|$ is the key length and t is the number of reuse. Computationally secure extractors use computational hardness assumptions to protect the source entropy leakage and their security guarantee is tied to the computational hardness of the underlying problem as well as the source entropy. Although, in theory, for a security level λ, the source W must have min-entropy m that is at least $\omega(\lambda)$, the best existing rrFE requires entropy that is linear in λ.

rrFE Security. Security of computational extractors have been formalized by two security games for reusability and robustness, and security proofs reduce security of the rrFE to the underlying computational assumption. *Post-quantum secure* rrFE rely on computational assumptions for which no efficient quantum algorithm is known, and so guarantee security even if there is a quantum computer. The reduction may be in *standard model* and directly reduce the security to the computational assumption, or in Random Oracle Model (ROM), relying on an abstract cryptographic object. Reductions in the *Quantum Random Oracle Model* (QROM) allow the adversary to make superposition queries to the random oracle, and hence model a more powerful adversary.

1.1 Our work

We make two types of contributions. On the negative side, we show all existing post-quantum rrFEs are either insecure, or have excessive entropy requirement, or do not follow the standard model of rrFEs (in particular, they have a weaker security model). On the positive side, we construct a post-quantum secure rrFE (and so the only rrFE system) with proved security in standard model, assuming the adversary has access to classical queries to the system and computation on a quantum computer. The entropy requirement of the construction matches the entropy requirement of most entropy efficient known FE. Table 1 summarizes our results.

FE Scheme	Entropy req.	Comp. Assmp.	PQ Sec.	Mod
WLG[50, Section 6.1]	$O(\lambda^2(\log(q))^3) + s$	LWE	✓	Std.
CLG [15, Fig. 18]**	$\Theta(\lambda^2) + s$	LPN	✓	Std.
DG [17, Section 3, 4]	$O(\lambda) + s$	Hash fun. intr.	✓	QROM
DG [17, Section 5]**	$O(\lambda) + s$ ($\approx 258\lambda + s$)	Ring-LPN	✓	Std.
ZLH [54, Fig. 13]	$O(\lambda) + s$	Isogeny	✓	Std.
Our FE	$O(\lambda) + s$ ($\approx 8\lambda + s$)	Ring-LPN	✓	Std.

Table 1. Existing (including our work) PQ secure rrFEs. Columns are as follows: *Entropy req.* : the minimum entropy requirement on source; *Comp. Assmp.:* The underlying computational assumption; *fun. intr*: function intractability; *PQ Sec.*: Post-quantum security; *Mod:* security proof model. *Std.*: standard model; *QROM.*: Quantum Random Oracle. λ is the security parameter, and s is the min-entropy loss of the source because of secure sketch. ✓ = property achieved; ✗= property is not provided; $\star\star$ We attack these constructions.

1. A critical review of existing post-quantum rrFEs. We analyze existing post-quantum secure rrFEs from the view point of security and entropy requirement, both directly affecting their usability in practice. All rrFEs, except the rrFE by Zhou et al. [54] follow the same model and security definition and so their security and entropy requirement are comparable. The rrFE in [54], however, significantly deviates from this model which makes the security properties not comparable. More details are given below.

Security. In section 3 and Appendix D, we present attacks on two post-quantum secure rrFE constructions in [15, Fig. 18] (DCC 2021) and [17, Section 5] (ACNS 2023), respectively. In Appendix E, we also present an attack on the rrFE construction of [15, Fig. 17] (DCC 2021), which has a similar structure and relies on the hardness of the Decisional Diffie–Hellman (DDH) assumption. Since DDH is not post-quantum secure (Shor's algorithm [44] efficiently solves the discrete logarithm problem), this construction is insecure in the post-quantum setting.

The attacked protocols follow the generic rrFE construction of Cui et al. [15, Fig. 8] and consist of a *homomorphic secure sketch with a linearity property* $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$, an *information-theoretic homomorphic extractor* Ext, and a *symmetric key encapsulation mechanism* (SKEM) $\mathsf{SKEM} = (\mathsf{SKEM.Encap}, \mathsf{SKEM.Decap})$ that satisfies two properties: *key-shift security* and *decapsulation uniformity* [15]. (For definition of primitives see section 2 and Appendix B.4.) The Init algorithm generates the public parameters of the rrFE scheme and the seed for the randomness extractor as a common reference string. The $\mathsf{rrFE.Gen}(w)$ function constructs helper data $s \leftarrow \mathsf{SS.Gen}(w)$ and the random key $k \hookleftarrow \mathsf{Ext}(w, r)$, where r is part of public random string CRS. It then uses k as the secret key in $\mathsf{SKEM.Encap}$ to obtain a random string K and the ciphertext $(K, C) \leftarrow \mathsf{SKEM.Encap}(k; r_{\mathsf{skem}})$, where r_{skem} is uniformly random. Finally K is parsed, $K = (K_1, K_2)$, to obtain the rrFE extracted key $R = K_2$. The rrFE helper data is $P := (s, C, K_1)$. The $\mathsf{rrFE.Rep}$ receives $P' := (s', C', K_1')$, recovers $\hat{w}$ from the sample w' and s' using $\mathsf{SS.Rec}(w', s')$, extracts $\hat{k}$ from $\hat{w}$ using Ext

and the CRS r and uses it in $\mathsf{SKEM.Decap}$ with C' to obtain $\hat{K}$. The $\hat{K}$ is parsed, $\hat{K} = (\hat{K}_1, \hat{K}_2)$, and if $K'_1 = \hat{K}_1$ holds, the $\mathsf{rrFE.Rep}$ accepts and outputs $R = \hat{K}_2$. The main difference in the constructions is in the construction of SKEM that relies on a computational assumption. Security proofs reduce rrFE security properties to this assumption.

Attacks. We use the bilinear (linearly in each input argument) property of XOR that makes the SKEM constructions *malleable*, that is an adversary can meaningfully alter the output to produce a *predictable change* in the decapsulated value without knowing the decapsulated value itself. This malleability allows an adversary to modify the helper data such that it passes the verification test of the $\mathsf{rrFE.Rep}$ algorithm and result in a key that is different from the key that is generated by the FE $\mathsf{rrFE.Gen}$ algorithm. The details of the attack in each case however is different and dependent on the actual computational assumption.

Cui et al. [15, Fig. 18]. The construction uses LPN (Learning Parity with Noise) as the underlying computational problem in the construction of SKEM. The attacks requires one query to the $\mathsf{rrFE.Gen}$ oracle (one helper data from the FE generation algorithm). We provide the attack in section 3.

Deo et al. [17, Section 5]. The construction uses *ring-LPN* as the underlying computational problem to improve the entropy requirement of the Cui et al. [15, Fig. 18] construction from $O(\lambda^2)$ to $O(\lambda)$. The construction can be broken using a single query to the $\mathsf{rrFE.Gen}$ oracle. We present the attack in Appendix D.

Cui et al. [15, Fig. 17]. The SKEM construction uses DDH assumption and it not post-quantum. In Appendix E we show an attack that breaks the algorithm and requires two queries to $\mathsf{rrFE.Gen}$.

Entropy requirement. A computational rrFE with proven security reduces its security to the hardness of an underlying computational problem instantiated with a uniformly random secret. The lower bound on the min-entropy of the source ensures that sufficient uniform randomness can be extracted for this secret, and naturally depends on the security parameter λ. In practice, for example, when protecting a cryptocurrency private key in a wallet using a biometric sample, one must ensure that the underlying source provides the required entropy. Two commonly used biometric sources are fingerprints and iris with entropy estimates $\sim$ 40 bits [41, Page 622][1] and $\sim$ 249 bits [16,45], respectively. To achieve λ bit security for the extracted key, the source must provide roughly (using an information theoretic argument outlined in section 4 and Lemma 1 in Appendix B.1) 2λ entropy. For $\lambda = 128$, biometric sources that come close to this bound include iris data and high-resolution fingerprint data. Thus, rrFEs with an $O(\lambda^2)$ entropy requirement are impractical when using widely deployed biometric sources, as the required entropy would be orders of magnitude higher than that of any commonly used biological source.

The entropy requirement of the known post-quantum rrFEs are summarized in Table 1. The constructions in [50, Section 6.1] and [15, Fig. 18] require

[1] This is a conservative estimate that can increase when higher-resolution fingerprint images are used.

$O(\lambda^2(\log(q))^3) + s$ and $O(\lambda^2) + s$ bit entropy, respectively, where s is the entropy loss of the secure sketch (*this notation will be used in the rest of the paper*) and $q \in \mathbb{N}$, making them impractical. The entropy requirements of the rrFE in [17, Section 3, 4], [17, Section 5], and [54, Fig. 13] is $O(\lambda) + s$, which makes them practically viable. The ring-LPN construction of Deo et al. [17, Section 5] requires min-entropy $m \approx 258\lambda + s$, which is linear in λ. However, because of the large constant factor (258), this linear requirement exceeds the value of the quadratic function λ^2 for commonly used security parameters (for example, for $80 \leq \lambda \leq 256$, we have $258\lambda > \lambda^2$). Thus, despite its linear asymptotic growth, the entropy requirement of the construction is very stringent in practice.

The security proofs in [17, Section 5] and [54, Fig. 13] are in the standard model and based solely on computational assumptions, while the scheme in [17, Section 3, 4] relies on Quantum Random Oracle Model (QROM) which is an abstract cryptographic object, and so a weaker theoretical guarantee. (We note that in practice a QROM is instantiated with a classical hash function.)

rrFE Model. For fair evaluation and comparison of rrFFs, we use the basic setup and definitions of rrFE in [49,50,15,17,37]. These constructions assume a public Common Random String (CRS) that is independent of the source that is generated during the system initialization.

Zhao et al. [54, Fig. 13] construction, however, deviates from this model and introduces an additional initialization step that is source dependent. In Section 4, we show that this new step, assumed to be executed only once and not accessible through the Gen and Rep oracles, results in a weaker security model, and therefore an incomparable construction. *To sum,* among the comparable post-quantum rrFEs (i.e., those following the same model), the existing ones are either insecure, or their entropy requirements exceed those of commonly used biometric sources, rendering them impractical.

2. A secure rrFE construction for $O(\lambda)$ min-entropy sources. We propose an rrFE construction for sources with min-entropy $m \in O(\lambda)$ using the following building blocks: a homomorphic secure sketch with linearity property (see definition 3 and section 2), a homomorphic (average-case) extractor, an instance of ring-LPN that is used to generated pseudorandom samples, a linear error correcting code (ECC) and a *key-manipulation secure message authentication code* (KMS-MAC) [14]. A KMS-MAC (Definition 4) provides security against an adversary who can tamper with a tagged message even if they are allowed to simultaneously tamper with the MAC key. Robustly reusable security of the construction is reduced to the hardness of the ring-LPN problem for which no quantum algorithm is known.

Construction outline. The Gen algorithm uses a homomorphic secure sketch with linearity property and a homomorphic (average-case) randomness extractor on the source sample w to generate s, that is used by secure sketch recovery algorithm, $\mathsf{SS.Rec}$, to recover w from w', and a uniformly random output string x that will serve as the secret in a ring-LPN instance, respectively. A sample of this instance will be used to "mask" the (error correcting encoded) final extracted key y_2, concatenated with the KMS-MAC key y_1, $c \leftarrow a \cdot x + e + \mathcal{E}(y1 \parallel y2)$,

a is the randomness of the sample and $\mathcal{E}$ denotes encoding using a linear error correcting code. The $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$ tag ensures integrity of the helper data given by $P = (s, a, c, \sigma)$.

The Rep algorithm receives $P' = (s', a', c', \sigma')$, possibly modified by the adversary. It takes the source sample w', recovers w using s and invokes the extractor Ext on w to obtain x'. It then recovers y'_1, y'_2 by removing the mask $a' \cdot x'$ and using the decoding function, $\mathcal{D}$, of the error correcting code. The final step is verifying the soundness using the verification function of the KMS-MAC.

1.2 Related Work

Post-quantum security. Our work focuses on post-quantum computational rrFEs and their entropy requirements.There is no post-quantum rrFE constructions with information theoretic security.

Post-quantum *reusable fuzzy extractor* (rFE) constructions have been constructed with computational [10,1,3,48] and information theoretic security [6]. Non-post-quantum rrFE constructions was first proposed in [49] using the common reference string model. The construction relies on the decisional Diffie-Hellman (DDH) problem and decisional linear (DLIN) problem. Other constructions of rrFF whose security rely on DDH assumption are in [50, Section 6.2], [15, Fig. 17] and [54, Fig. 14]. Table 1 provides a comparison of our rrFE with other existing post-quantum rrFEs. Table 2 in Appendix A compares our rrFE with a number of highly cited FEs, rFEs and rrFEs.

Entropy requirement. The constructions that are considered in this paper as well as the above rrFE constructions, are *for sources that have a lower-bound on their min-entropy.* This is the source model that was first used by Dodis et al. [21] for information-theoretic FEs and later widely used for both information theoretic [19,6,8,25,14,12] and computational [51,3,49,50,23,2,54,15,17,9,36,37] FEs.

Number of errors and structured sources. The number of error patterns within Hamming distance t grows as $B(n,t) = \sum_{i=0}^{i=t} \binom{n}{i}$, which depends on the source sample length n. Consequently, correcting the same number of errors in longer samples requires proportionally more redundancy, at least $\log |B(n,t)|$ parity bits (by the sphere-packing bound), leading to a corresponding reduction in effective min-entropy, and, once the entropy loss exceeds the available entropy budget, to information-theoretic impossibility results for fuzzy extractors, and severe limitations even in the computational setting. *Structured sources* have been introduced to overcome these limitations. In [10, Construction 1], sources with α-entropy k-samples are considered, meaning that a uniformly random subsample of k coordinates has min-entropy at least α, even though the global entropy rate may be low. Under suitable parameter instantiations, this assumption enables the construction of reusable fuzzy extractors with proved security in the random oracle model.

2 Preliminaries

We use λ to denote the security parameter. The probability distribution of a random variable (RV) X is denoted by $\mathsf{P}_X(x) = \Pr(X = x)$. The *min-entropy*

$H_\infty(X)$ of RV X over $\mathcal{X}$ is defined by $H_\infty(X) = -\log(\max_x(\mathrm{P}_X(x)))$. The *average conditional min-entropy* [21] of X given Y is $\tilde{H}_\infty(X|Y) = -\log \mathbb{E}_{y \leftarrow \mathcal{Y}} \max_{x \in \mathcal{X}} \mathrm{P}_{X|Y}(x|y)$. In this paper, we consider binary sources and take $\mathsf{dis}(w, w')$ to be the Hamming distance between w and w'. We also use $\mathsf{wt}(\delta)$ to denote the Hamming weight of a binary vector (or string) δ. F_2 represents the finite field with two elements, 0 and 1. Ber_τ denotes Bernoulli distribution with probability $\tau \in (0, \frac{1}{2})$. For a domain Y, $U(Y)$ denotes the uniform distribution over Y. When $\mathcal{X}$ is a set, $x \xleftarrow{\$} \mathcal{X}$ denotes x is sampled uniformly at random from the set $\mathcal{X}$. For a distribution D, we write $x \xleftarrow{\$} \mathsf{D}$ to denote that x is sampled according to the distribution D. The statistical distance between two RVs X and Y with the same domain $\mathcal{T}$ is defined by $\mathrm{SD}(X, Y) = \frac{1}{2} \sum_{v \in \mathcal{T}} |\Pr[X = v] - \Pr[Y = v]|$.

Extractor. Extractors are used to obtain uniformly random string from distributions with sufficient min-entropy.

Definition 1 (Strong (average case) extractor). $\mathsf{Ext} : \mathcal{X} \times \mathcal{I} \to \mathcal{Y}$ *is an average* $(\mathcal{X}, m, \mathcal{Y}, \epsilon)$*-extractor if for any two random variables* X *and* A*, where* $X \in \mathcal{X}$ *and* $\tilde{H}_\infty(X|A) \geq m$*, it holds that* $\mathsf{SD}(\mathsf{Ext}(X, I), I, A; U, I, A) \leq \epsilon$*, where* I *and* U *are uniformly random over* $\mathcal{I}$ *and* $\mathcal{Y}$ *respectively.*

Definition 2 (Homomorphic strong (average case) extractor). *An extractor* Ext *is homomorphic if* $\mathsf{Ext}(X_1 + X_2, I) = \mathsf{Ext}(X_1, I) + \mathsf{Ext}(X_2, I)$*, for any* $X_1, X_2 \in \mathcal{X}$ *and* $I \in \mathcal{I}$*.*

In Appendix B.1, we present an instantiation of a homomorphic average-case extractor.
A secure sketch [21] is a cryptographic primitive that produces public data, called a *sketch* s, about an input w, such that w can be recovered from any value w' that is close to w. A secure sketch must also minimize the information leaked about the input.

Definition 3 (Secure sketch). *An* $(\mathcal{M}, m, \tilde{m}, t)$*-secure sketch* (SS) *for metric space* $\mathcal{M}$ *with distance function* dis *consists of a pair of algorithms* $(\mathsf{SS.Gen}, \mathsf{SS.Rec})$*:*
$\mathsf{SS.Gen} : \mathcal{M} \to \mathcal{S}$*, that takes an input* $w \in \mathcal{M}$ *and outputs a sketch* $s \in \mathcal{S}$*.*
$\mathsf{SS.Rec} : \mathcal{M} \times \mathcal{S} \to \mathcal{M}$*, that takes* $w' \in \mathcal{M}$ *and a sketch* s *as input and outputs* $\tilde{w} \in \mathcal{M}$*, satisfying the following properties:*
Correctness. *If* $\mathsf{dis}(w, w') \leq t$ *and* $s = \mathsf{SS.Gen}(w)$*, then* $\mathsf{SS.Rec}(w', s) = w$*.*
Security. *For any random variable* W *over* $\mathcal{M}$*, if* $H_\infty(W) \geq m$*, then* $\tilde{H}_\infty(W|\mathsf{SS.Gen}(W)) \geq \tilde{m}$*.*

An $(\mathcal{M}, m, \tilde{m}, t)$*-secure sketch* (SS) *is homomorphic* if for all $w_1, w_2 \in \mathcal{M}$, $\mathsf{SS.Gen}(w_1, w_2) = \mathsf{SS.Gen}(w_1) + \mathsf{SS.Gen}(w_2)$.

An $(\mathcal{M}, m, \tilde{m}, t)$*-secure sketch* (SS) *is linear* [14] if for any $w \in \mathcal{M}$, $\tilde{s} \in \mathcal{S}$, and δ such that $\mathsf{wt}(\delta) \leq t$, let $s = \mathsf{SS.Gen}(w)$, $\tilde{w} = \mathsf{SS.Rec}(w + \delta, \tilde{s})$, and $\tilde{\delta} = \tilde{w} - w$, there exists a deterministic and efficient computable function f such that $\tilde{\delta} = \mathsf{f}(\delta, s, \tilde{s})$.

We use a homomorphic secure sketch with linearity property. In Appendix B.2, we recall an instantiation of homomorphic secure sketch from [14] with linearity property.

A message authentication code with key manipulation security ensures detection of any modification of a message and tag pair, even if an adversary can shift the key in addition of tampering with the message and tag pair.

Definition 4 (Message Authentication Code with Key Manipulation Security (KMS-MAC)). *[14, Definition 5] A $(\mathcal{K}, \mathcal{S}, \mathcal{T}, \epsilon)$-KMS-MAC consists of a MAC function* $\mathsf{MAC} : \mathcal{K} \times \mathcal{S} \to \mathcal{T}$ *that generates a tag $\sigma \in \mathcal{T}$ for a message $s \in \mathcal{S}$ under a secret key $k \in \mathcal{K}$, where $\mathcal{K}$ is a group, and verification function* $\mathsf{Ver} : \mathcal{K} \times \mathcal{S} \times \mathcal{T} \to \{0, 1\}$*, that take the key k and a message s' and a tag σ' and outputs* 0 *or* 1*, where* 0 *and* 1 *denote rejection and acceptance of a message and tag pair, respectively. A MAC algorithm is* correct *if for any $s \in \mathcal{S}$, we have:* $\Pr[k \leftarrow \mathcal{K}, \mathsf{Ver}(k, s, \mathsf{MAC}(k, s)) = 1] = 1$.
The KMS property requires, that for any $s \neq s' \in \mathcal{S}$, $\sigma, \sigma' \in \mathcal{T}$ and $\Delta \in \mathcal{K}$, it holds, $\Pr[\mathsf{Ver}(k + \Delta, s', \sigma') = 1 \mid \mathsf{MAC}(k, s) = \sigma] \leq \epsilon$*, where the probability is taken over the uniform distrubution of key $k \in_R \mathcal{K}$.*

In Appendix B.3, we give an instantiation of KMS-MAC that is used in our construction.

The ring-LPN problem is a computational problem for which no efficient quantum algorithm is currently known. It states that, for a computationally bounded adversary, it is hard to distinguish a sample generated from a ring-LPN distribution from a sample drawn uniformly over the same domain.

Definition 5 (Ring-LPN). *[28, Definition 1] For a polynomial $f(X)$ of degree n over F_2, ring $\mathsf{R} = \mathsf{F}_2[X]/f(X)$, and Bernoulli parameter $\tau \in (0, 1)$, the (decisional) ring-LPN distribution with secret $s \in \mathsf{R}$, denoted as $\mathsf{A}_\tau^{\mathsf{R},s}$, is a distribution over $\mathsf{R} \times \mathsf{R}$ whose samples are obtained as follows: sample $a \xleftarrow{\$} \mathsf{R}$, and $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$ (interpreted as a polynomial in R with each coefficient sampled from Ber_τ), and then output $(a, a \cdot s + e)$. The ring-LPN problem is ϵ_{RLPN}-hard if for every computationally bounded distinguisher $\mathcal{D}$ making polynomial number of queries,*

$$\left|\mathsf{Pr}[a \xleftarrow{\$} \mathsf{R} : \mathcal{D}^{\mathsf{A}_\tau^{\mathsf{R},s}} = 1] - \mathsf{Pr}[\mathcal{D}^{U(\mathsf{R}\times\mathsf{R})} = 1]\right| \leq \epsilon_{\mathsf{RLPN}}.$$

2.1 Fuzzy extractor

We use the definitional framework of Wen et al. [49] to define correctness, security, and reusability, and robustness of rrFE.

Definition 6 (Computational fuzzy extractor). *An* $(\mathcal{M}, m, \mathcal{R}, t, \epsilon)$-computational fuzzy extractor *for metric space $\mathcal{M}$ consists of three algorithms:*
$\mathsf{Init} : 1^\lambda \to \{0, 1\}^*$ *takes the security parameter λ as input and outputs a common reference string* crs. $\mathsf{crs} = \mathsf{Init}(1^\lambda)$.
$\mathsf{Gen} : \{0, 1\}^* \times \mathcal{M} \to \mathcal{R} \times \mathcal{P}$ *takes the common reference string* crs *and source*

sample $w \in \mathcal{M}$ *as input and outputs a key* $R \in \{0,1\}^\lambda$ *and a public helper data* $P \in \mathcal{P}$*.* $\mathsf{Gen}(\mathsf{crs}, w) = (R, P)$*.*
$\mathsf{Rep} : \{0,1\}^* \times \mathcal{M} \times \mathcal{P} \to \{\mathcal{R} \cup \perp\}$ *takes the common reference string* crs*, an element* $w' \in \mathcal{M}$ *and the helper data* $P \in \mathcal{P}$ *and outputs a key* R *or* $\perp$*.* $\mathsf{Rep}(\mathsf{crs}, w', P) = R/\perp$*.*

The correctness *and* security *of the extractor are defined as follows.*
Correctness*: For two samples,* w, w' *with* $\mathsf{dis}(w, w') \leq t$*, if* $\mathsf{crs} = \mathsf{Init}(1^\lambda)$ *and* $(R, P) = \mathsf{Gen}(W)$*, then* $\mathsf{Rep}(\mathsf{crs}, w', P) = R$ *where* $\mathsf{dis}(\cdot,\cdot)$ *is a distance measure.*
ϵ-Security*: For any distribution* W *over metric space* $\mathcal{M}$ *with* $H_\infty(W) \geq m$*, if* $\mathsf{crs} = \mathsf{Init}(1^\lambda)$*,* $(R, P) \leftarrow \mathsf{Gen}(crs, W)$*, then for all computationally bounded adversaries* $\mathcal{A}$*, we have:* $\big|\Pr[\mathcal{A}(\mathsf{crs}, P, R) = 1] - \Pr[\mathcal{A}(\mathsf{crs}, P, U) = 1]\big| \leq \epsilon$*, where* U *is a uniformly random over* $\mathcal{R}$*.*

Reusability of FE ensures security of the FE output key even if the same noisy source is used multiple times. Robustness provides security guarantee for the FE output against an adversary who can tamper with the helper data.

Definition 7 (Robustly reusable fuzzy extractor). *An* $\mathsf{rrFE} = (\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ *is an* $(\mathcal{M}, m, \mathcal{R}, t, \epsilon_{\mathsf{reu}}, \epsilon_{\mathsf{rob}})$-computational robustly reusable fuzzy extractor *(rrFE) if for all computationally bounded adversary* $\mathcal{A}$ *and for any distribution* W *over* $\mathcal{M}$ *such that* $H_\infty(W) \geq m$*, it holds that* $\mathsf{Adv}_{\mathcal{A}}(\mathsf{Exp}^{\mathsf{reu}}_{\mathcal{A},\mathsf{rrFE}}) = |\Pr[\mathsf{Exp}^{reu-0}_{\mathcal{A},\mathsf{rrFE}} = 1] - \Pr[\mathsf{Exp}^{reu-1}_{\mathcal{A},\mathsf{rrFE}} = 1]| \leq \epsilon_{\mathsf{reu}}$ *and* $\mathsf{Adv}_{\mathcal{A}}(\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}}) = \Pr[\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}} = 1] \leq \epsilon_{\mathsf{rob}}$*, where the experiment* $\mathsf{Exp}^{\mathsf{reu}}_{\mathsf{A},\mathsf{rrFE}}$ *and* $\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}}$ *are defined in Figure 1 (a) and Figure 1 (b) respectively.*

(a) $\underline{\mathsf{Exp}^{\mathsf{reu-b}}_{\mathcal{A},\mathsf{rrFE}}}$
1. $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$
2. $w \leftarrow W$
3. $(R^{(0)}, P^{(0)}) \leftarrow \mathsf{Gen}(w))$
4. $R^{(1)} \xleftarrow{\$} \mathcal{R}$
5. $b' \leftarrow \mathcal{A}^{O_{\mathsf{Gen}}}(\mathsf{crs}, R^{(b)}, P^{(0)})$
6. Return b'

$\underline{O_{\mathsf{Gen}}(\delta_i)}$
1. If $(\delta_i > t)$, Output $\perp$
2. Else,
3. $\quad (R_i, P_i) \leftarrow \mathsf{Gen}(\mathsf{crs}, w + \delta_i)$
4. $\quad$ Output (R_i, P_i)

(b) $\underline{\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}}}$
1. $\mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda)$
2. $w \leftarrow W$
3. $(R^{(0)}, P^{(0)}) \leftarrow \mathsf{Gen}(w))$
4. $Q := \{P^{(0)}\}$
5. $(\tilde{P}, \tilde{\delta}) \leftarrow \mathcal{A}^{O_{\mathsf{Gen}}}(\mathsf{crs}, P^{(0)}, R^{(0)})$
6. If $\mathsf{wt}(\tilde{\delta}) \leq t$, $\tilde{P} \notin Q$ and $\mathsf{Rep}(\mathsf{crs}, w + \tilde{\delta}, \tilde{P}) \neq \perp$
7. $\quad$ Output 1
8. Else,
9. $\quad$ Output 0

$\underline{O_{\mathsf{Gen}}(\delta_i)}$
1. If $(\mathsf{wt}(\delta_i) > t)$, Output $\perp$
2. Else,
3. $\quad (R_i, P_i) \leftarrow \mathsf{Gen}(\mathsf{crs}, w + \delta_i)$
4. $\quad Q := Q \cup \{P_i\}$
5. $\quad$ Output (R_i, P_i)

Fig. 1. Experiment (a) is used to define reusability of rrFE. Experiment (b) is used to define robustness of rrFE.

3 Revisiting the LPN based rrFE of Cui et al. [15]

In this section, we recall the robustly reusable FE construction of Cui et al. [15, Fig. 18] based on the hardness assumption of Learning Parity with Noise (LPN). The scheme is built from three components: a homomorphic linear secure sketch (for example, a syndrome-based secure sketch [20]), a linear randomness extractor and a symmetric key encapsulation mechanism (SKEM) that has a decapsulation uniformity property. Figure 4 in Appendix C depicts how these components are combined to construct the FE scheme.

Cui et al. [15, Section 5.3, Fig. 18] provided a construction for a robustly reusable FE that achieves post-quantum security. The security of the construction is based on the hardness of the LPN problem. For the sake of completeness, we describe the FE scheme almost verbatim. For completeness, the FE construction of Cui et al. [15, Fig. 18] is reproduced in Figure 2. The output of Init algorithm of the scheme including the seed for randomness extractor is in common reference staring.

$\underline{\mathsf{pp} \leftarrow \mathsf{Init}(1^\lambda):}$
$\tilde{\mathbf{R}} \leftarrow_\$ \mathbb{Z}_2^{m \times n}$.
Set $\mathfrak{n} = \lambda$.
Choose $\eta \in (0, 1/2)$.
Choose an $[\mathfrak{m}, \mathfrak{k}, d]$ code $\mathfrak{C}$
s.t. $\eta\mathfrak{m} < \mathfrak{t}$.
Set $\mathsf{pp}_{\mathsf{skem}} = (\mathfrak{C}, \mathfrak{n}, \eta)$
with $\mathfrak{C} = (\mathcal{E}, \mathcal{D})$.
Set $\mathsf{pp} := (\tilde{\mathbf{R}}, \mathsf{pp}_{\mathsf{skem}})$.
Return pp.

$\underline{(\mathsf{P}, \mathsf{R}) \leftarrow \overline{\mathsf{rrFE}}_{\mathsf{LPN}}.\mathsf{Gen}(\mathsf{pp}, \mathsf{w}; \mathsf{r}):}$
Parse $\mathsf{pp} = (\tilde{\mathbf{R}}, \mathsf{pp}_{\mathsf{skem}} = (\mathfrak{C}, \mathfrak{n}, \eta))$.
$\mathsf{s} \leftarrow \mathsf{syn}(\mathsf{w})$. // SS
Compute $\mathsf{k} = \tilde{\mathbf{R}} \cdot \mathsf{w}$. // $\overline{\mathsf{Ext}}$
Sample $\mathsf{r} \leftarrow_\$ \{0,1\}^{\mathfrak{k}+\mathfrak{n}+2\eta\mathfrak{m}\log\mathfrak{m}}$.
Transform k to matrix $\mathbf{X} \in \mathbb{Z}_2^{\mathfrak{n}\times\mathfrak{m}}$
Parse $\mathsf{r}^\top = (\mathsf{r}_1^\top, \mathsf{r}_2^\top, \mathsf{r}_3^\top)$
$\in \mathbb{Z}_2^{\mathfrak{k}} \times \mathbb{Z}_2^{\mathfrak{n}} \times \mathbb{Z}_2^{2\eta\mathfrak{m}\log\mathfrak{m}}$.
If $\mathsf{r}_2 = 0$, return $\perp$. // $\mathsf{SKEM}_{\mathsf{LPN}}$
Sample $\mathsf{v} \leftarrow \mathsf{Ber}_\eta^{\mathfrak{m}}(\mathsf{r}_3)$.
$\mathsf{y}^\top = \mathcal{E}(\mathsf{r}_1^\top) \oplus \mathsf{r}_2^\top \cdot \mathbf{X} \oplus \mathsf{v}^\top$.
$\mathsf{C} = (\mathsf{r}_2, \mathsf{y})$, $\mathsf{K} = \mathsf{r}_1^\top$.
Parse $\mathsf{K} = (\mathsf{K}_1, \mathsf{K}_2) \in \{0,1\}^{\omega(\log\lambda)} \times \{0,1\}^\lambda$.
Set $\sigma := \mathsf{K}_1$. $\mathsf{R} := \mathsf{K}_2$. $\mathsf{P} := (\mathsf{s}, \mathsf{C}, \sigma)$.
Return (P, R).

$\underline{\mathsf{R} \leftarrow \overline{\mathsf{rrFE}}_{\mathsf{LPN}}.\mathsf{Rep}(\mathsf{pp}, \mathsf{w}', \mathsf{P}):}$
Parse $\mathsf{P} = (\mathsf{s}, \mathsf{C}, \sigma)$.
$\tilde{\mathsf{w}} \leftarrow \mathsf{w}' - \mathsf{Decode}(\mathsf{syn}(\mathsf{w}') - \mathsf{s})$.
Compute $\tilde{\mathsf{k}} = \tilde{\mathbf{R}} \cdot \tilde{\mathsf{w}}$.
Transform $\tilde{\mathsf{k}}$ to matrix $\tilde{\mathbf{X}} \in \mathbb{Z}_2^{\mathfrak{n}\times\mathfrak{m}}$
Parse $\mathsf{C} = (\mathsf{r}_2, \mathsf{y})$.
If $\mathsf{r}_2 = 0$, return $\perp$.
$\tilde{\mathsf{K}} = \mathcal{D}(\mathsf{y}^\top \oplus \mathsf{r}_2^\top \cdot \tilde{\mathbf{X}})$.
Parse $\tilde{\mathsf{K}} = (\tilde{\mathsf{K}}_1, \tilde{\mathsf{K}}_2) \in \{0,1\}^{\omega(\log\lambda)} \times \{0,1\}^\lambda$.
Set $\tilde{\sigma} := \tilde{\mathsf{K}}_1$.
If $\tilde{\sigma} = \sigma$, $\tilde{\mathsf{R}} := \tilde{\mathsf{K}}_2$.
Else, $\tilde{\mathsf{R}} := \perp$.
Return $\tilde{\mathsf{R}}$.

Fig. 2. The LPN based robustly reusable FE scheme of Cui et al. [15], reproduced as verbatim for completeness.

3.1 Analysis of the FE scheme

An FE scheme is robust if it satisfies the robustness security in Definition 7. The construction in [15, Section 5.3, Fig. 18] is shown to be robust according to Definition 7. In this section, we prove that the construction is not robust. We prove this by showing that, given valid helper data generated by $\mathsf{rrFE.Gen}$ on a source w, an adversary can modify the helper data such that $\mathsf{rrFE.Rep}$ fails to detect the tempering of the helper data and produces a key that is different from the key generated by the $\mathsf{rrFE.Gen}$ algorithm. As a result of this attack, the sender that runs the $\mathsf{rrFE.Gen}$ algorithm and the receiver that executes the $\mathsf{rrFE.Rep}$ will have established two different keys. We provide the adversary's algorithm to perform the attack. Let $\mathcal{A}$ be an adversary attacking the robustness game Fig.1 (b) following Definition 7. The adversary runs the following algorithm.

Adversary $\mathcal{A}'$s Attack Algorithm: The algorithm has the following steps and succeeds with probability 1.

1. $\mathcal{A}$ queries the Gen oracle with an input δ, where $\mathsf{wt}(\delta) < t$.

2. Let the adversary receive a helper data (P, R) as the response to the query, where $(P, R) \leftarrow \mathsf{rrFE.Gen}(w + \delta, crs)$, $P = (s, C, \sigma)$, $C = (r_2, y)$ and $\sigma = K_1$ according to the $\mathsf{rrFE.Gen}$ algorithm in Figure 2. Upon receiving the helper data P, the adversary $\mathcal{A}$ chooses a $K'_2 \in \{0,1\}^\lambda$, computes $\mathcal{E}((0_\ell|K'_2)^T)$ and generates a forged helper data P', where $P' = (s, C', \sigma)$, $C' = (r_2, y \oplus y')$, $y'^T = \mathcal{E}((0_\ell|K'_2)^T)$, 0_ℓ denotes a sequence of ℓ number of 0s, and ℓ is the number of bits in K_1. $\mathcal{A}$ sends P' to the Rep oracle along with the input δ.

Note that

$$(y \oplus y')^T = y^T \oplus y'^T = \mathcal{E}({r_1}^T) \oplus r_2^T \cdot \mathbf{X} \oplus v^T \oplus \mathcal{E}((0_\ell|K'_2)^T) \tag{1}$$
$$= \mathcal{E}((K_1|K_2)^T) \oplus r_2^T \cdot \mathbf{X} \oplus v^T \oplus \mathcal{E}((0_\ell|K'_2)^T) \tag{2}$$
$$= \mathcal{E}((K_1|(K_2 \oplus K'_2))^T) \oplus r_2^T \cdot \mathbf{X} \oplus v^T \tag{3}$$

Equation 3 follows from the linearity of the error correcting code. Upon receiving P' and δ, the $\mathsf{rrFE.Rep}$ algorithm will output key $(K_2 + K'_2)$. This is because the adversary does not modify (s, σ), so the output of the Decode will be the same as the Decode's output that would have been if the helper data had not been altered. In addition, the verification of σ in the $\mathsf{rrFE.Rep}$ will succeed since the adversary does not change σ in the helper data and $\mathcal{D}((y + y')^T \oplus r_2^T \cdot \tilde{\mathbf{X}}) = (K_1, K_2 \oplus K'_2)$, and $\sigma = K_1$. Thus, the $\mathsf{rrFE.Gen}$ algorithm outputs K_2 as the extracted key R, but the $\mathsf{rrFE.Rep}$ algorithm produces $(K_2 + K'_2)$ as the extracted key. As a result, the robustness property does not hold.

Remark 1. The attack is shown for the LPN-based rrFE construction [15, Section 5.3, Fig. 18] that uses *perfect randomness*. It can be shown that the same attack also works for the LPN-based FE construction of Cui et al. [15, Section 5.3, Fig. 20], which is for *imperfect randomness*.

In Appendix D, we show an attack on the ring-LPN based rrFE construction of Deo et al. [17, Section 5], which proves the construction is not robust. In Appendix E, we present an attack on the DDH-based rrFE construction of Cui et al. [15, Fig. 17], which proves the scheme is not robust.

4 Entropy and rrFE Security Model

We evaluate rrFE schemes with proved security in terms of their entropy requirement, and security model.

Entropy. For an information-theoretic fuzzy extractor to securely extract a λ-bit key R that is statistically close to uniform from a noisy source W, the min-entropy m of W must be significantly larger than λ. This is because the fuzzy extractor leaks information through the public helper data P. The Leftover Hash Lemma 1 mandates that the conditional min-entropy must satisfy $\widetilde{H}_\infty(W|P) \geq \lambda - 2 - 2\log_2(\epsilon)$ to achieve a statistical distance of ϵ. A standard choice of $\varepsilon = 2^{-\lambda}$ for λ-bit security leads to the requirement $\widetilde{H}_\infty(W|P) \gtrsim 3\lambda$.

If we denote by ℓ an upper bound on the information leaked by the helper data P, then the initial min-entropy must satisfy $m \gtrsim 3\lambda + \ell$. In code-based instantiations, ℓ is determined by the redundancy needed to correct the noise level of the source and thus depends on the underlying error between source samples.

Achieving reusability and robustness in information-theoretic fuzzy extractors imposes additional entropy requirements. In particular, each reuse incurs entropy loss: correcting errors (e.g., via a secure sketch) leaks information about the source, and achieving robustness (e.g., through a MAC) requires additional entropy to withstand active attacks. Consequently, the total entropy requirement of the source grows approximately linearly with the number of allowed reuses. This stringent linear entropy barrier is a fundamental limitation of information-theoretic fuzzy extractors, and it renders them impractical for many real-world applications that rely on biometric sources with fixed and limited entropy. This shortcoming has motivated the development of computational fuzzy extractors, which achieve reusability and robustness with substantially lower entropy requirements by relying on cryptographic hardness assumptions rather than pure information-theoretic guarantees. (The increase is roughly, λ for adding robustness, and λ per each helper data in reusability.)

Computational fuzzy extractor. The following informal proposition shows that the min-entropy of the source must satisfy $m = \omega(\log(\lambda))$.

Proposition 1. *Let* $(\mathsf{Gen}, \mathsf{Rep})$ *be a computational fuzzy extractor for a source* W *with support* $\mathcal{W}$ *and min-entropy* $H_\infty(W) = m = O(log(\lambda))$*, where* λ *is the security parameter. Suppose that at least one of the following holds:*
(1) The helper data is a deterministic function of the secret. *There exists a function* f *such that for every* $w \in \mathcal{W}$*, the helper data output by* Gen *on input* w *is always* $P = f(w)$ *(*Gen *may still use randomness to choose* R*).*
(2) Efficient verification using Rep. *There exists public information* aux *(contained in* P*) such that, given any* $w' \in \mathcal{W}$*, the algorithm* $\mathsf{Rep}(w', P, \mathsf{aux})$ *behaves as follows:*
(i) If $\mathsf{dis}(w', w) \leq t$*, then* $\mathsf{Rep}(w', P, \mathsf{aux}) = R$ *with overwhelming probability.*
(ii) For other w'*, we have* $\mathsf{Rep}(w', P, \mathsf{aux}) = \perp$ *with overwhelming probability.*
Then, $(\mathsf{Gen}, \mathsf{Rep})$ *is not a secure computational fuzzy extractor.*

Proof (informal). Since $H_\infty(W) = m = O(\log(\lambda))$, the support $\mathcal{W}$ of W has size $|\mathcal{W}| \leq 2^m = \mathsf{poly}(\lambda)$. Thus, enumerating all $w \in \mathcal{W}$ is feasible in probabilistic polynomial time. Let $\mathsf{Gen}(w) = (R, P)$ for a sample w of W.
Case 1: deterministic helper data. Assume condition (1). For $w' \in \mathcal{W}$, the adversary computes $\hat{P} = f(w')$ and outputs the unique w' for which $\hat{P} = P$. Because P is a deterministic function of w, correctness of the scheme implies that, except with negligible probability, w is the only element of $\mathcal{W}$ that produces P and the adversary recovers w in polynomial time.
Case 2: verification using Rep. Assume condition (2) holds. The adversary enumerates all $w' \in \mathcal{W}$ and runs $\mathsf{Rep}(w', P)$ for each. If $\mathsf{Rep}(w', P)$ outputs $R' \neq \perp$, the adversary outputs R' (and identify the corresponding w' if needed). Else, it continues with another w'. By property (2), if $\mathsf{dis}(w', w) \leq t$ for the true w, we have $\mathsf{Rep}(w', P) = R$ with overwhelming probability. For all other $w' \in \mathcal{W}$, we get $\perp$ with overwhelming probability.

Therefore, with overwhelming probability, exactly one candidate in the polynomial size list yields a non- $\perp$ key, and that key is the true R. The adversary thus recovers R in probabilistic polynomial time. In either case, a polynomial-time adversary succeeds in recovering w or R. $\square$

However, this is only a minimal necessary condition. For practical λ-bit security, m should be linear in λ, typically $m \geq \lambda + k$ for some constant k. This ensures that brute-forcing the source ($\approx 2^m$) is at least as hard as breaking the underlying computational assumption ($\approx 2^\lambda$). While this entropy requirement ($m \approx \lambda$) is significantly lower than that of information-theoretic fuzzy extractors ($m \approx 2\lambda$), it still excludes very low-entropy sources such as fingerprints ($m \approx 40$) [41, Page 622]. The key enabling factor is the reliance on cryptographic assumptions, such as the hardness of learning with errors (LWE) problem [42]. These assumptions allow the construction to "amplify" the effective entropy of the source and provide security for longer keys, even when the initial entropy is small.

The entropy requirement of a fuzzy extractor depends on the source min-entropy and its structural properties, as well as on the specific computational assumption employed. Here, structure refers to the source's error and correlation model, which dictates the design of the error-correction component, while the computational assumption determines the parameter sizes of the construction and, critically, its post-quantum security.

4.1 Entropy requirement of rrFEs

In the following we will examine the entropy requirement of existing rrFEs that provide post-quantum security using the commonly used security model used in [49,50] and described in section 2 for security evaluation of our scheme. Security reduction in all cases (including our construction), except Deo et al. [17, Section 4, 3] is in standard model. In all cases, the constructions use a CRS (Common Reference String) which is uniformly random and is used as the uniform random seed of the randomness extractor.

Wen et al., PKC 2019 [50, Section 6.1]. Wen et al. [50, Section 4] gave a generic construction of rrFE that employs, a homomorphic secure sketch with linearity property, a homomorphic universal hash function, a pseudo-random function (PRF) with *unique-input key-shift secure PRF*. The instantiation of this PRF is based on LWE assumption. The output of the universal hash on the source sample w is represented as a $\lambda \log(q) \times \lambda \log(q)$ matrix with elements in $\mathbb{Z}_q$, $q \in \mathbb{N}$, and is used as the secret key of the PRF. Thus the PRF secret key has length $(\lambda \log(q))^2 \log(q)$ bits [50, Theorem 3], where $p, q \in \mathbb{N}$ such that $p|q$ and satisfies the relation in [50, Theorem 3]. Thus the minimum entropy requirement of the rrFE, assuming s bit entropy loss due to secure sketch for error correction, $O((\lambda \log(q))^2 \log(q)) + s$. More details is in Appendix F.

Cui et al., DCC 2021 [15, Fig. 18]. Cui et al. [15, Fig. 18] gave a generic construction of rrFE that replaces the need for perfect randomness with imperfect randomness, and has the following components: a homomorphic secure sketch with linearity property, a homomorphic strong seeded extractor, and a key-shift secure symmetric key encapsulation mechanism (SKEM). The construction is reproduced in Fig. 2 and used an SKEM instantiation that relies on LPN assumption. The key for the SKEM requires $\Theta(\lambda^2)$-bit [15, Corollary 2]. Assuming s bit entropy loss for the secure sketch is s bits, the rrFE entropy requirement is $\Theta(\lambda^2) + s$ bits.

Deo et al., ACNS 2023 [17, Section 3, 4, 5]. Deo et al. [17] gave two constructions of rrFE. An rrFE construction that is secure in QROM model and requires $O(\lambda) + s$ bits of entropy, where s is the entropy loss of secure sketch in the construction. The concrete entropy requirement is $\approx 4\lambda + s$ [17, Section 4.3] which is 512 for $\lambda = 128$, making it higher than what is offered by any commonly used biometric. The second construction [17, Section 5] improves the construction of rrFE of Cui et al. [15, Fig. 18] with the goal of reducing the required entropy. The construction uses an SKEM that relies on a ring-LPN instance, and is reproduced in Fig. 6. The SKEM secret key must have $O(\lambda)$ bit entropy, which for the ring-LPN-based construction, using parameters in [52], and used in [17, Section 5], requires W to have $\approx (258\lambda + s)$ bit entropy where s is entropy loss of secure sketch.

Our scheme. The ring-LPN based constructions of our generic construction of rrFE requires only $O(\lambda) + s$-bits of entropy, which is an improvement on the source entropy requirement of rrFE. More precisely, considering the current ring-LPN parameters [28], our rrFE requires $\approx 8\lambda + s$ bits entropy.

4.2 Zhou et al., Theoretical Comp. Sci. 2024 [54, Fig. 13].

Zhou et al. [54, Fig. 13] proposed an rrFE using an isogeny-based assumption, together with other building blocks, a homomorphic secure sketch with linearity property, a homomorphic average-case extractor, and a PRG that used the isogeny-based computational assumption. The entropy requirement of the construction is $O(\lambda) + s$ bits, where s is the entropy loss of the secure sketch.

Zhou et al. argue the challenge of constructing rrFE based on isogenies and propose "a slight modification to the traditional rrFE"[2] by adding an additional initialization algorithm that generates an "initial helper string P_{init}" by the user. Any future invocation of the Gen or Rep must take P_{init} as input. The construction, thus, is defined by four algorithms [54, Definition 16] that adds $\mathsf{rrFE.Init}$ to the definition of rrFE in [49,50,15] with the tuple of algorithms $(\mathsf{rrFE.Setup}, \mathsf{rrFE.Gen}, \mathsf{rrFE.Rep})$.

Similar to [49,50,15], the adversary can query $\mathsf{rrFE.Gen}$ and $\mathsf{rrFE.Rep}$ oracles, and wins under the same conditions (i.e. indistinguishability of the key for reusability, and inability to construct a helper data that is accepted by Rep for robustness- see Definition 7). The adversary however cannot query the initial helper string P_{init}.

Implications of this model is that the adversary cannot examine the full working of the Gen algorithm through *Gen* queries. Although it is easy to see that security in Cui et al. model implies security in Zhou et al. model by using a NULL string for P_{init}, establishing the relation between the two model remains an interesting open question. We note that the construction in Zhou et al. [54,

[2] Page 3, [54].

Fig. 13] only works for non-NULL P_{init} and so its security remains incomparable with other post-quantum rrFEs.

5 An rrFE Construction from Ring-LPN

(a) $\underline{\mathsf{rrFE.Init}(1^\lambda)}$

1. $r_{\mathsf{ext}} \xleftarrow{\$} \mathcal{S}$.
2. Choose a degree $n = \mathsf{Poly}(\lambda)$ polynomial f.
3. Set ring $\mathsf{R} = \mathsf{F}_2[X]/f(X)$.
4. For $\tau \in (0, \frac{1}{2})$, choose an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ with $n\tau \le t$.
5. Set $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$.
6. Output crs.

(b) $\underline{\mathsf{rrFE.Gen}(\mathsf{crs}, w)}$

1. Parse $\mathsf{crs} = (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$.
2. $s \leftarrow \mathsf{SS.Gen}(w)$.
3. $x \leftarrow \mathsf{Ext}(w, r_{\mathsf{ext}})$.
4. $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$; $y_2 \xleftarrow{\$} \mathcal{R}$; $a \xleftarrow{\$} \mathsf{R}$; $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$.
5. $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$.
6. $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$.
7. $P := (s, a, c, \sigma)$ and $R := y_2$.
8. Output (P, R).

(c) $\underline{\mathsf{rrFE.Rep}(\mathsf{crs}, P', w')}$

1. Parse $\mathsf{crs} = (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$.
2. Parse $P' = (s', a', c', \sigma')$.
3. $\tilde{w} \leftarrow \mathsf{SS.Rec}(w', s')$.
4. If $\mathsf{dis}(\tilde{w}, w') > t$,
5. Output $\perp$.
6. Else,
7. $x' \leftarrow \mathsf{Ext}(r_{\mathsf{Ext}}, \tilde{w})$.
8. $k' \leftarrow \mathcal{D}(c' + a' \cdot x')$.
9. Parse $k' = (y_1', y_2') \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$.
10. $\sigma' \leftarrow \mathsf{MAC}(y_1', (s', a', c'))$.
11. If $(\sigma = \sigma')$,
12. Output y_2'.
13. Else,
14. Output $\perp$.

Fig. 3. An rrFE scheme based on ring-LPN

We describe the construction in Figure 3. The construction uses the following building blocks:

– A homomorphic $(\mathcal{M}, m, \tilde{m}, t')$-secure sketch $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Rec})$ with linearity property for a metric space $\mathcal{M}$.
– A homomorphic average-case $(\mathcal{M}, \tilde{m}, \mathcal{X}, \epsilon_{\mathsf{ext}})$-extractor Ext.
– An $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$, where $k = \ell_{\mathcal{K}_{\mathsf{MAC}}} + \ell_{\mathcal{R}}$ and $n, \ell_{\mathcal{K}_{\mathsf{MAC}}}, \ell_{\mathcal{R}}$ are the bit-length of each element in $\mathcal{X}$, $\mathcal{K}_{\mathsf{MAC}}$ and $\mathcal{R}$ respectively.
– A key-manipulation secure $(\mathcal{K}_{\mathsf{MAC}}, \mathcal{S}_{\mathsf{MAC}}, \mathcal{T}_{\mathsf{MAC}}, \epsilon_{\mathsf{MAC}})$ massage authentication code (KMS-MAC) MAC.

Construction Outline. The $\mathsf{rrFE.Init}$ on input the security parameter λ generates the public parameters including the seed for extractor and description of the ring R and the Bernoulli parameter τ, description of the error correcting code $(\mathcal{E}, \mathcal{D})$ as a common reference string.

The $\mathsf{rrFE.Gen}$ algorithm applies a secure sketch generation algorithm and a randomness extractor on a source sample w generating s and a uniform key x. It then samples two random keys y_1 and y_2, computes $\mathcal{E}(y_1 \parallel y_2)$ and masks it with a ring-LPN sample using x as the secret key, generates $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, where a is uniformly random from a ring R and e is sampled from the Bernoulli distribution. and computes an authentication tag σ using a KMS-MAC algorithm using y_1 as the secret key, and outputs (s, a, c, σ) as helper data and y_2 the extracted key.

The $\mathsf{rrFE.Rep}$ algorithm takes a source sample w' and helper data (s', a', c', σ') as input and recovers $\tilde{w}$ using w' and s', uses the randomness extractor on $\tilde{w}$ to regenerate x', removes the mask $a' \cdot x'$ and recovers $y'_1 \parallel y'_2$ by computing $\mathcal{D}(c' + a' \cdot x')$ using the decoding algorithm, $\mathcal{D}$. It verifies the authentication tag σ' using the KMS-MAC algorithm with y'_1 as the secret key and outputs y'_2 as the extracted key if the verification succeeds.

Theorem 1. *The construction in Figure 3 is an* $(\mathcal{M}, m, \mathcal{R}, t', \epsilon_{\mathsf{reu}}, \epsilon_{\mathsf{rob}})$*-robustly reusable fuzzy extractor, where* $\epsilon_{\mathsf{reu}} = 2\epsilon_{\mathsf{ext}} + 2\epsilon_{\mathsf{RLPN}}$ *and* $\epsilon_{\mathsf{rob}} = \epsilon_{\mathsf{ext}} + \epsilon_{\mathsf{RLPN}} + \epsilon_{\mathsf{MAC}}$.

Proof Outline. We present an outline of the proof. The complete proof is in Appendix G.

Reusability. The proof consists of a sequence of games: G_0 to G_6.
Game G_0 is described in experiment $\mathsf{Exp}^{\mathsf{reu}-0}_{\mathcal{A},\mathsf{rrFE}}$ (Fig. 1 (a) in Definition 7).
Game G_1 is the same as game G_0, except in response to the ith generation query where $\mathcal{A}$ provides the shift δ_i, we use the homomorphic property of secure sketch (Section 2) to rewrite $\mathsf{SS.Gen}(w + \delta_i)$ as $\mathsf{SS.Gen}(w) + \mathsf{SS.Gen}\delta_i$, and the linearity property of extractor Ext to write $\mathsf{Ext}(w + \delta_i, r_{\mathsf{ext}})$ as $\mathsf{Ext}(w, r_{\mathsf{ext}}) + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$. We prove that $\Pr[G_0 \Rightarrow 1] = \Pr[G_1 \Rightarrow 1]$.
Game G_2 is the same as game G_1 except that the secret key of ring-LPN (that is the output of $\mathsf{Ext}(w, r_{\mathsf{ext}})$) is replaced by a uniformly sampled key from $\mathcal{X}$. We prove that $|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \epsilon_{\mathsf{ext}}$.
Game G_3 is the same as game G_2 except that the KMS-MAC secret key and R in line 6 and line 7 of $\mathsf{rrFE.Gen}$ are replaced with uniformly chosen values from respective domains (these were values used in computation of c in line 5 of $\mathsf{rrFE.Gen}$). We show that $|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}$.
Game G_4 is the same as game G_3 except that the secret key for KMS-MAC is the same y_1 that is being used in the computation of c in line 5 of $\mathsf{rrFE.Gen}$ instead of uniformly chosen from $\mathcal{K}_{\mathsf{MAC}}$. We then prove that
$|\Pr[G_3 \Rightarrow 1] - \Pr[G_4 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}$.
Game G_5 is the same as G_4 except that the generation of the secret key for ring-LPN, x, is reverted back to $x \leftarrow \mathsf{Ext}(w, r_{\mathsf{ext}})$. We show that
$|\Pr[G_4 \Rightarrow 1] - \Pr[G_5 \Rightarrow 1]| \leq \epsilon_{\mathsf{ext}}$.
Game G_6 is the same as game G_5 except that we revert back the conceptual changes done in the generation of (P_i, R_i) in game G_1. We show that
$\Pr[G_5 \Rightarrow 1] = \Pr[G_6 \Rightarrow 1]$. Combining these games, we prove that the FE construction is reusable.

Robustness. The proof of robustness also uses a sequence of games: G_0 to G_3.
Game G_0 is the robustness experiment $\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}}$ (Fig. 1 (b) in Definition 7).
Game G_1 is the same as game G_0 except in response to the ith generation query where $\mathcal{A}$ provides the shift δ_i, we use the homomorphic property of secure sketch (Section 2) to rewrite $\mathsf{SS.Gen}(w+\delta_i)$ as $\mathsf{SS.Gen}(w)+\mathsf{SS.Gen}(\delta_i)$, and the linearity property of extractor Ext to write $\mathsf{Ext}(w + \delta_i, r_{\mathsf{ext}})$ as $\mathsf{Ext}(w, r_{\mathsf{ext}}) + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$, and in response to the Rep oracle query with a shift δ^* and helper data $P^* := (s^*, a^*, c^*, \sigma^*)$, we use the linearity property of the secure sketch and the linearity property of extractor to write $\tilde{w} \leftarrow \mathsf{SS.Rec}(w + \delta^*, s^*)$ and $\tilde{x} \leftarrow \mathsf{Ext}(\tilde{w}, r_{\mathsf{Ext}})$ as

$\tilde{\tilde{\delta}} = \mathsf{f}(\delta^*, s, s^*)$ and $\tilde{x} = \mathsf{Ext}(w, r_{ext}) + \mathsf{Ext}(\tilde{\tilde{\delta}}, r_{\mathsf{ext}})$. We prove that
$\Pr[G_0 \Rightarrow 1] = \Pr[G_1 \Rightarrow 1]$.
Game G_2 is the same as game G_1 except that the secret key for ring-LPN is uniformly and independently sampled from the secret key space $\mathcal{X}$ of ring-LPN instead of being the output of $\mathsf{Ext}(w, r_{\mathsf{ext}})$. We then prove that
$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \epsilon_{\mathsf{ext}}$.
Game G_3 is the same as game G_2 except that the secret key for KMS-MAC and R are uniformly random in $\mathcal{K}_{\mathsf{MAC}}$ and $\mathcal{R}$, respectively, instead of the same $(y_1 \parallel y_2)$ that is being used in the ring-LPN-based computation $(a \cdot x + e + \mathcal{E}(y_1 \parallel y_2))$, and in the reproduction query, if a^* in the query is the same as one of the Gen oracle query's output, it selects the previously used key that was used for MAC in that query and then shifts it with a value $\varDelta$ known to the adversary before using it as the secret key for MAC for verifying the tag. We show that $|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}$, and from the key manipulation security of KMS-MAC, we have $|\Pr[G_3 \Rightarrow 1]| \leq \epsilon_{\mathsf{MAC}}$. Combining these results, we prove robustness. □

5.1 Concrete instantiation.

We instantiate the construction in Fig. 3, with security parameter $\lambda = 128$ (λ is the size of the FE output key). We first obtain the source parameters (including its entropy requirement) and then determine the remaining parameters of the building blocks.
(a) *Determining source parameters.*
We consider a source W with length $\bar{n}$, min-entropy m and $\mathsf{dis}(w, w') \leq t'$, and aim to estimate m for given $\bar{n}$ and t'. We (i) choose parameters of ring-LPN, (ii) select a concrete error correcting code $(\mathcal{E}, \mathcal{D})$, and (iii) obtain input entropy of W from the required Ext and secure sketch SS min-entropy.

- The Ext output, x, is the ring-LPN secret key x. The ring-LPN sample $b = a \cdot x + e$ has error vector e, drawn from a noise distribution $\mathsf{Ber}_\tau^{\mathsf{R}}$. We use a ring-LPN [28] instance with secret size 768 bits and the *Bernoulli parameter* $\tau = .125$ that is shown to provide 113-bit ($\epsilon_{\mathsf{RLPN}} = 2^{-113}$) security (for LPN [4], [35, Section 5.2]). The outline of the argument is in Remark 2.
- For $\lambda = 128$, key requirement of the system $(y_1 \parallel y_2)$ is $(128 + 2 \times 128) = 384$ bits. Thus, $\mathcal{E}$ must correct 12.5% error of the ring-LPN on the code block length of (at least) 768 bits. In Remark 3, we show that a $[1200, 384, 301]$ Low Density Parity Check (LDPC) [39,38,33] code where $301 = 2 \times .125 \times 1200 + 1$ minimum distance, is a good choice for $(\mathcal{E}, \mathcal{D})$.
- The Ext has output length of 1200 bits and so taking into account the entropy loss of Ext (using Left-over hash lemma 1 and $\epsilon_{\mathsf{ext}} = 2^{-128}$), the min-entropy of input W, must be $1200 + 2 \times 128 - 2 = 1454$.
- The rrFE input W is of length $\bar{n}$ and $\mathsf{dis}(w, w') \leq t'$. Using a syndrome-based secure sketch [21] based on an $[\bar{n}, \bar{k}, 2t' + 1]$ error correcting code requires a secure sketch output of length $(\bar{n} - \bar{k}) \geq 2t' + 1$, and so the min-entropy of W must satisfy $m \geq 1454 + (\bar{n} - \bar{k}) \geq 1454 + 2t' + 1$.

(b) rrFE *building blocks.*

- A homomorphic Syndrome-based $(\mathcal{M}, m, \tilde{m}, t')$- secure sketch ($\mathsf{SS}$) with linearity property using an $[\bar{n}, \bar{k}, 2t'+1]$-linear error correcting code, where $\mathcal{M} = \{0,1\}^{\bar{n}}$, $\tilde{m} = m - (\bar{n} - \bar{k})$.
- An $(\mathcal{M}, \tilde{m}, \{0,1\}^{m-(\bar{n}-\bar{k})+2-2\lambda}, 2^{-\lambda})$-randomness extractor ($\mathsf{Ext}$) using Toeplitz matrix $r_{\mathsf{ext}} \in \mathbb{Z}_2^{(m-(\bar{n}-\bar{k})+2-2\lambda)\times\bar{n}}$ in Appendix B.1. Ext output x has statistical distance $2^{-\lambda}$ from uniform.
- We use a $(\mathcal{K}_{\mathsf{MAC}}, \mathcal{S}_{\mathsf{MAC}}, \mathcal{T}_{\mathsf{MAC}}, \epsilon_{\mathsf{MAC}})$-KMS-MAC with $\mathcal{K}_{\mathsf{MAC}} = \mathsf{F}^2$, $\mathcal{S}_{\mathsf{MAC}} = \mathsf{F}^{\lceil \frac{\bar{n}-\bar{k}+1200}{128} \rceil}$ and $\mathcal{T}_{\mathsf{MAC}} = \mathsf{F}$ where $\mathsf{F} = \mathsf{GF}(2^{128})$ and $\epsilon_{\mathsf{MAC}} = 2^{-128}$, which is negligible. The MAC construction is given in Appendix B.3.

Entropy requirement comparison. For above parameter choices, our rrFE construction requires $\approx 12\lambda + (\bar{n} - \bar{k})$, which is an improvement over the ring-LPN based rrFE construction of Deo et al. [17, Section 5], which needs $258\lambda + (\bar{n} - \bar{k})$ min-entropy.

Remark 2. We use the results of [4], [35, Section 5.2], [22] for solving LPN problem to choose parameters of ring-LPN [28] instance used in our construction. The paper analysis is based on attacks that recover the LPN secret. In using the result in our setting we implicitly assume that the computational hardness of recovering LPN secret is similar to the indistinguishability of ring-LPN samples from uniform. According to the analysis of [4], [35, Section 5.2], [22], an LPN problem with secret key size 768 bits and the *Bernoulli parameter* $\tau = .125$ would require 2^{113} memory (and thus at least that much time) to solve when given approximately that many query access. For higher 131 bits security (2^{-131}-hard), secret size 1024-bit with the same *Bernoulli parameter* can be used.

Remark 3. For error probability .125, using Shannon's limit for binary symmetric channel, the achievable information rate is $\leq (1 - \mathsf{H}_2(.125)) \approx .46$, where H_2 is the binary entropy function. This means that for dimension (k) 384, we have the code block length (n) ≥ 840. Thus, a $[1200, 384, 2t+1]$ Low Density Parity Check (LDPC) [39,38,33] code that can correct 12.5% error is a suitable candidate for $(\mathcal{E}, \mathcal{D})$.

6 Conclusion

Biometric authentication is considered the gold standard of identification when secure access to extremely valuable data such as those stored in cryptocurrency or identity wallets are required. The primary cryptographic tools for using biometric data in authentication and key extraction are fuzzy extractors, and in practice their reusable and robust variants (rrFEs). We provided a critical assessment of existing post-quantum–secure rrFE constructions with 'proved' security, examining both their formal security guarantees and their applicability to commonly used biometric sources. We then proposed a new construction which, to the best of our knowledge, is the only rrFE offering post-quantum security in the standard model while requiring only $O(\lambda)$ source entropy.

For $\lambda = 128$, none of the constructions (including ours) can be used with a single (commonly used) biometric source. Reducing entropy requirement of rrFEs and/or using rrFE with multiple biometric sources are interesting directions for future research.

References

1. Q. Alamélou, P.-E. Berthier, C. Cachet, S. Cauchie, B. Fuller, P. Gaborit, and S. Simhadri. Pseudoentropic isometries: A new framework for fuzzy extractor reusability. In *AsiaCCS 2018*, pages 673–684, 2018.
2. D. Apon, C. Cachet, B. Fuller, P. Hall, and F.-H. Liu. Nonmalleable digital lockers and robust fuzzy extractors in the plain model. In S. Agrawal and D. Lin, editors, *Advances in Cryptology – ASIACRYPT 2022*, pages 353–383, Cham, 2022. Springer Nature Switzerland.
3. D. Apon, C. Cho, K. Eldefrawy, and J. Katz. Efficient, reusable fuzzy extractors from LWE. In *CSCML 2017, Proc.*, volume 10332 of *LNCS*, pages 1–18, 2017.
4. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, July 2003.
5. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key homomorphic prfs and their applications. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 410–428, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
6. X. Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the 11th ACM Conf. on Computer and communications security*, pages 82–91, 2004.
7. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In *Proc. EUROCRYPT 2005*, pages 147–163, 2005.
8. X. Boyen, Q. Mei, and B. Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. In *Proc. 12th ACM Conf. - CCS '05*, page 320–329, 2005.
9. C. Cachet, A. Hamlin, M. Rezapour, and B. Fuller. Upgrading fuzzy extractors. In C. Pöpper and L. Batina, editors, *Applied Cryptography and Network Security*, pages 156–182, Cham, 2024. Springer Nature Switzerland.
10. R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith. Reusable fuzzy extractors for low-entropy distributions. In *Annual Int. Conf. Theory Appl. Cryptographic Techniques*, pages 117–146. Springer, 2016.
11. R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. D. Smith. Reusable fuzzy extractors for low-entropy distributions. *J. Cryptol.*, 34(1):2, 2021.
12. K. Chen, P. Shen, K. Lv, and C. Chen. Secure sketch and fuzzy extractor with imperfect randomness: An information-theoretic study. In *Information and Communications Security*, pages 128–147, Cham, 2022. Springer International Publishing.
13. Cramer, Dodis, Fehr, Padró, and Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Advances in Cryptol. - EUROCRYPT 2008*.
14. R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. *IACR Cryptol. ePrint Arch.*, page 30, 2008.
15. N. Cui, S. Liu, D. Gu, and J. Weng. Robustly reusable fuzzy extractor with imperfect randomness. *Des. Codes Cryptogr.*, 89(5):1017–1059, 2021.
16. J. Daugman. How iris recognition works. *IEEE Trans. Circuits Syst. Video Technol.*, 14(1):21–30, 2004.
17. A. Deo and C. Grover. Robustly reusable fuzzy extractors in a post-quantum world. In M. Tibouchi and X. Wang, editors, *Applied Cryptography and Network Security*, pages 429–459, Cham, 2023. Springer Nature Switzerland.
18. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In M. Abe, editor, *Advances in Cryptology -*

ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, 2010.
19. Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. *IEEE Trans. Inf. Theory*, 58(9):6207–6222, 2012.
20. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
21. Y. Dodis, L. Reyzin, and A. D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*, pages 523–540, 2004.
22. A. Esser, R. Kübler, and A. May. LPN decoded. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 486–514. Springer, 2017.
23. H. Feng and Q. Tang. Computational robust (fuzzy) extractors for crs-dependent sources with minimal min-entropy. In *Theory of Cryptogr. Conf.*, pages 689–717. Springer, 2021.
24. B. Fuller, X. Meng, and L. Reyzin. Computational fuzzy extractors. In *Advances in Cryptol. - ASIACRYPT 2013*, LNCS, pages 174–193, 2013.
25. B. Fuller, L. Reyzin, and A. Smith. When are fuzzy extractors possible? *IEEE Transactions on Information Theory*, 66(8):5282–5298, 2020.
26. F. Hao, R. Anderson, and J. Daugman. Combining crypto with biometrics effectively. *IEEE Transactions on Computers*, 55(9):1081–1088, 2006.
27. C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.
28. S. Heyse, E. Kiltz, V. Lyubashevsky, C. Paar, and K. Pietrzak. Lapin: An efficient authentication protocol based on ring-lpn. In A. Canteaut, editor, *Fast Software Encryption*, pages 346–365, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
29. R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-Random Generation from One-Way Functions. In *STOC '89*, pages 12–24. ACM Press, 1989.
30. M. M. Islam, R. Safavi-Naini, and M. Kneppers. Scalable behavioral authentication. *IEEE Access*, 9:43458–43473, 2021.
31. A. Juels and M. Sudan. A fuzzy vault scheme. *Des. Codes Cryptogr.*, 38(2):237–257, 2006.
32. N. Karakaya, G. I. Alptekin, and Ö. D. Incel. Using behavioral biometric sensors of mobile phones for user authentication. *Procedia Computer Science*, 159:475–484, 2019.
33. F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.
34. J. Lee, J. Hwang, J. Choi, H. Oh, and J. Kim. SIMS : Self sovereign identity management system with preserving privacy in blockchain. *IACR Cryptol. ePrint Arch.*, page 1241, 2019.
35. É. Levieil and P.-A. Fouque. An improved lpn algorithm. In *Security and Cryptography for Networks*, pages 348–359. Springer, 2006.
36. Y. Li, S. Liu, D. Gu, and K. Chen. Reusable Fuzzy Extractor Based on the LPN Assumption. *The Computer Journal*, 63(12):1826–1834, 06 2020.

37. Y. Ma, P. Shen, X. Tian, K. Lv, and C. Chen. Reusable and robust fuzzy extractor for crs-dependent sources. *Cybersecur.*, 8(1):4, 2025.
38. D. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, 1999.
39. D. MacKay and R. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 32:1645–1646, 1996.
40. M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure pufs. In *2008 IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 670–673. IEEE, 2008.
41. N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, 2001.
42. O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM*, 56(6):1–40, sep 2009.
43. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, page 237–249, New York, NY, USA, 2010. Association for Computing Machinery.
44. P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, oct 1997.
45. S. Simhadri, J. Steel, and B. Fuller. Cryptographic authentication from the iris. In *ISC 22, 2019*, volume 11723, pages 465–485, 2019.
46. G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conf.*, pages 9–14, 2007.
47. M. A. Ulak, C. Giousouf, E. Benli, I. Engin, and S. Bahtiyar. Biowallet: A biometric digital wallet. In *Proceedings of the 12th International Conference on Systems (ICONS 2017)*, pages 23–27, Venice, Italy, April 2017. Thinkmind.
48. Y. Wen and S. Liu. Reusable fuzzy extractor from LWE. In *Inf. Secur. and Priv. - 23rd Australasian Conf., ACISP 2018*, volume 10946 of *LNCS*, pages 13–27.
49. Y. Wen and S. Liu. Robustly reusable fuzzy extractor from standard assumptions. In *ASIACRYPT 2018*, volume 11274 of *LNCS*, pages 459–489. Springer.
50. Y. Wen, S. Liu, and D. Gu. Generic constructions of robustly reusable fuzzy extractor. In *Public-Key Cryptography – PKC 2019*, page 349–378. Springer-Verlag, 2019.
51. Y. Wen, S. Liu, and S. Han. Reusable fuzzy extractor from the decisional diffie-hellman assumption. *Des. Codes Cryptogr.*, 86(11):2495–2512, 2018.
52. Y. Yu and J. Zhang. Lepton: Lpn-based kems with post-quantum security. https://csrc.nist.gov/CSRC/media/Presentations/Lepton/images-media/Lepton-April2018.pdf, 2018. `https://csrc.nist.gov/CSRC/media/Presentations/Lepton/images-media/Lepton-April2018.pdf`.
53. M. Zhandry. The magic of elfs. *J. Cryptol.*, 32(3):825–866, July 2019.
54. Y. Zhou, S. Liu, and S. Han. Robustly reusable fuzzy extractor from isogeny. *Theoretical Computer Science*, 1008:114677, 2024.

A Related work

Table 2 summarizes a comparison of our rrFE with a number of existing highly cited FEs, rFEs and rrFEs.

FE Scheme	Sec.	Comp. Assmp.	Reus	Rob	PQ Sec.	Mod
DRS04[21]	IT.	-	✗	✗	✓	Std.
Boyen04[6]	IT.	-	✓	✗	✓	Std.
Boyen04[6]	IT.	-	✓	✗	✓	RO
BDKOS05[7]	IT.	-	✗	✓	✓	RO
DKK12[19],CDF08[13]	IT.	-	✗	✓	✓	Std.
FMR13[24]	Comp.	LWE	✗	✗	✓	Std
CFP16[11],ABC18 [1]	Comp.	Hash fun. intr.	✓	✗	✓	Dig. Loc.
ACEK17[3]	Comp.	LWE	✓	✗	✓	Std.
WLH18[51]	Comp.	DDH	✓	✗	✗	Std.
WL18[48]	Comp.	LWE	✓	✗	✓	Std.
WL[49]	Comp.	DDH, DLIN	✓	✓	✗	Std.
WLG[50, Section 6.1]	Comp.	LWE	✓	✓	✓	Std.
LLGC [36]	Comp.	LPN	✓	✗	✓	Std.
FT [23]	Comp.	SXDH and exp. k-Linear	✗	✓	✗	Std.
CLG [15, Fig. 18]**	Comp.	LPN	✓	✓	✓	Std.
CLG [15, Fig. 17]**	Comp.	DDH	✓	✓	✗	Std.
ACFHL [2]	Comp.	Group Theoretic assmp.	✗	✓	✗	Std.
CSLC [12]	IT.	-	✗	✗	✓	Std.
DG [17, Section 3, 4]	Comp.	Hash fun. intr.	✓	✓	✓	QROM
DG [17, Section 5]**	Comp.	Ring-LPN	✓	✓	✓	Std.
ZLH [54, Fig. 13]	Comp.	Isogeny	✓	✓	✓	Std.
Our FE	Comp.	Ring-LPN	✓	✓	✓	Std.

Table 2. Comparison of our rrFE with other FEs. IT. and Comp. denote information-theoretic and computational security, respectively. Columns are as follows: Sec: whether the scheme achieves IT.Sec. or Comp. Sec.; Comp. Assmp.: The underlying computational assumption; Reus : the scheme is reusable; Rob :the scheme is robust; PQ Sec. - Post-quantum security; Mod - security proof model. Std.: standard model, ✓: feature is achieved, ✗: feature is not provided; fun. intr: function intractability; SXDH: symmetric external Diffie-Hellman assumption [18]; exp. k-Linear: exponentially hard k-linear assumption [53]; Group Theoretic assmp.: Group Theoretic assumption [2, Assumption 3.1]; RO: Random Oracle; Dig. Loc.: requires Digital Locker; QROM: Quantum random oracle model. ** We attack these constructions.

B Preliminaries

B.1 Extractor

A *universal hash family* can be used as a randomness extractor whose parameters satisfies the Leftover Hash Lemma [29].

Definition 8 (Universal hash family). *A family of functions* $h : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ *is called a* universal hash family *if* $\forall x, y \in \mathcal{X}$, $x \neq y$, *then we have that* $\Pr[h(x, S) = h(y, S)] \leq \frac{1}{|\mathcal{Y}|}$, *where the probability is taken over the uniform choices of* S *in* $\mathcal{S}$.

Lemma 1 (Generalized Leftover Hash Lemma [20]). *Let* $h : \mathcal{X} \times \mathcal{S} \to \{0,1\}^\ell$ *be a universal hash family. Then for any two random* *variables $A \in \mathcal{X}$ and $B \in \mathcal{Y}$, applying h on A can extract an almost uniform random variable whose length ℓ satisfies the following:*

$$\Delta(h(A,S),S,B;U_\ell,S,B) \leq \frac{1}{2}\sqrt{2^{-\tilde{H}_\infty(A|B)} \cdot 2^\ell},$$

where S is chosen uniformly at random from $\mathcal{S}$. Especially, if h is a universal hash family with $\mathcal{Y} = \{0,1\}^\ell$ satisfying $\ell \leq m + 2 - 2\log\frac{1}{\epsilon}$, then h is an $(\mathcal{X}, m, \mathcal{Y}, \epsilon)$-extractor.

In particular, let S be the set of all Toeplitz matrix A in $\mathbb{Z}_2^{\ell\times n}$ of the form

$$A := \begin{bmatrix} a_0 & a_{n-1} & a_{n-2} & \cdots & a_1 \\ a_1 & a_0 & a_{n-1} & \cdots & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_3 \\ \vdots & & & \ddots & \vdots \\ a_{\ell-1} & a_{\ell-2} & \cdots & \cdots & a_\ell \end{bmatrix}$$

where $k \leq m+2-2\log(\frac{1}{\epsilon})$, $W = \{0,1\}^n$. Then $\mathsf{Ext}(w, A) = A \cdot w$ is a $(\mathcal{X}, m, \mathcal{Y}, \epsilon)$-extractor [51, Section 2], where $\mathcal{X} = \{0,1\}^n$, $\mathcal{Y} = \{0,1\}^\ell$.

B.2 Secure sketch

For an $[n, k, 2t+1]$ linear error correcting code with parity check matrix $H \in \mathbb{Z}_2^{(n-k)\times n}$, the syndrome-based secure sketch consists of the following algorithms.

- $\mathsf{SS.Gen}(w)$: For an input $w \in \{0,1\}^n$, it outputs $s = \mathsf{syn}(w) = H \cdot w$.
- $\mathsf{SS.Rec}(w', s)$: $w' - \mathsf{Decode}(\mathsf{syn}(w') - s)$.

This secure sketch is homomorphic as $\mathsf{SS.Gen}(w+\delta) = H\cdot(w+\delta) = H\cdot w + H\cdot\delta = \mathsf{SS.Gen}(w) + \mathsf{SS.Gen}(\delta)$.

It also has linearity property [14] as

$$\tilde{\delta} = \tilde{w} - w = \mathsf{SS.Rec}(w', \tilde{s}) - w = w' - \mathsf{Decode}(\mathsf{syn}(w') - \tilde{s}) - w \quad (4)$$

$$= \delta - \mathsf{Decode}(\mathsf{syn}(w') - \tilde{s}) = \delta - \mathsf{Decode}(s + \mathsf{syn}(\delta) - \tilde{s}). \quad (5)$$

We define $\mathsf{f}(\delta, s, \tilde{s}) = (\delta - \mathsf{Decode}(s + \mathsf{syn}(\delta) - \tilde{s}))$ and $\tilde{\delta} = \mathsf{f}(\delta, s, \tilde{s})$.

B.3 Message authentication code with key manipulation security

We recall an instantiation of KMS-MAC provided in [14,19]. Define $\mathsf{MAC} : (GF(2^n))^2 \times (GF(2^n))^d \to GF(2^n)$ as follows:

$$\mathsf{MAC}((k,k'),\mathbf{m}) = k^{d+2} + \sum_{i=1}^{d} m_i k^i + k',$$

where $k, k' \in GF(2^n)$ and $\mathbf{m} = (m_1, \cdots, m_d) \in (GF(2^n))^d$. The security of the key manipulation secure MAC (KMS-MAC) [14] is $\epsilon_{\mathsf{MAC}} = \frac{d+2}{2^n}$, which is negligible.

B.4 Symmetric key encapsulation mechanism.

We recall the definition of symmetric key encapsulation mechanism [49] and its decapsulation uniformity property [15, Definition 16].

Definition 9 (Symmetric key encapsulation mechanism). *A symmetric key encapsulation mechanism* (SKEM) *with decapsulation uniformity consists of three algorithms* $\mathsf{SKEM.Setup}$*,* $\mathsf{SKEM.Encap}$ *and* $\mathsf{SKEM.Decap}$*. Let* $\mathcal{K}'$*,* $\mathcal{R}_e$*,* $\mathcal{C}$*,* $\mathcal{K}$ *be the secret key space, randomness space for* $\mathsf{SKEM.Encap}$*, ciphertext space and encapsulation key space respectively. The algorithms satisfy the the following properties.*

- $\mathsf{SKEM.Setup}(1^\lambda) \to \mathsf{pp}$ *that takes a security parameter* λ *as input and outputs public parameters* pp.
- $\mathsf{SKEM.Encap}(k; r) \to (K, C)$ *takes the secret key* k *as input, samples* $r \xleftarrow{\$} \mathcal{R}_e$ *and outputs en encapsulated key* K *and a ciphertext* C*.*
- $\mathsf{SKEM.Decap}(k, C) \to K'$ *takes the secret key* k *and a ciphertext* C *and outputs a key* $K' \in \mathcal{K}$ *or a reject symbol* $\perp$*.*

Correctness. *If* $\mathsf{pp} \leftarrow \mathsf{SKEM.Setup}(1^\lambda)$*,* $(K, C) \leftarrow \mathsf{SKEM.Encap}(k; r)$*, then it satisfies that* $\mathsf{SKEM.Decap}(k, C) = K$*.*
Key-shift pseudorandomness. *For all computationally bounded adversary* $\mathcal{A}$*, we have*

$$\Pr[b' = b : \mathsf{pp} \leftarrow \mathsf{SKEM.Setup}(1^\lambda), k \xleftarrow{\$} \mathcal{K}', b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{A}^{O_b^{\mathsf{ksp}}}(\mathsf{pp})] = \mathsf{negl}(\lambda),$$

where oracle O_b^{ksp} *is defined below.*
Decapsulation uniformity. *For any* $C \in \mathcal{C}$*, and* $K' \in \mathcal{K}$*, we have that*

$$\Pr[\mathsf{SKEM.Decap}(k, C) = K' : \mathsf{pp} \leftarrow \mathsf{SKEM.Setup}(1^\lambda), k \xleftarrow{\$} \mathcal{K}'] = \frac{1}{|\mathcal{K}|}.$$

The oracle O_b^{ksp} *is defined as follows.*

- $O_b^{\mathsf{ksp}}(\delta_i)$*: On input* δ_i*, if* $b = 0$*, it outputs uniformly random* $(K_i, C_i) \xleftarrow{\$} \mathcal{K}' \times \mathcal{C}$*. Otherwise, it outputs* $(K_i, C_i) \leftarrow \mathsf{SKEM.Encap}(k + \delta_i; r_i)$*.*

C Revisiting the LPN based robustly reusable FE scheme of Cui et al. [15, Fig. 18]

Figure 4 depicts how SS, Ext and SKEM are combined to construct the generic rrFE scheme of Cui et al. [15].

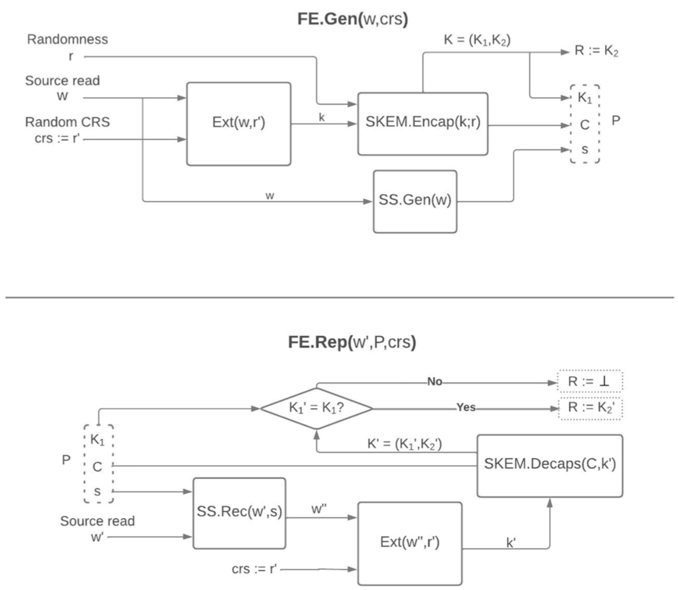

Fig. 4. A diagram of the robustly reusable FE scheme of Cui et al. [15], reproduced for completeness. SS is a homomorphhic sketch with linearity property, Ext is an homomorphic strong randomness extractor and SKEM is a symmetric key encapsulation mechanism.

D Revisiting the post-quantum standard model FE construction of Deo et al. [17, Section 5]

D.1 The FE scheme

Deo et al. [17, Section 5] presented an improvement to the robustly reusable FE construction of Cui et al. based on LPN. Similar to Cui et al.'s construction, Deo et al.'s construction has three building blocks: a linear homomorphic secure sketch, a randomness extractor and an SKEM. Figure 4 shows how these building blocks are combined to construct the FE algorithm. To instantiate, the construction however replaces the LPN based SKEM of Cui et al. with a Ring-LPN based SKEM, which improves the performance of the FE. For the sake of completeness, we first recall the LPN-based SKEM from Cui et al. [15], and then Ring-LPN based SKEM from Deo et al. [17, Section 5].

Figure 5 depicts the $\mathsf{SKEM}_{\mathsf{LPN}}$ scheme of Cui et al. and is reproduced as verbatim for completeness. It uses the following notations.

$[m, k, d]$ is a linear error correcting code $\mathcal{C}$ with encoding and decoding algorithms $(\mathcal{E}, \mathcal{D})$ respectively. $\mathsf{Ber}_\eta^m(r)$ takes a parameter r and outputs a vector of m binary elements, each drawn from a Bernoulli distribution with parameter η (i.e. the probability of 1 is η).

$\mathsf{pp_{skem}} \leftarrow \mathsf{SKEM.Setup}(1^\lambda)$: Set $n = \mathsf{poly}(\lambda)$. Choose $\eta \in (0, 1/2)$. Choose an $[m, k, d]$ code $\mathfrak{C} = (\mathcal{E}, \mathcal{D})$ s.t. $\eta m < t$. Set $\mathsf{pp_{skem}} = (\mathfrak{C}, n, \eta)$. Return $\mathsf{pp_{skem}}$.	$(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{SKEM.Encap}(\mathbf{X} \in \mathbb{Z}_2^{n \times m}; \mathsf{r})$: Parse $\mathsf{r}^\top = (\mathsf{r}_1^\top, \mathsf{r}_2^\top, \mathsf{r}_3^\top)$ $\in \mathbb{Z}_2^k \times \mathbb{Z}_2^n \times \mathbb{Z}_2^{2\eta m \log m}$. If $\mathsf{r}_2 = 0$, return $\perp$. $\mathsf{v} \leftarrow \mathsf{Ber}_\eta^m(\mathsf{r}_3)$. $\mathsf{y}^\top = \mathcal{E}(\mathsf{r}_1^\top) \oplus \mathsf{r}_2^\top \cdot \mathbf{X} \oplus \mathsf{v}^\top$. $\mathsf{C} = (\mathsf{r}_2, \mathsf{y})$, $\mathsf{K} = \mathsf{r}_1^\top$. Return (K, C).	$\mathsf{K}' \leftarrow \mathsf{SKEM.Decap}(\mathbf{X}, \mathsf{C})$: Parse $\mathsf{C} = (\mathsf{r}_2, \mathsf{y})$. If $\mathsf{r}_2 = 0$, return $\perp$. $\mathsf{K}' = \mathcal{D}(\mathsf{y}^\top \oplus \mathsf{r}_2^\top \cdot \mathbf{X})$. Return K'.

Fig. 5. $\mathsf{SKEM_{LPN}}$ scheme (from LPN) of Cui et al. [15], reproduced for completeness.

Deo et al. noted that, assuming n and m are $O(\lambda)$, the secret key of the $\mathsf{SKEM_{LPN}}$ has $O(\lambda^2)$ bits. Thus, to use this $\mathsf{SKEM_{LPN}}$, the randomness extractor in Figure 4 needs to extract $O(\lambda^2)$ uniform bits from the source. Therefore, the source needs to have $O(\lambda^2)$ bits of min-entropy. Deo et al. improved the rrFE algorithm by using a ring-LPN based SKEM that requires $O(\lambda)$ bits secret key.

Figure 6 describes the SKEM scheme of Deo et al. [17].

(a) $\mathsf{SKEM_{RLPN}.Setup}(1^\lambda)$ Choose a degree $n = \mathsf{poly}(\lambda)$ function f. Set $R = \mathbb{Z}[X]/f(X)$ and $R_2 = R/2R$. Choose κ and $m = \mathsf{poly}(\lambda)$ such that $\kappa n = m$ for some $\kappa \in \mathbb{Z}$. For $\tau \in (0, \frac{1}{2})$, choose an $[m, k, d = 2t+1]$ code $(\mathcal{E}, \mathcal{D})$ such that $m\tau \le t$.	(b) $\mathsf{SKEM_{RLPN}.Encap}(X)$ Parse $X = (x_1, x_2)$, where $x_1 \in R_2^\kappa$ and $x_2 = \mathbb{Z}_2^k$. Sample randomly $a \leftarrow_R R_2$. Sample $e \leftarrow \mathsf{Ber}_\tau^{\kappa \cdot n}$ $\tilde{x} \leftarrow \mathbb{Z}_2^k$ Output $C = (a, a \cdot x_1 \oplus e \oplus \mathcal{E}(\tilde{x}))$ and $K = x_2 \oplus \tilde{x}$.	(c) $\mathsf{SKEM_{RLPN}.Decap}(X, C)$ Parse C as $(c_1, c_2) \in R_2 \times R_2^\kappa$ Parse X as $(x_1, x_2) \in R_2^\kappa \times \mathbb{Z}_2$. Output $\mathcal{D}(c_2 - c_1 x_1) \oplus x_2$

Fig. 6. $\mathsf{SKEM_{RLPN}}$ scheme of Deo et al. [17].

D.2 Analysis of the FE scheme

Deo et al. proved that the construction of FE in Figure 4 with the ring-LPN based SKEM in Figure 6 is a robustly reusable FE according to Definition 7. We however prove that the construction is not robust. We prove this by showing that, after receiving a helper data in response to a single query to the $\mathsf{FE.Gen}$, an adversary can modify the helper data such that the $\mathsf{FE.Rep}$ will fail the tempering of the helper data, and generate a key which is different from the extracted key of the sender. As a result of this attack, the $\mathsf{FE.Gen}$ and the $\mathsf{FE.Rep}$ will have different extracted keys. We describe the adversaries algorithm to perform the attack. Let $\mathcal{A}$ be an adversary attacking the robustness game in Fig. 1 following Definition 7.

Adversary $\mathcal{A}$'s Attack Algorithm: The algorithm has two steps.

1. $\mathcal{A}$ queries the Gen oracle with an input δ, where $dis(\delta) < t$.
2. Let $\mathcal{A}$ receive a helper data (P, R) as a response to the query, where $(P, R) \leftarrow \mathsf{FE.Gen}(w + \delta, crs)$, $P = (s, C, \sigma)$, $C = (a, a \cdot x_1 \oplus e \oplus \mathcal{E}(\tilde{x}))$, and $\sigma = (x_2 \oplus \tilde{x})_{\lambda+1 \cdots \lambda+\ell}$ according to $\mathsf{FE.Gen}$ in Figure 4 and ring-LPN based SKEM

in Figure 6, and ℓ is the number of bits in K_1 (i.e. tag). Upon receiving the helper data P, $\mathcal{A}$ chooses a $x_3 \in \{0,1\}^\lambda$, computes $\mathcal{E}(0_\ell, x_3)$, where 0_ℓ denotes a string of ℓ number 0s. $\mathcal{A}$ then generates a new helper data P', where $P' = (s, C', \sigma)$, $C' = (a, a \cdot x_1 \oplus e \oplus \mathcal{E}(\tilde{x}) \oplus \mathcal{E}(0_\ell | x_3))$. $\mathcal{A}$ sends P' to the Rep oracle along with the input δ.

Notice that

$$a \cdot x_1 \oplus e \oplus \mathcal{E}(\tilde{x}) \oplus \mathcal{E}(0_\ell | x_3) = a \cdot x_1 \oplus e \oplus \mathcal{E}(\tilde{x} \oplus (0_\ell | x_3)) \tag{6}$$

Equation 6 follows from the linearity of error correcting code. Upon receiving P' and δ, $\mathsf{FE.Rep}$ will output $(x_2 \oplus \tilde{x})_{1 \cdots \lambda} \oplus x_3$ as the key. To describe the reason, we explain the execution of $\mathsf{FE.Rep}$ in more detail below. As the adversary does not modify (s, σ), the output of $\mathsf{SS.Rec}(w', s)$ and $\mathsf{Ext}(w'', r')$ in $\mathsf{FE.Rep}$ algorithm in Figure 4 will be the same as their respective output that would have been if the helper data had not been modified. As a result, $\mathcal{D}(a \cdot x_1 \oplus e \oplus \mathcal{E}(\tilde{x}) \oplus \mathcal{E}(0_\ell | x_3) \oplus a \cdot x_1) = (\tilde{x} \oplus (0_\ell | x_3))$. Therefore, the output of $\mathsf{SKEM}_{\mathsf{RLPN}}.\mathsf{Decap} = (x_2 \oplus \tilde{x} \oplus (0_\ell | x_3))$. So, in Figure 4, $K' = (K_1', K_2') = (x_2 \oplus \tilde{x} \oplus (0_\ell | x_3))$. The $\mathsf{FE.Rep}$ then computes $K_1' = (x_2 \oplus \tilde{x} \oplus (0_\ell | x_3))_{\lambda+1 \cdots \lambda+\ell} = (x_2 \oplus \tilde{x})_{\lambda+1 \cdots \lambda+\ell} = \sigma = K_1$. Therefore, in Figure 4, the verification of $K_1' = K_1$ will succeed. Hence, $\mathsf{FE.Rep}$ will output a key $K_2' = (x_2 \oplus \tilde{x} \oplus (0_\ell | x_3))_{1 \cdots \lambda}$, which is different from $\mathsf{FE.Gen}$ algorithm's output key $K_1 = (x_2 \oplus \tilde{x})_{1 \cdots \lambda}$.

E Revisiting the DDH based robustly reusable FE construction of Cui et al. [15, Fig. 17]

E.1 The FE scheme

As depicted in Figure 4, the DDH based robustly reusable FE scheme in Cui et al. [15, Section 5.3, Fig. 17] also consists of three building blocks: a homomorhic linear secure sketch, a randomness extractor and an SKEM. The DDH based FE scheme uses the same cryptographic tools to realize the homomorhic linear secure sketch and the randomness extractor that are used for the LPN based FE scheme. It however uses a DDH based scheme to realize the SKEM component. The robustly reusable DDH based FE construction of Cui et al. [15, Section 5.3, Fig. 17] is given in Figure 7, reproduced as verbatim for completeness. The security of the construction is based on the DDH assumption. The output of the Init algorithm of the construction is in common reference string.

E.2 Analysis of the FE scheme

The construction is proved to be robust in Cui et al. [15, Section 5.3, Fig. 17] according to the robustness game in Fig. 1 (b) following Definition 7. We however prove that the construction is not robust. We prove this by showing that after making two queries to the Gen oracle, an adversary can generate a new helper data that will be accepted by the FE Rep algorithm. As a result of the attack,

$\underline{\mathsf{pp} \leftarrow \mathsf{Init}(1^\lambda)}$: $\tilde{\mathbf{R}} \leftarrow_\$ \mathbb{Z}_2^{m\times n}$. $(\mathbb{G}, q, g) \leftarrow \mathsf{GGen}(1^\lambda)$. $\mathsf{pp}_{\mathsf{skem}} := (\mathbb{G}, q, g)$. Set $\mathsf{pp} := (\tilde{\mathbf{R}}, \mathsf{pp}_{\mathsf{skem}})$. Return pp.	$\underline{(\mathsf{P}, \mathsf{R}) \leftarrow \overline{\mathsf{rrFE}}_{\mathsf{DDH}}.\mathsf{Gen}(\mathsf{pp}, \mathsf{w}; \mathsf{r})}$: Parse $\mathsf{pp} = (\tilde{\mathbf{R}}, \mathsf{pp}_{\mathsf{skem}} = (\mathbb{G}, q, g))$. $\mathsf{s} \leftarrow \mathsf{syn}(\mathsf{w})$. // SS Compute $\mathsf{k} = \tilde{\mathbf{R}} \cdot \mathsf{w}$. // $\overline{\mathsf{Ext}}$ Sample $\mathsf{r} \leftarrow_\$ \mathbb{Z}_q^*$. $\boxed{\mathsf{C} = g^{\mathsf{r}}, \mathsf{K} = \mathsf{C}^{\mathsf{k}}.}$ // $\mathsf{SKEM}_{\mathsf{DDH}}$ Parse $\mathsf{K} = (\mathsf{K}_1, \mathsf{K}_2) \in \{0,1\}^{\omega(\log\lambda)} \times \{0,1\}^\lambda$. Set $\sigma := \mathsf{K}_1$. $\mathsf{R} := \mathsf{K}_2$. $\mathsf{P} := (\mathsf{s}, \mathsf{C}, \sigma)$. Return (P, R).	$\underline{\mathsf{R} \leftarrow \overline{\mathsf{rrFE}}_{\mathsf{DDH}}.\mathsf{Rep}(\mathsf{pp}, \mathsf{w}', \mathsf{P})}$: Parse $\mathsf{P} = (\mathsf{s}, \mathsf{C}, \sigma)$. $\tilde{\mathsf{w}} \leftarrow \mathsf{w}' - \mathsf{Decode}(\mathsf{syn}(\mathsf{w}') - \mathsf{s})$. Compute $\tilde{\mathsf{k}} = \tilde{\mathbf{R}} \cdot \tilde{\mathsf{w}}$. $\boxed{\tilde{\mathsf{K}} \leftarrow \mathsf{C}^{\tilde{\mathsf{k}}}.}$ Parse $\tilde{\mathsf{K}} = (\tilde{\mathsf{K}}_1, \tilde{\mathsf{K}}_2) \in \{0,1\}^{\omega(\log\lambda)} \times \{0,1\}^\lambda$. Set $\tilde{\sigma} := \tilde{\mathsf{K}}_1$. If $\tilde{\sigma} = \sigma$, $\tilde{\mathsf{R}} := \tilde{\mathsf{K}}_2$. Else, $\tilde{\mathsf{R}} := \perp$. Return $\tilde{\mathsf{R}}$.

Fig. 7. The DDH based robustly reusable FE scheme of Cui et al. [15], reproduced for completeness.

the Rep algorithm will return an extracted key. Let $\mathcal{A}$ be an adversary attacking the robustness game in Fig. 1 (b) following Definition 7. The algorithm for $\mathcal{A}$ is given below.

Adversary $\mathcal{A}$'s Attack Algorithm. $\mathcal{A}$ performs the following two steps.

1. $\mathcal{A}$ sends two queries to the Gen oracle with the same input δ, for some δ with $\mathsf{wt}(\delta) < t$.
2. Let $\mathcal{A}$ receive the helper data (P_1, R_1) and (P_2, R_2) as the responses to those two queries, where $(P_1, R_1) \leftarrow \mathsf{rrFE.Gen}(w + \delta, crs)$, $(P_2, R_2) \leftarrow \mathsf{rrFE.Gen}(w + \delta, crs)$, $P_1 = (s_1, C_1, \sigma_1)$, $C_1 = g^{r_1}$, $P_2 = (s_2, C_2, \sigma_2)$, $C_2 = g^{r_2}$ according the FE scheme in Figure 7. Let ℓ be the length of σ in bits. $\mathcal{A}$ then computes $C' = C_1 \cdot C_2$, $\sigma' = ((\sigma_1|R_1)\cdot(\sigma_2|R_2))_{1\cdots\ell}$, where '|' denotes concatenation of two strings, and $(x)_{a\cdots b}$ denotes the bit sequence from the ath bit to the bth bit of x. Let $P' = (s_1, C', \sigma')$. $\mathcal{A}$ sends P' to the Rep oracle along with the same input δ.

Since $\mathsf{rrFE.Gen}$ uses the same crs, w and δ in both queries, the secure sketch output is the same for both Gen oracle queries, and hence $s_1 = s_2$. Notice that

$$C' = C_1 \cdot C_2 = g^{r_1 + r_2} \text{ and} \tag{7}$$

$$(\sigma_1|R_1) \cdot (\sigma_2|R_2) = g^{r_1 k} \cdot g^{r_2 k} = g^{(r_1+r_2)k} = (C')^k \tag{8}$$

Let $\tilde{R} = ((\sigma_1|R_1) \cdot (\sigma_2|R_2))_{(\ell+1)\cdots(\ell+1+\lambda)}$. Upon receiving P', the $\mathsf{rrFE.Rep}$ algorithm will output $\tilde{R}$ as the extracted key. This is because the verification of σ' will succeed as, from equation 8, we see that $\sigma' = ((C')^k)_{1\cdots\ell}$. Therefore, the robustness property does not hold.

Remark 4. We have shown that the above attack works on the DDH-based robustly reusable FE construction of Cui et al. [15, Section 5.3, Fig. 17], which uses *perfect randomness*. It can be shown that the same attack also works on Cui et al.'s robustly reusable FE construction [15, Section 5.3, Fig. 19] based on DDH, which incorporates *imperfect randomness*.

F Source entropy requirement of the LWE based rrFE of Wen et al. [50, Fig. 14]

In this section we compute possible parameters and required source entropy of the LWE based rrFE of Wen et al. [50, Fig. 14]. The parameters are chosen such that they satisfy the conditions stated in [50, Theorem 3]. In the theorem, ϵ and λ are independent variables, and other parameters are determined based on the chosen values of ϵ and λ. We describe how we computed the parameters of the scheme.

Our goal is to find the smallest min-entropy of a sample that is needed to extract a final key of length λ. That is, we need to find the minimum entropy M of a sample w. Note that this entropy is partially lost due to a secure sketch. However, we ignore the entropy loss due to the secure sketch. The evaluated min-entropy here is without subtracting the entropy loss due to secure sketch.

1. We choose the security parameter $\lambda = 8.63 \times 10^9$. The reason for choosing a high value of λ is to obtain a reasonable value of the length of the secure sketch, as well as to make the value $(2m)^\ell cBp/q$ negligible, which we will calculate later in this section.
2. To choose parameters that satisfy [50, Theorem 3], we note that $(2m)^\ell cBp/q$ must be negligible. ℓ is a parameter that determines the length of the secure sketch. More specifically, $|s| = \frac{\ell}{2}$, where $|s|$ is the length of the secure sketch in bits. m is a parameter that is directly related to M.
3. We choose the smallest values for c: $c = 1$. Note that c is used in the matrix $C = [-c, c]^{m\times m}$. There is no other restriction on c.
4. Let $abs(x)$ denote the absolute value of x. B is chosen such that $\Pr[abs(x) \leq B : x \leftarrow \Psi_\alpha]$ holds with overwhelming probability. Following Boneh et al. [5, Proof of Theorem 5.1], we have $B = m \times \omega(\sqrt{\log(m)})$. We choose $B = m \times \sqrt{\log(m)}$.
5. ϵ is a parameter that affects the hardness of LWE, and $0 < \epsilon < 1$. ϵ and λ determine the noise level of LWE. When λ is fixed, an increase in ϵ implies a decrease in the standard deviation of the noise level, and a decrease in ϵ implies an increase in the standard deviation of the noise level. Note that if we decrease ϵ, the parameter ℓ decreases and hence the length of the secure sketch output also decreases. This is because $\ell = 2 \times |s|$. We choose $\epsilon = .25$.
6. Then we compute α using the formula $\alpha = 2^{-\lambda^\epsilon}$. For $\lambda = 8.63 \times 10^9$ and $\epsilon = .25$, we obtain $\alpha = 2^{-(8.63\times 10^9)^{.25}} \approx 2^{-304.79}$.
7. Note that $q = O(\sqrt{(}\lambda)/\alpha)$. To make $(2m)^\ell cBp/q$ negligible, we choose $q = 2^{500} \times (\sqrt{(}\lambda)/\alpha)$. That is, we choose the constant term in the order relation $q = O(\sqrt{(}\lambda)/\alpha)$ to be 2^{500}. We obtain
$q = 2^{500} \times (\sqrt{(}8.63 \times 10^9)/2^{-304.79}) = 2^{500+304.79} \times 92897.79 \approx 2^{821.29}$.
8. Using [50, Theorem 3], we compute $m = \lceil \lambda \times \log(q) \rceil$. For the chosen value of λ and ϵ, we obtain $m \approx \lceil 2^{42.69} \rceil$. Let K be the secret key. Then $|K| = m^2 \times \log(q) \approx 2^{83.38+9.68} \approx 2^{93.06}$, which implies that for $\lambda = 8.63 \times 10^9$ and $\epsilon = .25$, the min-entropy of the source must be at least $2^{93.06}$.

9. Following [50, Theorem 3], we then compute $p = 2^{(\lambda)^{\epsilon} - \omega(\log(\lambda))}$. To find p, keeping $(2m)^{\ell} cBp/q$ negligible, we choose $p = 2^{(\lambda)^{\epsilon} - \omega(\log(\lambda))} \approx 2^{304.79 \breve{} 33.01} = 2^{271.78}$. For the chosen set of parameters, $\ell = \lambda^{\epsilon}/\log(\lambda) \approx 10$, and $(2m)^{\ell} cBp/q = 2^{-100}$. That is, for a security parameter $\lambda = 8.63 \times 10^9$ bits, we obtain a security level of 100 bits. The length of the secure sketch $s = \frac{\ell}{2} \approx 5$. The length of the secure sketch is low, for which it is not possible to find a suitable secure sketch. For $\lambda < 8.63 \times 10^9$ with $\epsilon = 0.25$, the length of the secure sketch will be less than 5.

Table 3 represents a parameter set for Wen et al.'s FE construction [50, Theorem 3].

λ	c	B	ϵ	α	q	m	$\|K\|$	p	ℓ	Δ_{rrFE}	$\|s\|$
8.63×10^9	1	$m \times \sqrt{\log(m)}$	0.25	$2^{-304.79}$	$2^{821.29}$	$\lceil 2^{42.69} \rceil$	$2^{93.06}$	$2^{271.78}$	10	2^{-100}	5

Table 3. Parameter sets for Wen et al.'s FE scheme [50, Theorem 3]. $\Delta_{\mathsf{rrFE}} = (2m)^{\ell} cBp/q$.

Figure 8 shows how the value $(2m)^{\ell} cBp/q$ varies for different values of the security parameter. In the figure, x represents the values of the security parameter λ, and y (i.e. $f(x)$) represents the value $-\log((2m)^{\ell} cBp/q)$ with $\epsilon = 0.25$. The figure shows that when the security parameter $\lambda = x = 8.63 \times 10^9$, $y = -\log((2m)^{\ell} cBp/q) = 100$, i.e. $(2m)^{\ell} cBp/q = 2^{-100}$.

G Proof of Theorem 1

Proof. We first prove *reusability* of FE and then it's *robustness* property according to Definition 7.

Reusability. We prove the theorem using a sequence of games: G_0 to G_5 as shown in Figure 9. The changes from G_j to G_{j+1} are underlined. The games are executed between a Challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Game G_0 is the experiment $\mathsf{Exp}^{\mathsf{reu}-0}_{\mathcal{A},\mathsf{rrFE}}$. More precisely,

1. Challenger $\mathcal{C}$ samples $r_{\mathsf{ext}} \xleftarrow{\$} \mathcal{S}$, chooses a degree $n = \mathsf{Poly}(\lambda)$ polynomial f. For $\tau \in (0, \frac{1}{2})$, it chooses an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ such that $n\tau \le t$, sets $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$ and sends crs to $\mathcal{A}$.
2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $x \leftarrow \mathsf{Ext}(w, r_{\mathsf{ext}})$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}^{\mathsf{R}}_{\tau}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.
3. Upon receiving a Gen oracle query with shift $\delta_i \in \mathcal{M}$, $\mathcal{C}$ checks if $\mathsf{wt}(\delta_i) \le t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, it computes $s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)$, $x_i \leftarrow \mathsf{Ext}(w + \delta_i, r_{\mathsf{ext}})$, samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$,

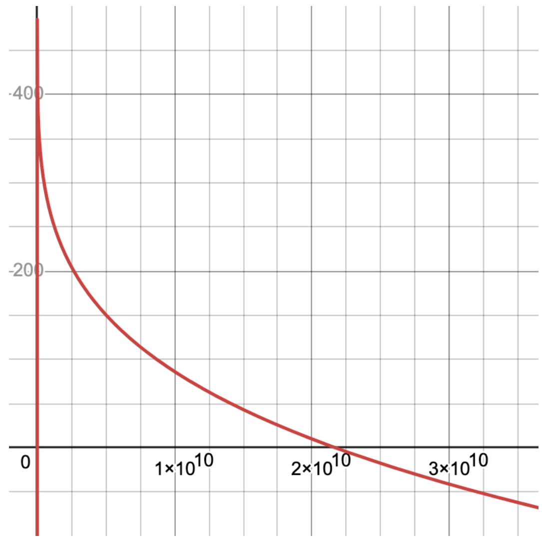

Fig. 8. A graph for computation of the function $f(x) = -\log\left(\left(\left(\left(2\left(x\ \log\left(\frac{2^{500}\sqrt{x}}{\left(2^{-(x)^{.25}}\right)}\right)\right)\right)\right)^{\frac{(x)^{.25}}{\log(x)}}\left(x\log\left(\frac{2^{500}\sqrt{x}}{2^{-(x)^{.25}}}\right)\right)\Delta_{\mathsf{f}}\right)\right)$, where $\Delta_{\mathsf{f}} = \sqrt{\log\left(\left(x\log\left(\frac{2^{500}\sqrt{x}}{2^{-(x)^{.25}}}\right)\right)\right)} 2^{\left(x^{(.25)}-\log(x)\right)}\left(\frac{2^{-(x)^{.25}}}{2^{500}\sqrt{x}}\right)$. The x axis represents the security parameter λ, and the y axis represents $f(x) = -\log((2m)^{\ell}cBp/q)$ with $\epsilon = 0.25$.

$y_{i2} \xleftarrow{\$} \mathcal{R}$, $a_i \xleftarrow{\$} \mathsf{R}$, $e_i \xleftarrow{\$} \mathsf{Ber}_{\tau}^{\mathsf{R}}$, computes $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$ and $R_i := y_{i2}$, returns (P_i, R_i) to $\mathcal{A}$.

4. If the adversary $\mathcal{A}$ outputs a bit b', the game outputs b'.

It can be seen that

$$\Pr[\mathsf{Exp}_{\mathcal{A},\mathsf{rrFE}}^{\mathsf{reu}-0} \Rightarrow 1] = \Pr[G_0 \Rightarrow 1] \tag{9}$$

Game G_1 is the same as game G_0, except some conceptual changes in the generation of (P_i, R_i). More precisely,

3. Upon receiving a Gen oracle query with shift $\delta_i \in \mathcal{M}$, $\mathcal{C}$ checks if $\mathsf{wt}(\delta_i) \leq t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, it computes $\underline{s_i \leftarrow s + \mathsf{SS.Gen}(\delta_i)}$ and $\underline{x_i \leftarrow x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})}$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_{\tau}^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.

(a) Games $G_0 - G_6$

$r_{\text{ext}} \xleftarrow{\$} \mathcal{S}$.
Choose a degree $n = \mathsf{Poly}(\lambda)$ polynomial f.
Set ring $\mathsf{R} = \mathsf{F}_2[X]/f(X)$.
For $\tau \in (0, \frac{1}{2})$, choose an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ with $n\tau \leq t$.
Set $\mathsf{crs} := (r_{\text{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$.

$w \leftarrow W$.
$s \leftarrow \mathsf{SS.Gen}(w)$.
$x \leftarrow \mathsf{Ext}(w, r_{\text{ext}})$.
$\boxed{x \xleftarrow{\$} \mathcal{X}.}$ // $G_2 - G_4$
$y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$; $y_2 \xleftarrow{\$} \mathcal{R}$; $a \xleftarrow{\$} \mathsf{R}$; $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$.
$c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$.
$\boxed{\boxed{y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}.}}$ // G_3
$\boxed{\boxed{y_2 \xleftarrow{\$} \mathcal{R}.}}$ // $G_3 - G_6$
$\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$.
$P := (s, a, c, \sigma)$ and $R := y_2$.
$b' \leftarrow \mathcal{A}^{O^{\mathsf{Gen}}(\cdot)}(P, R)$.
Output b'.

(b) $\mathsf{O}^{\mathsf{Gen}}(\delta_i)$

If $\mathsf{wt}(\delta_i) > t$,
  Output $\perp$.
Else,
  $s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)$.
  $\boxed{s_i = s + \mathsf{SS.Gen}(\delta_i).}$ // $G_1 - G_5$
  $x_i \leftarrow \mathsf{Ext}(w + \delta_i, r_{\text{ext}})$.
  $\boxed{x_i = x + \mathsf{Ext}(\delta_i, r_{\text{ext}}).}$ // $G_1 - G_5$
  $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$; $y_{i2} \xleftarrow{\$} \mathcal{R}$; $a_i \xleftarrow{\$} \mathsf{R}$.
  $e_i \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$.

  $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$.
  $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$.
  $P_i := (s_i, a_i, c_i, \sigma_i)$ and $R_i := y_{i2}$.

  Output (P_i, R_i).

Fig. 9. Games G_0 to G_5 for the proof of *reusability* of rrFE

Claim 1 $\Pr[G_0 \Rightarrow 1] = \Pr[G_1 \Rightarrow 1]$

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

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

.

By the homomorphic property of Ext, we have

$$x_i = \mathsf{Ext}(w + \delta_i, r_{\text{ext}}) = \mathsf{Ext}(w, r_{\text{ext}}) + \mathsf{Ext}(\delta_i, r_{\text{ext}}) = x + \mathsf{Ext}(\delta_i, r_{\text{ext}}).$$

Hence, we have

$$\Pr[G_0 \Rightarrow 1] = \Pr[G_1 \Rightarrow 1] \tag{10}$$

Hence, the claim follows. □

Game G_2 is the same as game G_1, except that the secret key for ring-LPN is uniformly and independently sampled from the secret key space $\mathcal{X}$ of ring-LPN instead of being the output of $\mathsf{Ext}(w, r_{\text{ext}})$. More precisely,

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $\underline{x \xleftarrow{\$} \mathcal{X}}$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.

Note that, this change is also reflected in $O^{\mathsf{Gen}}(\delta_i)$ as $x_i = x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$.

Claim 2 $|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \epsilon_{\mathsf{ext}}$

Proof. Assume that the adversary makes at most ρ queries to Gen oracle. Since $H_\infty(W) \geq m$ and SS is an $(m, \tilde{m}, t)$ secure sketch, we have

$$H_\infty(W|\mathsf{SS.Gen}(W)) \geq \tilde{m}. \tag{11}$$

Since Ext is an average-case $(\mathcal{M}, \tilde{m}, \mathcal{X}, \epsilon_{\mathsf{ext}})$-strong extractor, following equation 11, we have

$$\mathsf{SD}(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}}; U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \leq \epsilon_{\mathsf{ext}}, \tag{12}$$

where U is an uniformly and independently distributed random variable in $\mathcal{X}$.

Therefore, for any PPT adversary $\mathcal{B}$, we have

$$|\Pr[\mathcal{B}(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1] - \Pr[\mathcal{B}(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1]| \leq \epsilon_{\mathsf{ext}}. \tag{13}$$

We need to prove that $|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \epsilon_{\mathsf{ext}}$. We prove this by showing that if an adversary $\mathcal{A}$ can distinguish between G_1 and G_2, we can construct another adversary $\mathcal{B}$ that can distinguish between $(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$ and $(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$. The adversary $\mathcal{B}$ uses the adversary $\mathcal{A}$. Let the adversary $\mathcal{B}$ be given (x, s, r_{ext}), where x is either $\mathsf{Ext}(W, r_{\mathsf{ext}})$ or an uniformly and independently chosen value from $\mathcal{X}$, and $s = \mathsf{SS.Gen}(W)$. The aim of $\mathcal{B}$ is to distinguish between $(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$ and $(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$. $\mathcal{B}$ simulates the games G_1 and G_2 for $\mathcal{A}$ as follows.

1. $\mathcal{B}$ chooses a degree $n = \mathsf{Poly}(\lambda)$ polynomial f. For $\tau \in (0, \frac{1}{2})$, it chooses an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ such that $n\tau \leq t$, sets $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$ and returns crs to $\mathcal{A}$.
2. $\mathcal{B}$ samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.
3. Upon receiving $\mathcal{A}$'s query with shift $\delta_i \in \mathcal{M}$, $\mathcal{B}$ verifies if $\mathsf{wt}(\delta_i) \leq t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{B}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ computes $s_i \leftarrow s + \mathsf{SS.Gen}(\delta_i)$, $x_i \leftarrow x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$, samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \xleftarrow{\$} \mathcal{R}$, $a_i \xleftarrow{\$} \mathsf{R}$, $e_i \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$ and $R_i := y_{i2}$, returns (P_i, R_i) to $\mathcal{A}$.
4. If the adversary $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs b'.

It can be seen that, if $x = \mathsf{Ext}(W, r_{\mathsf{ext}})$, $\mathcal{B}$ perfectly simulates the game G_1; and if x is uniformly random in $\mathcal{X}$, $\mathcal{B}$ perfectly simulates the game G_2. Therefore,

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]|$$

$$\leq |\Pr[\mathcal{B}(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1] - \Pr[\mathcal{B}(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1]|$$
$$\leq \epsilon_{\mathsf{ext}}. \tag{14}$$

Hence, Claim 2 follows. □

Game G_3 is the same as game G_2 except that the secret key for KMS-MAC and R are uniformly chosen at random from $\mathcal{K}_{\mathsf{MAC}}$ and $\mathcal{R}$, respectively, instead of the same $(y_1 \parallel y_2)$ that is being used in the ring-LPN-based computation $(a \cdot x + e + \mathcal{E}(y_1 \parallel y_2))$. More precisely,

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $x \xleftarrow{\$} \mathcal{X}$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_{\tau}^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$. It <u>samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$ and $y_2 \xleftarrow{\$} \mathcal{R}$ uniformly and independently</u>, computes $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.

Claim 3 $|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}$.

Proof. We prove the claim by showing that, if an adversary $\mathcal{A}$ can distinguish between the game G_2 and G_3, we can construct another adversary $\mathcal{B}$ that can implement an attack on the ring-LPN problem (Definition 5). Let the adversary $\mathcal{B}$ receive $(a, p, \mathsf{R}, f, \tau)$, where (a, p) is either (i) generated by a ring-LPN distribution by computing $p = a \cdot x + e$ where $a \xleftarrow{\$} \mathsf{R}$, $x \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_{\tau}^{\mathsf{R}}$, or (ii) uniformly chosen at random from $(\mathsf{R} \times \mathsf{R})$. The aim of $\mathcal{B}$ is to decide whether (a, p) is generated by the ring-LPN distribution or sampled uniformly at random from $(\mathsf{R} \times \mathsf{R})$. $\mathcal{B}$ simulates games G_2 and G_3 for $\mathcal{A}$ as follows.

1. $\mathcal{B}$ samples $r_{\mathsf{ext}} \xleftarrow{\$} \mathcal{S}$, chooses an $[n, k, 2t + 1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ such that $n\tau \leq t$, sets $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$ and returns crs to $\mathcal{A}$.
2. $\mathcal{B}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$. It samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, computes $c = p + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.
3. Upon receiving $\mathcal{A}$'s query with shift $\delta_i \in \mathcal{M}$, $\mathcal{B}$ checks if $\mathsf{wt}(\delta_i) \leq t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ computes $s_i \leftarrow s + \mathsf{SS.Gen}(\delta_i)$, $\Delta_i \leftarrow \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$, samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \xleftarrow{\$} \mathcal{R}$. $\mathcal{B}$ queries its ring-LPN oracle to get (a_i, p_i), where $a_i \xleftarrow{\$} \mathsf{R}$, $p_i = a_i \cdot x + e_i$, $e_i \xleftarrow{\$} \mathsf{Ber}_{\tau}^{\mathsf{R}}$. It computes $c_i = p_i + a_i \cdot \Delta_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$ and $R_i := y_{i2}$, returns (P_i, R_i) to $\mathcal{A}$.
4. If $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs b'.

It can be seen that, if (a, p) is generated by the ring-LPN distribution, $\mathcal{B}$ perfectly simulates the game G_2; otherwise, if (a, p) is uniformly random in $(\mathsf{R} \times \mathsf{R})$, $\mathcal{B}$ perfectly simulates the game G_3. Therefore,

$$|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}.$$

Hence, Claim 3 follows. □

Game 4 is the same as game G_3 except that the secret key for $\mathsf{KMS}_{\mathsf{MAC}}$ is the same y_1 that is being used in the ring-LPN-based computation $(a \cdot x + e + \mathcal{E}(y_1 \parallel y_2))$ instead of uniformly and independently chosen from $\mathcal{K}_{\mathsf{MAC}}$. More precisely,

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $x \stackrel{\$}{\leftarrow} \mathcal{X}$, samples $y_1 \stackrel{\$}{\leftarrow} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \stackrel{\$}{\leftarrow} \mathcal{R}$, $a \stackrel{\$}{\leftarrow} \mathsf{R}$, $e \stackrel{\$}{\leftarrow} \mathsf{Ber}_{\tau}^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, samples $y_2 \stackrel{\$}{\leftarrow} \mathcal{R}$ uniformly and independently, computes $\sigma \leftarrow \mathsf{MAC}(\underline{y_1}, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.

Claim 4 $|\Pr[G_3 \Rightarrow 1] - \Pr[G_4 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}$.

Proof. We prove the claim by showing that if an adversary $\mathcal{A}$ can distinguish between game G_3 and game G_4, we can construct another adversary $\mathcal{B}$ that can successfully mount an attack on the ring-LPN problem (Definition 5). Let the adversary $\mathcal{B}$ receive $(a, p, \mathsf{R}, f, \tau)$, where (a, p) is either (i) uniformly chosen at random from $(\mathsf{R} \times \mathsf{R})$, or (ii) generated by a ring-LPN distribution by computing $p = a \cdot x + e$ where $a \stackrel{\$}{\leftarrow} \mathsf{R}$, $x \stackrel{\$}{\leftarrow} \mathsf{R}$, $e \stackrel{\$}{\leftarrow} \mathsf{Ber}_{\tau}^{\mathsf{R}}$. The aim of $\mathcal{B}$ is to decide whether (a, p) is uniformly chosen at random from $(\mathsf{R} \times \mathsf{R})$ or sampled from the ring-LPN distribution. $\mathcal{B}$ simulates games G_3 and G_4 for $\mathcal{A}$ as follows.

1. $\mathcal{B}$ samples $r_{\mathsf{ext}} \stackrel{\$}{\leftarrow} \mathcal{S}$, chooses an $[n, k, 2t + 1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ such that $n\tau \leq t$, sets $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$ and returns crs to $\mathcal{A}$.
2. $\mathcal{B}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, samples $y_1 \stackrel{\$}{\leftarrow} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \stackrel{\$}{\leftarrow} \mathcal{R}$, computes $c = p + \mathcal{E}(y_1 \parallel y_2)$, samples $y_2 \stackrel{\$}{\leftarrow} \mathcal{R}$, computes $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.
3. Upon receiving $\mathcal{A}$'s query with shift $\delta_i \in \mathcal{M}$, $\mathcal{B}$ checks if $\mathsf{wt}(\delta_i) \leq t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ computes $s_i \leftarrow s + \mathsf{SS.Gen}(\delta_i)$, $\Delta_i \leftarrow \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$, samples $y_{i1} \stackrel{\$}{\leftarrow} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \stackrel{\$}{\leftarrow} \mathcal{R}$. $\mathcal{B}$ queries its ring-LPN oracle and obtains (a_i, p_i), where $a_i \stackrel{\$}{\leftarrow} \mathsf{R}$, $p_i = a_i \cdot x + e_i$, $e_i \stackrel{\$}{\leftarrow} \mathsf{Ber}_{\tau}^{\mathsf{R}}$. It computes $c_i = p_i + a_i \cdot \Delta_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, computes $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$ and $R_i := y_{i2}$, returns (P_i, R_i) to $\mathcal{A}$.
4. If $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs b'.

It can be seen that, if (a, p) is sampled uniformly at random from $(\mathsf{R} \times \mathsf{R})$, $\mathcal{B}$ perfectly simulates game G_3; otherwise, if (a, p) is sampled according to the ring-LPN distribution, $\mathcal{B}$ perfectly simulates game G_4. Therefore,

$$|\Pr[G_3 \Rightarrow 1] - \Pr[G_4 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}.$$

Hence, Claim 4 follows. □

Game G_5 is the same as G_4 except that the generation of x is revert back to $x \leftarrow \mathsf{Ext}(w, r_{\mathsf{ext}})$. More precisely,

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $\underline{x \leftarrow \mathsf{Ext}(w, r_{\mathsf{ext}})}$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, samples $y_2 \xleftarrow{\$} \mathcal{R}$, computes $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$ and $R := y_2$, returns (P, R) to $\mathcal{A}$.

Note that this change is also reflected in $O^{\mathsf{Gen}}(\delta_i)$ as $x_i = x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$.

Claim 5 $|\Pr[G_4 \Rightarrow 1] - \Pr[G_5 \Rightarrow 1]| \leq \epsilon_{\mathsf{ext}}$

Proof. The proof of the claim is similar to the proof of Claim 2 as the modifications from game G_4 to G_5 is similar to that from game G_1 to G_2. We omit the proof here. □

Game G_6 is the same as game G_5 except that we revert back the conceptual changes done in the generation of (P_i, R_i) in game G_1. More precisely,

3. Upon receiving a Gen oracle query with shift $\delta_i \in \mathcal{M}$, $\mathcal{C}$ checks if $\mathsf{wt}(\delta_i) \leq t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, it computes $\underline{s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i),\ x_i \leftarrow \mathsf{Ext}(w + \delta_i, r_{\mathsf{ext}})}$, samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \xleftarrow{\$} \mathcal{R}$, $a_i \xleftarrow{\$} \mathsf{R}$, $e_i \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$ and $R_i := y_{i2}$, returns (P_i, R_i) to $\mathcal{A}$.

Claim 6 $\Pr[G_5 \Rightarrow 1] = \Pr[G_6 \Rightarrow 1]$

Proof. The proof of the claim is similar to the proof of Claim 1 as the modification from game G_5 to G_6 is similar to that from game G_0 to G_1. We omit the proof here. □

Note that game G_6 is exactly the same as experiment $\mathsf{Exp}^{\mathsf{reu}-1}_{\mathcal{A},\mathsf{rFE}}$. Hence, we have

$$\Pr[\mathsf{Exp}^{\mathsf{reu}-1}_{\mathcal{A},\mathsf{rrFE}} \Rightarrow 1] = \Pr[G_6 \Rightarrow 1] \tag{15}$$

Taking Eq. 9, Claim 1-6, Eq. 15 together, we have

$$\mathsf{Adv}_{\mathcal{A}}(\mathsf{Exp}^{\mathsf{reu}}_{\mathcal{A},\mathsf{rrFE}}) \leq 2\epsilon_{\mathsf{Ext}} + 2\epsilon_{\mathsf{RLPN}}. \tag{16}$$

Robustness. We now prove robustness property of the FE construction. We prove the *robustness* of the rrFE scheme using a series of games: G_0 to G_5 as shown in Figure 10. The changes from the game G_i to G_{i+1} are underlined. The games are executed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Game G_0 is the robustness experiment $\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}}$. More precisely,

1. Challenger $\mathcal{C}$ samples $r_{\mathsf{ext}} \xleftarrow{\$} \mathcal{S}$, chooses a degree $n = \mathsf{Poly}(\lambda)$ polynomial f. For $\tau \in (0, \frac{1}{2})$, it chooses an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ such that $n\tau \leq t$, sets $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$ and sends crs to $\mathcal{A}$.

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $x \leftarrow \mathsf{Ext}(w, r_{\mathsf{ext}})$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$, $R := y_2$ and $Q := \{P\}$, returns (P, R) to $\mathcal{A}$.
3. Upon receiving a Gen oracle query with shift $\delta_i \in \mathcal{M}$, $\mathcal{C}$ checks if $\mathsf{wt}(\delta_i) \leq t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, it computes $s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)$, $x_i \leftarrow \mathsf{Ext}(w + \delta_i, r_{\mathsf{ext}})$, samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \xleftarrow{\$} \mathcal{R}$, $a_i \xleftarrow{\$} \mathsf{R}$, $e_i \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, computes $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$, $R_i := y_{i2}$ and $Q := Q \cup \{P_i\}$, returns (P_i, R_i) to $\mathcal{A}$.
4. Upon receiving a forgery (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{wt}(\delta^*) > t$ or $P^* \in Q$, $\mathcal{C}$ returns 0. Else, it parses $P^* = (s^*, a^*, c^*, \sigma^*)$, computes $\tilde{w} \leftarrow \mathsf{SS.Rec}(w + \delta^*, s^*)$, $\tilde{x} \leftarrow \mathsf{Ext}(\tilde{w}, r_{\mathsf{ext}})$, $\tilde{k} \leftarrow \mathcal{D}(c^* + a^* \cdot \tilde{x})$, and parses $\tilde{k} = (\tilde{y}_1, \tilde{y}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$, and computes $\tilde{\sigma} \leftarrow \mathsf{MAC}(\tilde{y}_1, (s^*, a^*, c^*))$. If $\tilde{\sigma} = \sigma^*$, $\mathcal{C}$ returns 1, and the game returns 1. Else, $\mathcal{C}$ returns 0, and the game returns 0.

It can be seen that

$$\Pr[\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}} \Rightarrow 1] = \Pr[G_0 \Rightarrow 1] \tag{17}$$

Game G_1 is the same as game G_0 except the following conceptual changes.

3. Upon receiving a Gen oracle query with shift $\delta_i \in \mathcal{M}$, $\mathcal{C}$ checks if $\mathsf{wt}(\delta_i) \leq t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, it computes $\underline{s_i = s + \mathsf{SS.Gen}(\delta_i)}$, and $\underline{x_i = x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})}$, samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \xleftarrow{\$} \mathcal{R}$, $a_i \xleftarrow{\$} \mathsf{R}$, $e_i \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, computes $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$, $R_i := y_{i2}$ and $Q := Q \cup \{P_i\}$, returns (P_i, R_i) to $\mathcal{A}$.
4. Upon receiving a forgery (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{wt}(\delta^*) > t$ or $P^* \in Q$, $\mathcal{C}$ returns 0. Else, it parses $P^* = (s^*, a^*, c^*, \sigma^*)$, computes $\underline{\tilde{\delta} = \mathsf{f}(\delta^*, s, s^*)}$, and $\underline{\tilde{x} = x + \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}})}$, $\tilde{k} \leftarrow \mathcal{D}(c^* + a^* \cdot \tilde{x})$, and parses $\tilde{k} = (\tilde{y}_1, \tilde{y}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$, and computes $\tilde{\sigma} \leftarrow \mathsf{MAC}(\tilde{y}_1, (s^*, a^*, c^*))$. If $\tilde{\sigma} = \sigma^*$, $\mathcal{C}$ returns 1, and the game returns 1. Else, $\mathcal{C}$ returns 0, and the game returns 0.

By homomorphic property and linearity property of secure sketch, and homomorphic property of Ext, we have

$$\Pr[G_0 \Rightarrow 1] = \Pr[G_1 \Rightarrow 1] \tag{18}$$

Game G_2 is the same as game G_1 except that the secret key for ring-LPN is uniformly and independently sampled from the secret key space $\mathcal{X}$ of ring-LPN instead of being the output of $\mathsf{Ext}(w, r_{\mathsf{ext}})$. More precisely,

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $\underline{x \xleftarrow{\$} \mathcal{X}}$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$, $R := y_2$, and $Q := \{P\}$, returns (P, R) to $\mathcal{A}$.

Note that, this change is also reflected in $O^{\mathsf{Gen}}(\delta_i)$ as $x_i = x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$, and in the generation of $\tilde{x}$ as $\tilde{x} = x + \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}})$.

Claim 7 $|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \le \epsilon_{\mathsf{ext}}$

Proof. The proof of the claim is similar to the proof of Claim 2. Let the adversary make at most ρ queries to the Gen oracle. As $H_\infty(W) \ge m$ and SS is an $(m, \tilde{m}, t)$ secure sketch, we have that

$$H_\infty(W|\mathsf{SS.Gen}(W)) \ge \tilde{m}. \tag{19}$$

From equation 19 and noting that Ext is an average-case $(\mathcal{M}, \tilde{m}, \mathcal{X}, \epsilon_{\mathsf{ext}})$-extractor, we have

$$\mathsf{SD}(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}}; U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \le \epsilon_{\mathsf{ext}}, \tag{20}$$

where U is an uniformly and independently distributed random variable over $\mathcal{X}$.

Hence, for all PPT adversary $\mathcal{B}$, we have

$$|\Pr[\mathcal{B}(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1] - \Pr[\mathcal{B}(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1]| \le \epsilon_{\mathsf{ext}}. \tag{21}$$

We need to prove that $|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \le \epsilon_{\mathsf{ext}}$. To prove it, we show that if an adversary $\mathcal{A}$ can distinguish between the game G_1 and G_2, we can construct an adversary $\mathcal{B}$ that can distinguish between $(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$ and $(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$. Let the adversary $\mathcal{B}$ be given (x, s, r_{ext}), where x is either $\mathsf{Ext}(W, r_{\mathsf{ext}})$ or an uniformly and independently chosen value from $\mathcal{X}$, and $s = \mathsf{SS.Gen}(W)$. The goal of $\mathcal{B}$ is to distinguish between $(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$ and $(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}})$. $\mathcal{B}$ simulates the games G_1 and G_2 for $\mathcal{A}$ as follows.

1. $\mathcal{B}$ chooses a degree $n = \mathsf{Poly}(\lambda)$ polynomial f. For $\tau \in (0, \frac{1}{2})$, it chooses an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ such that $n\tau \le t$, sets $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$ and returns crs to $\mathcal{A}$.
2. $\mathcal{B}$ samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$, $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$, $R := y_2$, and $Q := \{P\}$, returns (P, R) to $\mathcal{A}$.
3. Upon receiving $\mathcal{A}$'s query with shift $\delta_i \in \mathcal{M}$, $\mathcal{B}$ verifies if $\mathsf{wt}(\delta_i) \le t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{B}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ computes $s_i \leftarrow s + \mathsf{SS.Gen}(\delta_i)$, $x_i \leftarrow x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$, samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \xleftarrow{\$} \mathcal{R}$, $a_i \xleftarrow{\$} \mathsf{R}$, $e_i \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$, computes $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$, $R_i := y_{i2}$ and $Q := Q \cup \{P_i\}$, returns (P_i, R_i) to $\mathcal{A}$.
4. Upon receiving a forgery (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{wt}(\delta^*) > t$ or $P^* \in Q$, $\mathcal{B}$ returns 0. Else, it parses $P^* = (s^*, a^*, c^*, \sigma^*)$, computes $\tilde{\delta} = \mathsf{f}(\delta^*, s, s^*)$, $\tilde{x} = x + \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}})$, $\tilde{k} \leftarrow \mathcal{D}(c^* + a^* \cdot \tilde{x})$, parses $\tilde{k} = (\tilde{y}_1, \tilde{y}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$, and computes $\tilde{\sigma} \leftarrow \mathsf{MAC}(\tilde{y}_1, (s^*, a^*, c^*))$. If $\tilde{\sigma} = \sigma^*$, $\mathcal{B}$ returns 1, and the game returns 1. Else, $\mathcal{B}$ returns 0, and the game returns 0.

If $x = \mathsf{Ext}(w, r_{\mathsf{ext}})$, $\mathcal{B}$ perfectly simulates the game G_1. Otherwise, if x is uniformly random in $\mathcal{X}$, $\mathcal{B}$ perfectly simulates the game G_2. Therefore,

$$\begin{aligned}
&|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \\
&\leq |\Pr[\mathcal{B}(\mathsf{Ext}(W, r_{\mathsf{ext}}), \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1] - \Pr[\mathcal{B}(U, \mathsf{SS.Gen}(W), r_{\mathsf{ext}}) \Rightarrow 1]| \\
&\leq \epsilon_{\mathsf{ext}}. \qquad (22)
\end{aligned}$$

Hence, Claim 7 follows. □

Game G_3 is the same as game G_2 except that the secret key for KMS-MAC and R are uniformly random in $\mathcal{K}_{\mathsf{MAC}}$ and $\mathcal{R}$, respectively, instead of the same $(y_1 \parallel y_2)$ that is being used in the ring-LPN-based computation $(a \cdot x + e + \mathcal{E}(y_1 \parallel y_2))$, and in the reproduction query, if a^* in the query is the same as one of the Gen query's output, it uses the previously used key that was used for MAC in that query and then shifts it with a value Δ known to the adversary. More precisely,

2. $\mathcal{C}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, $x \xleftarrow{\$} \mathcal{X}$, samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, $a \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}^{\mathsf{R}}_{\tau}$, computes $c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$. It $\underline{\text{samples } y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}} \text{ and } y_2 \xleftarrow{\$} \mathcal{R} \text{ uniformly and independently}}$, computes $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$, $R := y_2$, and $Q := \{P\}$, returns (P, R) to $\mathcal{A}$.
4. Upon receiving a forgery (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{wt}(\delta^*) > t$ or $P^* \in Q$, $\mathcal{C}$ returns 0. Else, it parses $P^* = (s^*, a^*, c^*, \sigma^*)$, computes $\tilde{\delta} = \mathsf{f}(\delta^*, s, s^*)$, and $\tilde{x} = x + \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}})$, $\tilde{k} \leftarrow \mathcal{D}(c^* + a^* \cdot \tilde{x})$, and parses $\tilde{k} = (\tilde{y}_1, \tilde{y}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$. $\underline{\text{If } (a^* = a)\text{, it computes } \tilde{\Delta} = \mathcal{D}(c^* + c + a^* \cdot \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}}))\text{, parses}}$ $\underline{\tilde{\Delta} = (\tilde{\Delta}_1, \tilde{\Delta}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}\text{, sets } \tilde{y_1} = y_1 + \tilde{\Delta}_1\text{. Else if } (a^* = a_i) \text{ for some}}$ $\underline{i\text{, it computes } \tilde{\Delta}_i = \mathcal{D}(c^* + c_i + a^* \cdot \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}}))\text{, parses}}$ $\underline{\tilde{\Delta}_i = (\tilde{\Delta}_{i1}, \tilde{\Delta}_{i2}) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}\text{, sets } \tilde{y_1} = y_{i1} + \tilde{\Delta}_{i1}\text{. Otherwise, it chooses}}$ $\underline{\text{uniformly random } \tilde{y_1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}}$. It then computes $\tilde{\sigma} \leftarrow \mathsf{MAC}(\tilde{y}_1, (s^*, a^*, c^*))$. If $\tilde{\sigma} = \sigma^*$, $\mathcal{C}$ returns 1, and the game returns 1. Else, $\mathcal{C}$ returns 0, and the game returns 0.

Claim 8 $|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \epsilon_{\mathsf{RLPN}}$.

Proof. The proof of the claim is similar to the proof of Claim 3. We show that, if an adversary $\mathcal{A}$ can distinguish between the game G_2 and G_3, we can construct an adversary $\mathcal{B}$ that can mount a key-shift attack on the ring-LPN problem (Definition 5). Let the adversary $\mathcal{B}$ receive $(a, p, \mathsf{R}, f, \tau)$, where (a, p) is either (i) generated by a ring-LPN distribution by computing $p = a \cdot x + e$ where $a \xleftarrow{\$} \mathsf{R}$, $x \xleftarrow{\$} \mathsf{R}$, $e \xleftarrow{\$} \mathsf{Ber}^{\mathsf{R}}_{\tau}$, or (ii) uniformly chosen at random from $(\mathsf{R} \times \mathsf{R})$. The aim of $\mathcal{B}$ is to decide whether (a, p) is generated by the ring-LPN distribution or sampled uniformly at random from $(\mathsf{R} \times \mathsf{R})$. $\mathcal{B}$ simulates games G_2 and G_3 for $\mathcal{A}$ as follows.

1. $\mathcal{B}$ samples $r_{\mathsf{ext}} \xleftarrow{\$} \mathcal{S}$, chooses an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ such that $n\tau \le t$, sets $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$ and returns crs to $\mathcal{A}$.
2. $\mathcal{B}$ samples $w \leftarrow W$, invokes $s \leftarrow \mathsf{SS.Gen}(w)$, It samples $y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_2 \xleftarrow{\$} \mathcal{R}$, computes $c = p + \mathcal{E}(y_1 \parallel y_2)$, computes $\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$, sets $P := (s, a, c, \sigma)$, $R := y_2$ and $Q := \{P\}$, returns (P, R) to $\mathcal{A}$.
3. Upon receiving $\mathcal{A}$'s query with shift $\delta_i \in \mathcal{M}$, $\mathcal{B}$ checks if $\mathsf{wt}(\delta_i) \le t$. If $\mathsf{wt}(\delta_i) > t$, $\mathcal{C}$ returns an error symbol $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ computes $s_i \leftarrow s + \mathsf{SS.Gen}(\delta_i)$ and $\Delta_i \leftarrow \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$. $\mathcal{B}$ samples $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$, $y_{i2} \xleftarrow{\$} \mathcal{R}$. $\mathcal{B}$ queries its ring-LPN oracle to get (a_i, p_i), where $a_i \xleftarrow{\$} \mathsf{R}$, $p_i = a_i \cdot x + e_i$, $e_i \xleftarrow{\$} \mathsf{Ber}_{\tau}^{\mathsf{R}}$. It computes $c_i = p_i + a_i \cdot \Delta_i + \mathcal{E}(y_{i1} \parallel y_{i2})$, $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$, sets $P_i := (s_i, a_i, c_i, \sigma_i)$, $R_i := y_{i2}$ and $Q := Q \cup \{P_i\}$, returns (P_i, R_i) to $\mathcal{A}$.
4. Upon receiving a forgery (P^*, δ^*) from $\mathcal{A}$, if $\mathsf{wt}(\delta^*) > t$ or $P^* \in Q$, $\mathcal{B}$ returns 0. Else, it parses $P^* = (s^*, a^*, c^*, \sigma^*)$, computes $\tilde{w} \leftarrow \mathsf{SS.Rec}(w + \delta^*, s^*)$, $\tilde{\delta} = \mathsf{f}(\delta^*, s, s^*)$. If $(a^* = a)$, it computes $\tilde{\Delta} = \mathcal{D}(c^* + c + a^* \cdot \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}}))$, parses $\tilde{\Delta} = (\tilde{\Delta}_1, \tilde{\Delta}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$, sets $\tilde{y_1} = y_1 + \tilde{\Delta}_1$. Else if $(a^* = a_i)$ for some i, it computes $\tilde{\Delta_i} = \mathcal{D}(c^* + c_i + a^* \cdot \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}}))$, parses $\tilde{\Delta_i} = (\tilde{\Delta}_{i1}, \tilde{\Delta}_{i2}) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$, sets $\tilde{y_1} = y_{i1} + \tilde{\Delta}_{i1}$. Otherwise, it chooses uniformly random $\tilde{y_1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$. It then computes $\tilde{\sigma} \leftarrow \mathsf{MAC}(\tilde{y}_1, (s^*, a^*, c^*))$. If $\tilde{\sigma} = \sigma^*$, $\mathcal{C}$ returns 1, and the game returns 1. Else, $\mathcal{C}$ returns 0, and the game returns 0.

It can be seen that, if (a, p) is generated by the ring-LPN distribution, $\mathcal{B}$ perfectly simulates the game G_2; otherwise, if (a, p) is uniformly random in $(\mathsf{R} \times \mathsf{R})$, $\mathcal{B}$ perfectly simulates the game G_3. Therefore,

$$|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \le \epsilon_{\mathsf{RLPN}}.$$

Hence, Claim 8 follows. □

Claim 9 $|\Pr[G_3 \Rightarrow 1]| \le \epsilon_{\mathsf{MAC}}$.

Proof. Since the key for the MAC is uniformly random or the key is offset by a value known to the adversary, from the security of KMS-MAC (Definition 4), we have that $|\Pr[G_3 \Rightarrow 1]| \le \epsilon_{\mathsf{MAC}}$. □

Combining Eq. 17, Eq. 18, Claim 7, 8, 9 together, we have

$$\mathsf{Adv}_{\mathcal{A}}(\mathsf{Exp}^{\mathsf{robs}}_{\mathcal{A},\mathsf{rrFE}}) \le \epsilon_{\mathsf{Ext}} + \epsilon_{\mathsf{RLPN}} + \epsilon_{\mathsf{MAC}}.$$

□

(a) Games $G_0 - G_3$

$r_{\mathsf{ext}} \xleftarrow{\$} \mathcal{S}$.
Choose a degree $n = \mathsf{Poly}(\lambda)$ polynomial f.
Set ring $\mathsf{R} = \mathsf{F}_2[X]/f(X)$.
For $\tau \in (0, \frac{1}{2})$, choose an $[n, k, 2t+1]$ linear error correcting code $(\mathcal{E}, \mathcal{D})$ with $n\tau \leq t$.
Set $\mathsf{crs} := (r_{\mathsf{ext}}, \mathsf{R}, f, \tau, (\mathcal{E}, \mathcal{D}))$.

$w \leftarrow W$.
$s \leftarrow \mathsf{SS.Gen}(w)$.
$x \leftarrow \mathsf{Ext}(w, r_{\mathsf{ext}})$.
$\boxed{x \xleftarrow{\$} \mathcal{X}.}$ // $G_2 - G_3$
$y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$; $y_2 \xleftarrow{\$} \mathcal{R}$; $a \xleftarrow{\$} \mathsf{R}$; $e \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$.
$c = a \cdot x + e + \mathcal{E}(y_1 \parallel y_2)$.
$\boxed{\boxed{y_1 \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}.}}$ // G_3
$\boxed{\boxed{y_2 \xleftarrow{\$} \mathcal{R}.}}$ // G_3
$\sigma \leftarrow \mathsf{MAC}(y_1, (s, a, c))$.
$P := (s, a, c, \sigma)$, $R := y_2$ and $Q := \{P\}$.
$(P^*, \delta^*) \leftarrow \mathcal{A}^{\mathsf{O}^{\mathsf{Gen}}(\cdot)}(P, R)$.
If $(\mathsf{wt}(\delta^*) > t$ or $P^* \in Q)$,
  Output 0.
Else,
  Parse $P^* = (s^*, a^*, c^*, \sigma^*)$.
  $\tilde{w} \leftarrow \mathsf{SS.Rec}(w + \delta^*, s^*)$.
  $\tilde{\delta} = \mathsf{f}(\delta^*, s, s^*)$. //$G_1 - G_3$
  $\tilde{x} \leftarrow \mathsf{Ext}(\tilde{w}, r_{\mathsf{Ext}})$.
  $\tilde{x} = x + \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}})$. //$G_1 - G_3$
  $\tilde{k} \leftarrow \mathcal{D}(c^* + a^* \cdot \tilde{x})$.
  Parse $\tilde{k} = (\tilde{y}_1, \tilde{y}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}$.
  $\boxed{\boxed{\text{If } (a^* = a),}}$ //$G_3 - G_3$
  $\boxed{\boxed{\tilde{\Delta} = \mathcal{D}(c^* + c + a^* \cdot \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}})).}}$ //G_3
  $\boxed{\boxed{\text{Parse } \tilde{\Delta} = (\tilde{\Delta}_1, \tilde{\Delta}_2) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}.}}$ //G_3
  $\boxed{\boxed{\tilde{y_1} = y_1 + \tilde{\Delta}_1}}$ //G_3
  $\boxed{\boxed{\text{Else if } (a^* = a_i) \text{ for some } i,}}$ //G_3
  $\boxed{\boxed{\tilde{\Delta}_i = \mathcal{D}(c^* + c_i + a^* \cdot \mathsf{Ext}(\tilde{\delta}, r_{\mathsf{ext}})).}}$ //G_3
  $\boxed{\boxed{\text{Parse } \tilde{\Delta}_i = (\tilde{\Delta}_{i1}, \tilde{\Delta}_{i2}) \in \mathcal{K}_{\mathsf{MAC}} \times \mathcal{R}.}}$ //G_3
  $\boxed{\boxed{\tilde{y_1} = y_{i1} + \tilde{\Delta}_{i1}}}$ //G_3
  $\boxed{\boxed{\text{Else,}}}$ //G_3
  $\boxed{\boxed{\tilde{y_1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}.}}$ //G_3
  $\tilde{\sigma} \leftarrow \mathsf{MAC}(\tilde{y}_1, (s^*, a^*, c^*))$.
  If $(\tilde{\sigma} = \sigma^*)$,
    Output 1.
  Else,
    Output 0.

(b) $\mathsf{O}^{\mathsf{Gen}}(\delta_i)$

If $\mathsf{wt}(\delta_i) > t$,
  Output $\perp$.
Else,
  $s_i \leftarrow \mathsf{SS.Gen}(w + \delta_i)$.
  $s_i = s + \mathsf{SS.Gen}(\delta_i)$. // $G_1 - G_3$
  $x_i \leftarrow \mathsf{Ext}(w + \delta_i, r_{\mathsf{ext}})$.
  $x_i = x + \mathsf{Ext}(\delta_i, r_{\mathsf{ext}})$. // $G_1 - G_3$
  $y_{i1} \xleftarrow{\$} \mathcal{K}_{\mathsf{MAC}}$; $y_{i2} \xleftarrow{\$} \mathcal{R}$; $a_i \xleftarrow{\$} \mathsf{R}$.
  $e_i \xleftarrow{\$} \mathsf{Ber}_\tau^{\mathsf{R}}$.
  $c_i = a_i \cdot x_i + e_i + \mathcal{E}(y_{i1} \parallel y_{i2})$.
  $\sigma_i \leftarrow \mathsf{MAC}(y_{i1}, (s_i, a_i, c_i))$.
  $P_i := (s_i, a_i, c_i, \sigma_i)$ and $R_i := y_{i2}$.
  $Q := Q \cup \{P_i\}$.
  Output (P_i, R_i).

Fig. 10. Games G_0 to G_3 for the proof of *robustness* of rrFE