A Reusable Fuzzy Extractor with Practical Storage Size: Modifying Canetti *et al.* 's Construction

Jung Hee Cheon, Jinhyuck Jeong✉, Dongwoo Kim, Jongchan Lee
{jhcheon, wlsyrlekd✉, dwkim606, jclee0208}@snu.ac.kr

Department of Mathematical Sciences, Seoul National University, Seoul, Korea

Abstract. After the concept of a Fuzzy Extractor (FE) was first introduced by Dodis *et al.* , it has been regarded as one of the candidate solutions for key management utilizing biometric data. With a noisy input such as biometrics, FE generates a public helper value and a random secret key which is reproducible given another input *similar* to the original input. However, "helper values" may cause some leakage of information when generated repeatedly by correlated inputs, thus *reusability* should be considered as an important property. Recently, Canetti *et al.* (Eurocrypt 2016) proposed a FE satisfying both *reusability* and *robustness* with inputs from low-entropy distributions. Their strategy, the so-called Sample-then-Lock method, is to sample many partial strings from a noisy input string and to lock one secret key with each partial string independently.
In this paper, modifying this reusable FE, we propose a new FE with size-reduced helper data hiring a threshold scheme. Our new FE also satisfies both reusability and robustness, and requires much less storage memory than the original. To show the advantages of this scheme, we analyze and compare our scheme with the original in concrete parameters of the biometric, IrisCode. As a result, on 1024-bit inputs, with false rejection rate 0.5 and error tolerance 0.25, while the original requires about 1TB for each helper value, our scheme requires only 300MB with an additional 1.35GB of common data which can be used for all helper values.

Keywords: Fuzzy extractors, reusability, key derivation, digital lockers, threshold scheme, biometric authentication

1 Introduction

Biometrics are metrics derived from biological characteristics inherent to each individual, such as fingerprints, iris patterns, facial features, gait, etc. A noteworthy property of this biometric information is inseparability. Biometric information cannot be separated from its owner, and can be used to authenticate a person without requiring other keys or passwords. However, biometric authentication has its problems; First, once biometric information is leaked to an adversary it is not easy to revoke. This makes protecting biometric information more crucial. Second, whenever one generates biometric data from their biological source using a device, small errors occur naturally because of the various environments and conditions.

This obstacle causes much harder problems in "Privacy-preserving Biometric Authentication" since classical cryptographic systems are constructed so that even little errors in inputs lead to huge errors in outputs. For privacy-preserving biometric authentication, there are recent works [1–4] using cryptographic tools, especially, homomorphic encryption. They propose a secure biometric authentication system which is executed with encrypted biometrics, to prevent an adversary from obtaining any information about the biometrics. Such an authentication system, however, may lose its power if the secret key is leaked and thus secret key management is a subject of major concern. Storing the secret key in secure memory and tamper-resistant hardware such as TrustZone and SoftwareGuardExtensions might be a solution, but these hardwares are too expensive, and/or can be vulnerable to physical attacks. For these reasons, generating a secret key whenever biometrics are scanned was proposed as an alternative solution, and the notion of Fuzzy Extractors (FE) was introduced by Dodis *et al.* It is a cryptographic primitive which extracts the same key from noisy inputs [5, 6].

More precisely, a fuzzy extractor consists of two algorithms; a generating algorithm (Gen) and a reproducing algorithm (Rep). Gen generates a random secret key and a public helper value from input biometrics. Rep reproduces the same key from the helper value and a biometric, when it is sufficiently similar to the original used in the Gen algorithm.

For the security of a FE, there are some important properties such as *robustness* and *reusability.* A fuzzy extractor is robust if an adversary cannot forge a given helper value in a way that Rep outputs a wrong key even though the input biometric is legitimate. This robustness is quite important, since in a non-robust FE, a user cannot trust the key generated by Rep, rendering the FE meaningless. On the other hand, a FE is reusable if it remains secure even if several pairs (random key, and related helper value) issued from correlated inputs are revealed to an adversary. Considering biometric authentication via FE, reusability guarantees that the authentication system is still safe for future use even if some helper values and related keys of a user have been compromised.

In [7], Apon *et al.* modified the constuction of [8] based on the LWE-assumption making it reusable with a common matrix for every input of Gen. Unfortunately, it fails to satisfy robustness since it is susceptible to trivial forgery. In Eurocrypt 2016, Canetti *et al.* proposed a reusable fuzzy extractor [9]. It is the first reusable robust fuzzy extractor without assumptions on correlations of multiple readings of the source, applying the sample-then-lock method with cryptographic digital lockers. It can tolerate $\frac{cn \ln n}{k}$ errors in a given n-bit input allowing running time in n^c with a security parameter of at most k. However, some biometrics such as IrisCode have error linear ($20\% \sim 30\%$) in n.

In this paper, we point out that Canetti *et al.* 's fuzzy extractor is inappropriate for these cases; it requires too much storage space for the helper value. In their construction, each locker acts as an oracle to check each partial substring of the input biometric, outputting the original secret key if that substring is correct. Therefore, a smaller substring size directly leads to a decrease in the security of the fuzzy extractor. Without diminishing the size of substrings, the number of lockers should increase exponentially, leading to impractical storage requirement in cases with linear errors of input.

The main idea of our construction is to overcome this oracle by modifying the digital lockers and using shorter substrings. We also exploit a (perfect) threshold scheme to divide each locker, preserving security. More precisely, we provide m modified lockers, and each unordered τ-pair of them is applied with a recovery algorithm of a threshold scheme for reproducing the secret key. As a result, the probability that each modified locker is unlocked successfully becomes larger under the same security, leading to a crucial decrease of storage for the helper values. Although time consumption increases as a side-effect, this trade-off is favorable because it can be relieved with parallel computing. More precisely, our contribution can be summarized as follows;

- Combining the reusable FE of [9] and a threshold scheme, we propose a new size-reduced reusable fuzzy extractor satisfying robustness.[1] Our construction is based on the same or weaker conditions on the biometric source distribution than Canetti *et al.* 's construction.
- We analyze this new FE and the original with concrete parameters focusing on the biometric IrisCode. As a result, we highly reduced the amount of storage space required. For example, when using a 1024 bit biometric with false rejection rate[2] 0.5, the original requires about 6GB of each helper value for error tolerance 0.2, 1TB for 0.25, and 270TB for 0.3. On the other hand, our scheme requires only 1.6MB for 0.2, 300MB for 0.25, and 111GB for 0.3 with an additional 1.35GB of common data which is commonly used for every helper value. One can find more information in Table 1 and Table 2.
- In fact, there is a trade-off between required time and storage space; approximately, a decrease by a factor of 10^3 in storage space causes an tenfold increase in required time. We implement our scheme as a proof-of-concept with parallel computing via Cuda, and show that the trade-off can be relieved outstandingly.

Road Map. In Section 2, we provide some preliminaries for our work. In Section 3, we briefly introduce the reusable fuzzy extractor of Canetti *et al.* with concrete analysis. In Section 4, we give our construction of new fuzzy extractor and analysis of it.

[1] Robustness can easily be satisfied by the random-oracle-based transform of [10] as mentioned in [9]. Thus, we only focus on the reusability in this paper.

[2] The false rejection rate is the probability that the reproducing algorithm Rep fails to regenerate the secret value even though a legitimate input is given.

2 Preliminaries

Through this paper, for a natural number a, $|a|$ denotes the bit size of a. Here we mostly adhere to the notations used by Canetti *et al.* , for convenience.

2.1 Entropy

Let X_i be a random variable over some alphabet $\mathcal{Z}$ for $i = 1, \dots, n$. We denote by a random variable $X = X_1, \dots, X_n := (X_1, \dots, X_n)$. The *minentropy* $H_\infty(X)$ of X is defined as

$$H_\infty(X) = -\log[\max_x Pr(X = x)],$$

and the *average (conditional) minentropy* $\tilde{H}_\infty(X|Y)$ of X given Y defined as

$$\tilde{H}_\infty(X|Y) = -\log[\mathbb{E}_y \max_x Pr(X = x|Y = y)].$$

The *computational distance* between variables X and Y is defined by $\delta^D(X,Y) = |\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]|$ for a given distinguisher D, and for a class of distinguishers $\mathcal{D}$ we define $\delta^{\mathcal{D}}(X,Y) = \max_{D \in \mathcal{D}} \delta^D(X,Y)$. We will consider the class $\mathcal{D}_s$ of distinguishers (circuit) of size at most s which output a single bit.

2.2 Fuzzy extractor and reusability

Fuzzy extractors (FE) consist of two algorithms; Gen and Rep. Gen takes an input w such as biometric data and outputs an extracted string r and a helper value $p \in \{0,1\}^*$. Rep takes as input w' and p and outputs the previous r whenever w' is similar to w. In this work, we focus on computational fuzzy extractors. (For the information-theoretic notions, see [6].) The formal definition of computational fuzzy extractors and their notion of security follows.

Definition 1 (Computational Fuzzy Extractors [8]) *Given a metric space $(\mathcal{M}, \mathsf{dis})$, let $\mathcal{W}$ be a family of probability distributions over $\mathcal{M}$. A pair of randomized procedures "generate" (Gen) and "reproduce" (Rep) is an $(\mathcal{M}, \mathcal{W}, \kappa, t)$-computational fuzzy extractor that is $(\varepsilon_{sec}, s_{sec})$-hard with error δ if* Gen *and* Rep *satisfy the following properties:*

- *The generate procedure* Gen *on input $w \in \mathcal{M}$ outputs an extracted string $r \in \{0,1\}^\kappa$ and a helper string $p \in \{0,1\}^*$.*
- ***Correctness** The reproduction procedure* Rep *takes an element $w' \in \mathcal{M}$ and a bit string $p \in \{0,1\}^*$ as inputs. The correctness property guarantees that if $\mathsf{dis}(w,w') \leq t$ and $(r,p) \leftarrow \mathsf{Gen}(w)$, then* $\Pr[\mathsf{Rep}(w',p) = r] \geq 1 - \delta$ *where the probability is over the randomness of* (Gen, Rep).
- ***Security** For any distribution $W \in \mathcal{W}$, the string r is pseudorandom conditioned on p, that is* $\delta^{D_{s_{sec}}}((R,P),(U_\kappa,P)) \leq \varepsilon_{sec}$.

Fuller *et al.* proposed a computational fuzzy extractor based on the Learning with Error (LWE) problem [8]. However, their construction does not satisfy *robustness* and *reusability*, which mean the security against an adversary forging a given helper value while avoiding detection,[3] and the security of a reissued pair $(r,p) \leftarrow \mathsf{Gen}(w)$ when an adversary has extorted some pairs $(r_i, p_i) \leftarrow \mathsf{Gen}(w_i)$ for correlated w and w_i's, respectively.

The formal definition of a reusable fuzzy extractor is as follows:

Definition 2 (Reusable Fuzzy Extractor [9]) *Let $\mathcal{W}$ be a family of distributions over $\mathcal{M}$. Let* (Gen, Rep) *be a $(\mathcal{M}, \mathcal{W}, \kappa, t)$-computational fuzzy extractor that is $(\varepsilon_{sec}, s_{sec})$-hard with error δ. Let $(W^1, W^2, \dots, W^\rho)$ be ρ correlated random variables such that each $W^j \in \mathcal{W}$. Let D be an adversary. Define the following game for all $j = 1, \dots, \rho$:*

- ***Sampling** The challenger samples $w^j \leftarrow W^j$ and $u \leftarrow \{0,1\}^\kappa$.*
- ***Generation** The challenger computes $(r^j, p^j) \leftarrow \mathsf{Gen}(w^j)$.*

[3] We refers the formal definition of robustness to [11].

– ***Distinguishing*** *The advantage of D is*

$$Adv(D) := \Pr[D(r^1,\dots,r^{j-1},r^j,r^{j+1},\dots,r^\rho,p^1,\dots,p^\rho)=1] - \Pr[D(r^1,\dots,r^{j-1},u,r^{j+1},\dots,r^\rho,p^1,\dots,p^\rho)=1].$$

$(\mathsf{Gen},\mathsf{Rep})$ *is $(\rho,\varepsilon_{sec},s_{sec})$-reusable if for all $D\in\mathcal{D}_{s_{sec}}$ and for all $j=1,\dots,\rho$, the advantage is at most ε_{sec}.*

The first reusable fuzzy extractor without assumptions about the correlations on multiple readings of the source is proposed by Canetti *et al.* in Eurocrypt 2016 using the digital lockers with sample-then-lock construction [9]. We analyze this scheme with concrete parameters focusing on the biometric IrisCode. It requires too much storage space to tolerate up to 20% or more errors in 1024-bit iris code. To overcome this problem, we propose a modified FE exploiting threshold scheme, which satisfies both robustness and reusability. More details including Canetti *et al.* 's construction and analysis of it are in Section 3. Construction of our new fuzzy extractor is in Section 4.

On the other hand, recently, another reusable fuzzy extractor has been proposed by [7] adapting the LWE-based FE [8]. They presented a generic technique for converting any weakly reusable FE to a strongly reusable one in the random-oracle model, and made a (strongly) reusable FE by modifying the original LWE-based FE into a weakly reusable one. Furthermore, they provided a construction of a strongly reusable FE based on the LWE assumption, not relying on the random oracles. However, it does not satisfy robustness. On the contrary, Canetti *et al.* [9]'s constructions can easily be made robust by the random-oracle-based transform of [10], and so can our modification.

2.3 (τ,m)-threshold scheme

The (τ,m)-threshold scheme is a secret sharing scheme with participants m and threshold τ. It consists of a Distribution Algorithm $\mathsf{DA}_{\tau,m}$ and a Recovery Algorithm $\mathsf{RA}_{\tau,m}$. $\mathsf{DA}_{\tau,m}$ takes a secret s, and divides it into m shares which are distributed to each participant. $\mathsf{RA}_{\tau,m}$ takes τ inputs, and outputs the original secret s only if each τ input is the corresponding share generated by $\mathsf{DA}_{\tau,m}(s)$. For the security of this threshold scheme, an adversary with less than τ shares should not be able to obtain any information about the secret.

The basic idea of a secret sharing scheme was introduced by Shamir and Blakely independently [12,13]. Shamir's scheme is based on polynomial interpolation, and it requires heavy computation for $\mathsf{DA}_{\tau,m}$ and $\mathsf{RA}_{\tau,m}$ due to the employment of a τ-degree polynomial. To reduce computational costs, a new secret sharing scheme using just EXCLUSIVE-OR (XOR) operations was proposed for special cases, such as $(2,3),(2,m),(3,m)$-threshold schemes by Ishizu *et al.* , Fujii *et al.* , Kuihara *et al.* , respectively [14–16]. Finally, Kurihara *et al.* proposed a (τ,m)-threshold scheme [17] generalizing previous schemes.

Perfect (τ,m)-threshold scheme. In the (τ,m)-threshold scheme, leakage of information about the secret can be measured by entropy. Let $H(X)$ denote the Shannon entropy of a random variable X. Let $s\in\mathcal{S}$ and $s_i\in\mathcal{S}_i$ be a secret and a share respectively, and S, S_i be the random variables of secrets and shares, respectively.
A (τ,m)-secret sharing scheme is *perfect* if

$$H(S|S_I)=\begin{cases}0 & \text{if } I \text{ contains } k \text{ or more elements}\\ H(S) & \text{otherwise}\end{cases}$$

where $I=\{i_1,i_2,\dots,i_j\}\subseteq\{1,2,\dots,N\}$, and $S_I=S_{i_1}S_{i_2}\dots S_{i_j}:=(S_{i_1},S_{i_2},\dots,S_{i_j})$.

Kurihara *et al.* 's (τ,m)-threshold scheme [17]. In fact, our scheme can be instantiated with any *perfect* secret sharing scheme. For the clarity of description and the concrete parameter comparison with Canetti *et al.* , we utilize Kurihara *et al.* 's (τ,m)-threshold scheme [17]. As far as we know, it is one of the most efficient (τ,m)-threshold schemes which are *perfect*. From now on, (τ,m)-threshold scheme refers to Kurihara *et al.* 's (τ,m)-threshold scheme. In the following, we list some properties of $\mathsf{DA}_{\tau,m}$ and $\mathsf{RA}_{\tau,m}$ of Kurihara *et al.* 's scheme used in this paper.

1. $\mathsf{DA}_{\tau,m}$ can only be constructed for a prime m. For a general m, one can take a prime m_p larger than m, run DA_{τ,m_p}, and discard the surplus shares.
2. For a fixed $D \in \mathbb{Z}_{>0}$, and an input secret $s \in \{0,1\}^{D(m_p-1)}$, $\mathsf{DA}_{\tau,m}(s)$ outputs $s_i \in \{0,1\}^{D(m_p-1)}$ for $i = 1, 2, \ldots, m_p$.
3. $\mathsf{RA}_{\tau,m}$ takes as input τ shares of secrets, and outputs s if all τ inputs are correct shares. For a set $S' = \{s'_1, \ldots, s'_\tau\}$, we denote $\mathsf{RA}_{\tau,m}(S') := \mathsf{RA}_{\tau,m}(s'_1, \ldots, s'_\tau)$.
4. $\mathsf{DA}_{\tau,m}$ requires at most $\tau D m_p(m_p-1)$ XOR operations.
5. Each $\mathsf{RA}_{\tau,m}$ requires at most $\tau D m_p(m_p-1)$ XOR operations given $D(m_p-1)$ by $\tau D(m_p-1)$ binary matrices. (Each of which can be generated by $O(\tau^3 m_p^3)$ bitwise XOR operations).

3 Canetti *et al.* 's Reusable Fuzzy Extractor

As mentioned before, Canetti *et al.* proposed a reusable fuzzy extractor using digital lockers and sample-then-lock construction. In this section, we review their construction and give an analysis on concrete parameters focusing on the case when the input biometric is IrisCode.

3.1 Sources with α-entropy k-samples

As in the Canetti *et al.* 's construction [9], we assume that the source $W = W_1W_2\ldots W_n$, consisting of strings of length n over some alphabet $\mathcal{Z}$ is a source with *α-entropy k-samples*, i.e., $\tilde{H}_\infty(W_{j_1}W_{j_2}\ldots W_{j_k}|j_1, j_2, \ldots j_k) \geq \alpha$ for k uniformly random indices $1 \leq j_1, j_2, \ldots, j_k \leq n$.

3.2 Digital lockers

A digital locker is a kind of symmetric encryption scheme which is secure even if many correlated keys have already been used before [18]. It is composed of two algorithms; lock, and unlock. The lock algorithm encrypts val (a value) with key (a key), and outputs $\mathsf{lock}(\mathsf{key}, \mathsf{val})$. The unlock algorithm decrypts $\mathsf{lock}(\mathsf{key}, \mathsf{val})$ with given key', outputs val if $\mathsf{key} = \mathsf{key}'$, and aborts ($\perp$) otherwise. The digital locker can be instantiated as $\mathsf{lock}(\mathsf{key}, \mathsf{val}) = (\mathsf{nonce}, H(\mathsf{nonce}, \mathsf{key}) \oplus (\mathsf{val}\|0^s))$ where nonce is a nonce , $\|$ denotes concatenation, and s is a security parameter. unlock is instantiated by XORing($\oplus$) $H(\mathsf{nonce}, \mathsf{key}')$ with $\mathsf{lock}(\mathsf{key}, \mathsf{val})$. H can be a random oracle [19], or a cryptographic hash function with specific properties [20]. Note that nonce is usually different for each lock, and by hashing it with key, the correlation between keys disappears. For the following definition of digital lockers, let $\mathsf{idealUnlock}(\mathsf{key}, \mathsf{val})$ be the oracle that returns val when given key, and $\perp$ otherwise.

Definition 3 (Digital locker) *The pair of algorithms* $(\mathsf{lock}, \mathsf{unlock})$ *with security parameter* λ *is an* ℓ*-composable secure digital locker with error* γ *if the following holds:*

- ***Correctness*** *For all* key *and* val, $\Pr[\mathsf{unlock}(\mathsf{key}, \mathsf{lock}(\mathsf{key}, \mathsf{val})) = \mathsf{val}] \geq 1 - \gamma$. *Furthermore, for any* $\mathsf{key}' \neq \mathsf{key}$, $\Pr[\mathsf{unlock}(\mathsf{key}', \mathsf{lock}(\mathsf{key}, \mathsf{val})) = \perp] \geq 1 - \gamma$.
- ***Security*** *For every PPT adversary A and every positive polynomial p, there exists a (possibly inefficient) simulator S and a polynomial $q(\lambda)$ such that for any sufficiently large* s, *any polynomial-long sequence of values* $(\mathsf{val}_i, \mathsf{key}_i)$ *for* $i = 1, \ldots, \ell$, *and any auxiliary input* $z \in \{0,1\}^*$,

$$\left|\Pr\left[A\left(z, \{\mathsf{lock}\left(\mathsf{key}_i, \mathsf{val}_i\right)\}_{i=1}^{\ell}\right) = 1\right] - \Pr\left[S\left(z, \{|\mathsf{key}_i|, |\mathsf{val}_i|\}_{i=1}^{\ell}\right) = 1\right]\right| \leq \frac{1}{p(\mathsf{s})}$$

where S is allowed $q(\lambda)$ oracle queries to the oracles $\{\mathsf{idealUnlock}(\mathsf{key}_i, \mathsf{val}_i)\}_{i=1}^{\ell}$.

3.3 Description

The main idea of Canetti *et al.* 's scheme [9] is that a random string $r \in \{0,1\}^\kappa$ is locked multiple times by some substrings v_1, . . . , v_ℓ of an input string w and thus each locked value can be unlocked only with $v_1, \ldots, v_\ell$, respectively. To reproduce the same r, one must extract substrings v'_1, . . . , v'_ℓ corresponding to $v_1, \ldots, v_\ell$, at least one of which must be identical to its counterpart, and proceed to unlock with those substrings.

Construction(Sample-then-Lock, [9]). Let $\mathcal{M} = \{0,1\}^n$ be an input space and $w = w_1 \dots w_n \in \mathcal{M}$, where $w_i \in \{0,1\}$. Let ℓ be a positive integer and let $(\mathsf{lock}, \mathsf{unlock})$ be an ℓ-composable secure digital locker with error γ. To recover the random value r in Rep, information on how the substrings are generated should be stored. Thus a helper value p containing the indices of the bits of $w = w_1 \dots w_n$ which are used for each substring is generated along with r in Gen. The algorithms are in the next table.

Algorithm 1: Gen and Rep of Canetti *et al.*'s Reusable Fuzzzy Extractor

```
Gen
Input: $w = w_1 \dots w_n$
1. Sample $r \xleftarrow{\$} \{0,1\}^\kappa$
2. For $i = 1, \dots, \ell$
     (i) Uniformly choose $j_{i,m} \xleftarrow{\$} \{1, \dots, n\}$
         for each $1 \le m \le k$
    (ii) $v_i \leftarrow w_{j_{i,1}} \dots w_{j_{i,k}}$
   (iii) $c_i \leftarrow \mathsf{lock}(v_i, r)$
    (iv) $p_i \leftarrow c_i, (j_{i,1}, \dots, j_{i,k})$
3. Output $(r, p)$ where $p = p_1 \dots p_\ell$
```

```
Rep
Input: $w' = w'_1 \dots w'_n$, $p = p_1 \dots p_\ell$
1. For $i = 1, \dots, \ell$
     (i) Parse $p_i$ as $c_i, (j_{i,1}, \dots, j_{i,k})$
    (ii) $v'_i \leftarrow w'_{j_{i,1}} \dots w'_{j_{i,k}}$
   (iii) $r_i \leftarrow \mathsf{unlock}(v'_i, c)$
         If $r_i \neq \perp$, then output $r_i$.
2. Output $\perp$
```

3.4 Analysis on concrete parameters

In this subsection, we give an analysis of Canetti *et al.* 's fuzzy extractor with concrete parameters with IrisCode as the input biometric. To make the False Rejection Rate (FRR) less than δ, it requires the following condition:

$$\left(1 - \left(1 - \frac{t}{n}\right)^k\right)^\ell + \ell \cdot \gamma \le \delta.$$

Using the approximation $e^x \approx 1 + x$, they suggested parameter conditions $\ell \cdot \gamma \le \delta/2$, $tk = cn \log n$, and $\ell \approx n^c \log \frac{2}{\delta}$ for some constant c. Note that under these parameter conditions, we have $\left(1 - \left(1 - \frac{t}{n}\right)^k\right)^\ell \approx (1 - e^{-\frac{tk}{n}})^\ell \approx \exp(-\ell e^{-\frac{tk}{n}}) \approx \delta/2$ where e is the natural constant.

However, if $\mathsf{lock}(\mathsf{key}, \mathsf{val}) = (\mathsf{nonce}, H(\mathsf{nonce}, \mathsf{key}) \oplus \mathsf{val}||0^s)$ where H is a hash function, we can set better parameters since $\gamma = 2^{-s}$ is small enough. In our parameter setting, we set $\delta = 1/2$, $\kappa = 128$, and use SHA2[4] with 224-bit output as an instantiation of H. Then, $\mathsf{lock}(v_i, r)$ has an error rate $\gamma \approx 2^{128-224} = 2^{-96}$, and $\ell \cdot \gamma$ is negligible. Therefore, we set parameters so that the first term of the above condition is slightly smaller than $\delta = 1/2$, instead of $\delta/2$. Now, we have $\left(1 - \left(1 - \frac{t}{n}\right)^k\right)^\ell \approx \exp(-\ell e^{-\frac{tk}{n}}) \lesssim \delta$ from $\ell \approx n^c \log \frac{1}{\delta} = e^{\frac{tk}{n}}$ and $tk = cn \log n$.[5]

Error tolerance. Many researches have indicated that the Threshold Hamming Distance $T := \frac{t}{n}$ of IrisCode should lie between 20% and 35% [21–23]. According to this, we set $T = 0.2, 0.25, 0.3, 0.35$.

Security. With the helper value p, an adversary without biometric information can run a brute force attack on digital locker $\mathsf{lock}(v_i, r)$ with an exhaustive search for v_i which is a partial biometric of a user. Therefore, $k = |v_i|$ must be larger than at least the security parameter λ. We set $k = \lambda = 80$.[6]

Iteration number. Given $T = t/n$, k, and $\delta = 0.5$, we set iteration number $\ell \approx e^{\frac{tk}{n}}$ so that the false rejection rate is smaller than 0.5.

[4] One can also use SHA3 or other hash functions.

[5] We take $\delta = 1/2$ for convenience. One can achieve $\delta = 1/2^b$ increasing ℓ to $b\ell$.

[6] In fact, we should take into account the min-entropy of the partial biometric, but we will assume that the min-entropy is k for simplicity.

Storage space. The helper value p consists of two parts; indices and locks for each iteration. The indices for each iteration represent k among n bit positions of the biometric, and requires $(k \log n)$-bits of storage space. On the other hand, since we use SHA2-224, $|r| = \kappa = 128$, $k = 80$, and the output size of hash function is 224bits. We set the nonce for the hash input to 144 bits[7]. As we need ℓ iterations, the total storage space for lockers is $\ell \cdot (k \log n + 368)$ bits.

Time consumption. To measure actual time consumption, we implemented Canetti *et al.* 's reusable fuzzy extractor as a C++ program. We used g++ 5.4.0 to compile C++ source codes under the C++ 11 standard and ran them on a GNU/Linux ubuntu 4.4.0-62-generic machine that has a Intel(R) Xeon(R) E5-2620 v4 2.10GHz CPU with a 64GB RAM and a x86_64 architecture. We measured the average time for 1 unlock under various sets of parameters, and obtained results as displayed in the table below.

Table 1: Security, storage space and time consumption with $\delta = 1/2$, $\kappa = 128$, SHA2-224.

Security	Biometric	Error Tolerance	Iterations	Storage space (Byte)			Rep
k	n	T	ℓ	index	lock	Total	Time(unlock) (μs)
80	512	0.20	4.41×10^7	3.97G	2.03G	6.00G	12.6
80	512	0.25	6.85×10^9	617G	315G	932G	12.6
80	512	0.30	1.87×10^{12}	168T	86.0T	254T	12.6
80	512	0.35	7.79×10^{14}	70.1P	35.8P	106P	12.6
80	1024	0.20	4.00×10^7	4.00G	1.84G	5.84G	13.9
80	1024	0.25	6.85×10^9	685G	315G	1T	13.9
80	1024	0.30	1.87×10^{12}	187T	86.0T	273T	13.9
80	1024	0.35	6.90×10^{14}	69.0P	31.8P	101P	13.9
80	2048	0.20	4.00×10^7	4.40G	1.84G	6.24G	15.5
80	2048	0.25	6.85×10^9	754G	315G	1.07T	15.5
80	2048	0.30	1.77×10^{12}	194T	81.3T	276T	15.5
80	2048	0.35	6.50×10^{14}	71.5P	29.9P	101P	15.5

In Table 1, we present security, storage space, and time required for each unlock with concrete parameters.[8] The maximum required time of Rep is $\ell \times$ Time(unlock). As fully carrying out all ℓ iterations of Rep is unfeasible for most parameter sets due to the large storage space requirements, we ran Rep for a much smaller number of iterations and computed the average running time for each single iteration of Rep and measured the storage memory theoretically.

The form of digital lockers are the same for all cases, and time for unlock changes little by input size. Note that the iteration ℓ and Storage space highly (exponentially) depends on T, but not on n.

4 Our Construction and Analysis

Note that, in Canetti *et al.* 's scheme, $tk = cn \log n$ and $l \approx n^c \log \frac{2}{\delta}$ give large ℓ values, leading to large storage space for $T \in [0.2, 0.35]$. One easy strategy for reducing memory requirements is reducing k. However, a smaller k value implies less security, since an adversary can easily unlock $\mathsf{lock}(\mathsf{key}, \mathsf{val})$ if $k = |\mathsf{key}|$ is small.

We solve this problem by preventing adversaries from checking their guesses on each individual lock. For this purpose, we use a modified digital locker $(\mathsf{lock}', \mathsf{unlock}')$. It is a symmetric encryption scheme very similar to the original digital locker except for one difference; unlock' outputs a random string

[7] In fact, we should take the size of nonce so that the resulting locker is ℓ-composable, i.e., no collision occurs among ℓ nonces. In our cases, 144(=224-80) bit is sufficient for the size of nonce.

[8] Canetti *et al.* [9] mentioned that with sophisticated samplers, one can decrease the required storage. However, it can only decrease the storage for index, and the storage for locks can not be decreased.

instead of $\perp$ when $\mathsf{key}' \neq \mathsf{key}$. With this modified digital locker, adversaries can not check whether their guesses are right or not, since they can not distinguish a random string from val in our construction.

However, a fuzzy extractor must output $\perp$ when the input is not legitimate. We additionally exploit a (τ, m)-threshold scheme to enable legitimacy checking. More precisely, we encrypt each share with the modified lock, so that the adversary can recover the original secret s only if he or she has found τ or more correct shares by unlocking corresponding lock's with their correct keys. Then, the legitimacy check of the recovered secret s' is done by $\mathsf{unlock}(s', \mathsf{lock}(s, r))$.

4.1 Construction

The details of our construction are as follows. First, the modified digital locker can be instantiated as the original digital locker with the reduction of the zero-padding portion, i.e., $\mathsf{lock}'(\mathsf{key}, \mathsf{val}) := (\mathsf{nonce}, \pi \circ H(\mathsf{nonce}, \mathsf{key}) \oplus \mathsf{val})$ for $\mathsf{val} \in \{0,1\}^v$ and $\mathsf{key} \in \{0,1\}^k$, where $\pi : \{0,1\}^\mu \longrightarrow \{0,1\}^v$ is the canonical projection of the first v bits of vectors in $\{0,1\}^\mu$, the output space of hash H. Unlock' is similar to unlock, XORing ($\oplus$) lock' with $\pi \circ H(\mathsf{nonce}, \mathsf{key}')$. The notion of security for the modified digital locker is the same as that of the original digital locker, except that if $\mathsf{key}' \neq \mathsf{key}$, $\mathsf{unlock}'(\mathsf{key}', \mathsf{lock}(\mathsf{key}, \mathsf{val}))$ outputs $\mathsf{val}' \neq \mathsf{val}$ which is indistinguishable from a uniformly random string. H can be a random oracle or the same cryptographic hash function H as in the original digital locker.

The Gen algorithm takes as input a bit string w with length n. For a divisor d of n,[9] we consider the set $\mathbb{P}_d(n)$ of partitions $\mathcal{P} = \{B_j : |B_j| = d\}_{j=1}^m$ of $[n] = \{1, \ldots, n\}$ where $m = n/d$.[10] For a partition $\mathcal{P}_i \in \mathbb{P}_d(n)$, we denote $v_{i,j} = w_{B_j} := w_{j_1}, \ldots, w_{j_d}$, where $B_j = \{j_1, \ldots, j_d\} \in \mathcal{P}_i$. We first choose a random string $r \in \{0,1\}^\kappa$ and lock it with a random secret $s_i \in \{0,1\}^k$ resulting in $\mathsf{lock}(s_i, r)$.[11]

Next we split this s_i into several shares $\{s_{i,j}\}_{j=1}^m$ using the Distribution Algorithm $\mathsf{DA}_{\tau,m}$ of the $(\tau, m)-$threshold scheme. We now choose a random partition $\mathcal{P}_i \in \mathbb{P}_d(n)$, which specifies $v_{i,j}$'s for $j = 1, \ldots, m$. Finally, lock the shares $s_{i,j}$ with the substrings $v_{i,j}$ of w using the modified locker, resulting in $\mathsf{lock}'(v_{i,j}, s_{i,j})$. We iterate this process N times, and output the public helper value which can be represented by $\{\mathsf{lock}(s_i, r), \mathsf{lock}'(v_{i,j}, s_{i,j})|_{j=1}^m, \mathcal{P}_i\}_{i=1}^N$.

The Rep algorithm is simple. Each partition $\mathcal{P}_i$ in the helper value specifies $v_{i,j}^*$'s from the input w^*. Unlock all modified $\mathsf{lock}'(v_{i,j}, s_{i,j})$'s with $v_{i,j}^*$'s. Finally, use Recovery Algorithm $\mathsf{RA}_{\tau,m}$ to recover s_i from $s_{i,j}^*$, and check if the recovered s_i^* is correct by unlocking $\mathsf{lock}(s_i, r)$. Output r if at least one of such unlocks was successful, and output $\perp$ otherwise.

Algorithm 2: Gen and Rep of our RFE

```
Gen
Input: $w = w_1 \ldots w_n$
1. Sample $r \xleftarrow{\$} \{0,1\}^\kappa$
2. For $i = 1, \ldots, N$
    (i) Choose $\mathcal{P}_i \in \mathbb{P}_d(n)$ and sample $s_i \xleftarrow{\$} \{0,1\}^k$
    (ii) $s_{i,1}, \ldots, s_{i,m} \leftarrow \mathsf{DA}_{\tau,m}(s_i)$ where $m = n/d$
    (iii) for $j = 1, \ldots, m$
        (iii)-1 $v_{i,j} = w_{B_j}$ where $\{B_1, \ldots, B_m\} = \mathcal{P}_i$
        (iii)-2 $c_{i,j} \leftarrow \mathsf{lock}'(v_{i,j}, s_{i,j})$
    (iv) $d_i \leftarrow \mathsf{lock}(s_i, r)$
    (v) $p_i \leftarrow c_{i,1}, \ldots, c_{i,m}, \mathcal{P}_i, d_i$
3. Output $(r, p)$ where $p = (p_1 \ldots p_N)$
```

```
Rep
Input: $w^* = w_1^* \ldots w_n^*$, $p = (p_1 \ldots p_N)$
1. For $i = 1, \ldots, N$
    (i) Parse $p_i$ as $c_{i,1}, \ldots, c_{i,m}, \mathcal{P}_i, d_i$
    (ii) For $j = 1, \ldots, m$
        (ii)-1 $v_{i,j}^* = w_{B_j}^*$ where $\{B_1, \ldots, B_m\} = \mathcal{P}_i$
        (ii)-2 $s_{i,j}^* \leftarrow \mathsf{unlock}'(v_{i,j}^*, c_{i,j})$
    (iii) For each subset $S$ of $\{s_{i,j}^*\}_{j=1}^m$ with cardinality $\tau$,
        (iii)-1 $s_i^* \leftarrow \mathsf{RA}_{\tau,m}(S)$
        (iii)-2 $r_i^* \leftarrow \mathsf{unlock}(s_i^*, d_i)$, and if $r_i^* \neq \perp$ then output $r_i^*$.
2. Output $\perp$
```

[9] We can also consider a divisor d of $n' \leq n$, and follow the construction taking n' instead of n.

[10] For convenience, we only consider the partitions whose elements have the same cardinality. An analogous statement can be made for more general partitions.

[11] Note that, in (τ, m) threshold scheme, the size of secret k is $D(m_p - 1)$ for some $D \in \mathbb{Z}_{>0}$. We take D satisfying proper security.

4.2 Parameters and security analysis

Correctness and security. To ensure correctness of the FE, the parameters must satisfy

$$\mathsf{FRR} := \Pr[\perp \leftarrow \mathsf{Rep}(w^*) | \mathsf{dis}(w, w^*) \leq t] \leq \delta.$$

To compute this probability, for fixed $\mathcal{P}_i$ and w^* with $\mathsf{dis}(w, w^*) = t$, let

$$q = \Pr\left[s = s_i^* | s_i^* \leftarrow \mathsf{RA}_{\tau,m}(S) \text{ for some } S \in \mathbb{P}_\tau(\{s_{i,j}^*\}_{j=1}^m)\right]. \qquad (1)$$

Note that q is independent from the index i. Then, FRR is at most $(1-q)^N + N \cdot \gamma$ considering incorrectness arising from error γ in the lockers. As in 3.4 we ignore $N \cdot \gamma$ and set $(1-q)^N \approx 1 - qN \lesssim \delta = 1/2$

Here, we state a lemma calculating the exact value of q. A proof is given in the appendix.

Lemma 4 *Let $\mathcal{M} = \{0,1\}^n$ be the input space of the reusable fuzzy extractor in Construction with parameters $n, d, \lambda, \tau, \delta, t$ as previously defined. For an input $w = w_1 w_2 \dots w_n$, let $(r, p) \leftarrow Gen(W)$. If a certifier has a query input $w^* = w_1^* \dots w_n^*$ with $\mathsf{dis}(w, w^*) = t$,*

$$q := \Pr(r_i^* = r) = \frac{{}_{\tau m}C_\tau}{{}_nC_t} \sum_{\eta=\tau}^{m} (-1)^{\eta-\tau} \cdot \frac{{}_{m-\tau}C_{\eta-\tau} \times {}_{n-\eta d}\,C_t}{\eta} \text{ for all } i = 1, \dots, N.$$

Here ${}_aC_b$ denotes the usual binomial coefficient $\frac{a!}{b!(a-b)!}$ for integers a, b such that $0 \leq b \leq a$.

We can easily see that our fuzzy extractor is reusable, as is Canetti *et al.* 's.

Theorem 5 *Let λ be a security parameter and $\mathcal{W}$ be a family of sources with α-entropy k-samples over $\mathcal{Z}^n$ where $\alpha = \omega(\log \lambda)$. Then for any $s_{sec} = \mathsf{poly}(\lambda)$ there exists an $\epsilon_{sec} = \mathsf{ngl}(\lambda)$ such that Construction is a $(\mathcal{Z}^n, \mathcal{W}, \kappa, t)$- computational fuzzy extractor that is $(\epsilon_{sec}, s_{sec})$-hard with error $\delta = (1-q)^N + mN \cdot \gamma$, where the formula for q is given in Lemma 4.*

Proof. The proofs for correctness and security are analogous to those of [9] and are in the appendix.

Reusability. As in [9], reusability follows easily from the security of digital lockers. To enable ρ reuses, we need $N(m+1) \cdot \rho$ composable digital lockers. Then we can simulate an adversary given $r^1, \dots, r^{i-1}, r^{i+1}, \dots, r^\rho$, and $p^1, \dots, p^\rho$ as a simulator with $r^1, \dots, r^{i-1}, r^{i+1}, \dots, r^\rho$ as auxiliary input in the security of digital locker (see Definition 3). Now, we can prove the reusability similarly to Theorem 5.

Theorem 6 *Fix ρ and let all the variables be as in Theorem 5, except that $(\mathsf{lock}, \mathsf{unlock})$ is $N(m+1) \cdot \rho$ - composable instead of $N(m+1)$ - composable*[12] *(for κ-bit values and keys over $\mathcal{Z}^k$). Then for all $s_{sec} = \mathsf{poly}(n)$ there exists some $\epsilon_{sec} = \mathsf{ngl}(n)$ such that Construction is a $(\rho, \epsilon_{sec}, s_{sec})$-reusable fuzzy extractor.*

Comparison with [9]. In Canetti *et al.* 's work [9], they used the subsets of strings (biometrics) to lock and take multiple samples for correctness. However, for reliable error tolerance, they required too many samples, resulting in the use of enormous amounts of memory space as displayed in Table 1. We divide said subsets into small pieces and use the threshold scheme to diminish storage space requirement. As a result, our scheme consumes more time as it requires multiple RA operations in recovering the secret. We will show that this can be resolved through the use of parallel computing. In [9], the source of w needed to be *α-entropy k-samples*, i.e., $\tilde{H}_\infty(W_{j_1} W_{j_2} \dots W_{j_k} | j_1, j_2, \dots j_k) \geq \alpha$ for k uniformly random indices $1 \leq j_1, j_2, \dots, j_k \leq n$. Our construction requires a slightly different condition regarding the distribution of the source : $\tilde{H}_\infty(W_{j_1} W_{j_2} \dots W_{j_k} | j_1, j_2, \dots j_k) \geq \alpha$ for k uniformly random indices $1 \leq j_1, j_2, \dots, j_k \leq n$ selected without repetition.

[12] Canetti *et al.* 's construction requires ℓ or $\ell\rho$ -composable digital lockers, and $\ell \geq N(m+1)$ in our parameter settings.

4.3 Analysis on concrete parameters

To analyze our scheme as in 3.4 with concrete parameters, we calculated the storage space and number of operations needed when employing Kurihara *et al.* 's threshold scheme. We set $\delta = 1/2$, $\kappa = 128$, $T = \frac{t}{n} = 0.2, 0.25, 0.3$, $\tilde{k} := \tau d \geq \lambda = 80$ and used SHA2-224 as the hash function as in section 3.4.

Security. To recover r, an adversary equipped with helper value p must correctly guess at least τ of the $d-$bit keys for lock''s. Therefore, $\tau \cdot d$ should be at least $\lambda = 80$, the security parameter. (Note that as in Canetti *et al.* 's scheme, we should consider the min-entropy of the partial biometric of length τd.)

Iteration number. For given $T = \frac{t}{n}$, $\tilde{k} = \tau d$, and $\delta = 0.5$, we can find iteration number N such that $\mathrm{FRR} \leq (1-q)^N + N \cdot \gamma \leq 0.5$ where q is defined in Lemma 4. As in section 3.4, $N\gamma$ is negligible.

Storage space. The helper value p again consists of two parts; indices and digital lockers. Indices for each iteration indicate which among m sets in a partition of $[n]$ each biometric bit belongs to, and take up roughly $(n \log m)$-bits of memory space. The size of a locker (of either type) is the sum of the output size 224 bits of hash function SHA2-224 and that of the nonce in the hash input which is 144 bits. Since we need $m+1$ lockers (1 for $\mathsf{lock}(s_i, r)$) each for a total of N iterations, the total memory required for p is $N \cdot (n \log m + (224+144) \cdot (m+1))$ bits. This is denoted as "Help.val." in Table 2. For efficient computation of the secret sharing scheme, we will additionally store $\binom{m}{\tau}$ precomputed $(m_p - 1) \times \tau(m_p - 1)$ binary matrices needed for each of the $\binom{m}{\tau}$ recovery algorithms. The matrices are reused for all N iterations. The amount of memory space dedicated to these matrices is denoted as "Mat. for RA" in Table 2.

Time consumption. We implemented our fuzzy extractor in the same environment as in Section 3.4.

Here we give a table for the required storage space, time consumption, and security of our reusable fuzzy extractor. Again, we did not run the program for all N iterations, but instead ran it for a smaller number of iterations multiple times to obtain average values of the time costs of the unlock' and $(\mathsf{RA} + \mathsf{unlock})$ operations. "All unlock' " denotes the time for (ii), and "$1(\mathsf{RA} + \mathsf{unlock})$" denotes the time for each subset S in (iii) of Rep (Algorithm 2).

Table 2: The table for the storage space, time consumption and security of our scheme.

Security	Biometric	Error Tolerance	d	τ	m	Iterations	Storage Space (Byte)				Time / Iteration (μs)	
$\tilde{k} = \tau d$	n	T				N	Mat. for RA	index	lock	Help.val.	All unlock'	$1(\mathsf{RA} + \mathsf{unlock})$
80	512	0.2	16	5	32	1674	1.47G	0.54M	2.45M	3.00M	184	25.2
80	512	0.2	20	4	25	38612	44.6M	11.5M	44.4M	55.9M	146	16.3
80	512	0.25	16	5	32	3.82×10^5	1.47G	122M	562M	685M	184	25.2
80	512	0.3	16	5	32	1.98×10^8	1.47G	63.5G	292G	355G	184	25.2
80	1024	0.2	20	4	51	516	1.35G	0.37M	1.21M	1.59M	292	39.0
81	1024	0.2	27	3	37	26786	34.0M	17.9M	45.6M	63.5M	428	18.8
80	1024	0.25	20	4	51	97751	1.35G	71.0M	228M	300M	292	39.0
80	1024	0.3	20	4	51	3.63×10^7	1.35G	26.3G	85.1G	111G	292	39.0
81	2048	0.2	27	3	75	1546	616M	2.47M	5.33M	7.80M	440	60.9
81	2048	0.25	27	3	75	326030	616M	520M	1.12G	1.64G	440	60.9

In our FE, Rep takes at most $N \cdot \left(\binom{m}{\tau} \cdot \mathsf{Time}(\mathsf{RA} + \mathsf{unlock}) + \mathsf{Time}(\text{All } \mathsf{unlock}')\right)$ time. The maximum time for Gen is $N \cdot \big(\mathsf{Time}(\mathsf{DA} + \mathsf{lock}) + \mathsf{Time}(\text{All } \mathsf{lock}')\big)$.[13]

We visualized the trade-off between time and helper value storage space in Fig. 1.[14] Every point in the figure comes from either Table 1 or Table 2. The amount of required memory tends to decrease by

[13] Since $\mathsf{Time}(\mathsf{RA}) \approx \mathsf{Time}(\mathsf{DA})$, maximal time of Rep is much bigger than that of Gen, and we only consider the time of Rep.

[14] The space for "Mat. for DA" is excluded since it is a common data for every users. It doesn't affect the tendency in this graph overall.

a factor of approximately 10^3, i.e. from GB to MB(or TB to GB) whenever time consumption increases tenfold. Although time consumption seems impractical for both FEs, this can be solved with parallel computing methods since Rep consists of mutually independent iterative routines. We actually implemented our scheme with parallel computing using CUDA as proof of this (though not optimized), and the obtained positive results. We compiled CUDA and C++(test driver) codes using nvcc v7.5.17 with the SM53 architecture and under the C++ 11 standard. Then we ran the program on the aforementioned GNU/Linux machine with the same CPU, with an additional NVIDIA GeForce GTX 1080 GPU attached for the parallel computing. For the case $(n, p, d, \tau, m) = (1024, 0.2, 27, 3, 37)$, the algorithm Rep takes only 151 seconds, which is 20 times faster than without parallelization.

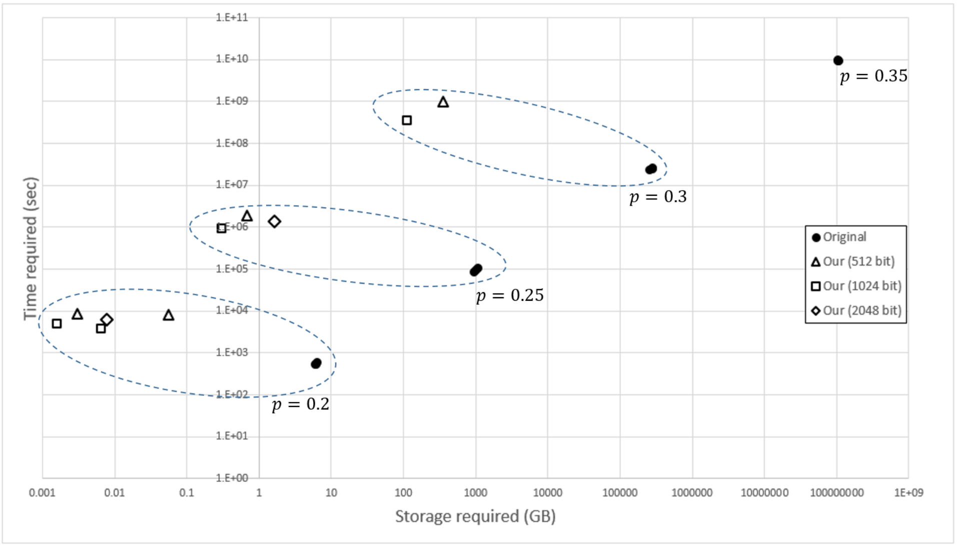

Fig. 1: A log-scaled graph of storage space for helper values and time (Original and Ours)

5 Conclusion

We analyzed the reusable fuzzy extractor of Canetti *et al.* with concrete parameters regarding iris authentication with IrisCode and found out that the required storage space is too large to be used in practice. To solve this problem, we propose a modified reusable fuzzy extractor using a perfect threshold scheme. Our modification cuts down the memory cost by a considerable amount. Though this approach yields a trade-off between memory and time costs, this can be resolved through parallel computing, since Rep consists of independent subroutines. When fully parallelized, our scheme reduces memory requirements from GB or TB to MB in many cases, while still operating in reasonable time.

Acknowledgements. The authors would like to thank the anonymous reviewers of ACISP 2018 for their valuable comments. This work were supported by Samsung Electronics, Co., Ltd. (No. 0421-20150074).

References

1. Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: Privacy Enhancing Technologies, 9th International Symposium, PETS 2009. Proceedings, pp. 235–253 (2009). doi:10.1007/978-3-642-03168-7_14
2. Kulkarni, R., Namboodiri, A.M.: Secure hamming distance based biometric authentication. In: International Conference on Biometrics, ICB 2013, pp. 1–6 (2013). doi:10.1109/ICB.2013.6613008
3. Karabat, C., Kiraz, M.S., Erdogan, H., Savas, E.: THRIVE: threshold homomorphic encryption based secure and privacy preserving biometric verification system. EURASIP J. Adv. Sig. Proc. **2015**, 71 (2015). doi:10.1186/s13634-015-0255-5
4. Cheon, J.H., Chung, H., Kim, M., Lee, K.: Ghostshell: Secure biometric authentication using integrity-based homomorphic evaluations. IACR Cryptology ePrint Archive **2016**, 484 (2016)

5. Dodis, Y., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, pp. 523–540 (2004). doi:10.1007/978-3-540-24676-3_31
6. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008). doi:10.1137/060651380
7. Apon, D., Cho, C., Eldefrawy, K., Katz, J.: Efficient, reusable fuzzy extractors from lwe. In: International Conference on Cyber Security Cryptography and Machine Learning, pp. 1–18. Springer (2017). doi:10.1007/978-3-319-60080-2_1
8. Fuller, B., Meng, X., Reyzin, L.: Computational fuzzy extractors. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 174–193. Springer (2013). doi:10.1007/978-3-642-42033-7_10
9. Canetti, R., Fuller, B., Paneth, O., Reyzin, L., Smith, A.D.: Reusable fuzzy extractors for low-entropy distributions. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 117–146 (2016). doi:10.1007/978-3-662-49890-3_5
10. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 147–163. Springer (2005). doi:10.1007/11426639_9
11. Dodis, Y., Kanukurthi, B., Katz, J., Reyzin, L., Smith, A.D.: Robust fuzzy extractors and authenticated key agreement from close secrets. IEEE Trans. Information Theory **58**(9), 6207–6222 (2012). doi:10.1109/TIT.2012.2200290
12. Shamir, A.: How to share a secret. Communications of the ACM (11), 612–613 (1979). doi:10.1145/359168.359176
13. Blakley, G.R.: Safeguarding cryptographic keys. In: Proc. AFIPS 1979 National Computer Conference, pp. 313–317 (1979). doi:10.1109/AFIPS.1979.98
14. Ishizu, H., Ogihara, T.: A study on long-term storage of electronic data. In: Proc. IEICE General Conf., D-9-10, 1, p. 125 (2004)
15. Fujii, Y.: A fast (2, n)-threshold scheme and its application. Proc. CSS2005 pp. 631–636 (2005)
16. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A fast (3, n)-threshold secret sharing scheme using exclusive-or operations. IEICE transactions on fundamentals of electronics, communications and computer sciences **91**(1), 127–138 (2008). doi:10.1093/ietfec/e91-a.1.127
17. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A new (k, n)-threshold secret sharing scheme and its extension. Information Security pp. 455–470 (2008). doi:10.1007/978-3-540-85886-7_31
18. Canetti, R., Kalai, Y.T., Varia, M., Wichs, D.: On symmetric encryption and point obfuscation. In: Theory of Cryptography Conference, pp. 52–71. Springer (2010). doi:10.1007/978-3-642-11799-2_4
19. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: International conference on the theory and applications of cryptographic techniques, pp. 20–39. Springer (2004). doi:10.1007/978-3-540-24676-3_2
20. Canetti, R., Dakdouk, R.R.: Obfuscating point functions with multibit output. In: Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, pp. 489–508 (2008). doi:10.1007/978-3-540-78967-3_28
21. Hollingsworth, K.P., Bowyer, K.W., Flynn, P.J.: Improved iris recognition through fusion of hamming distance and fragile bit distance. IEEE transactions on pattern analysis and machine intelligence **33**(12), 2465–2476 (2011). doi:10.1109/TPAMI.2011.89
22. Daugman, J.: Probing the uniqueness and randomness of iriscodes: Results from 200 billion iris pair comparisons. Proceedings of the IEEE **94**(11), 1927–1935 (2006). doi:10.1109/JPROC.2006.884092
23. Desoky, A.I., Ali, H.A., Abdel-Hamid, N.B.: Enhancing iris recognition system performance using templates fusion. Ain Shams Engineering Journal **3**(2), 133–140 (2012). doi:10.1109/ISSPIT.2010.5711758

A The Proof of Theorems

Lemma 4 *Let $\mathcal{M} = \{0,1\}^n$ be the input space of the reusable fuzzy extractor in Construction with parameters $n, d, \lambda, \tau, \delta, t$ defined as in Parameter Setting. For an input $w = w_1 w_2 \dots w_n$, let $(r, p) \leftarrow Gen(W)$. If a certifier has a query input $w^* = w_1^* \dots w_n^*$ with $\mathsf{dis}(w, w^*) = t$,*

$$q := \Pr(r_i^* = r) = \frac{\tau_m C_\tau}{{}_n C_t} \sum_{\eta=\tau}^{m} (-1)^{\eta-\tau} \cdot \frac{{}_{m-\tau}C_{\eta-\tau} \times {}_{n-\eta d}\, C_t}{\eta} \text{ for all } i = 1, \dots, N.$$

Here ${}_aC_b$ denotes the usual binomial coefficient $\frac{a!}{b!(a-b)!}$ for integers a, b such that $0 \le b \le a$.

Proof. Let A_j be the set of events with $r^*_{i,j} = r_{i,j}$, which is also the set of events with $v^*_{i,j} = v_{i,j}$. Let us define $A_J := \bigcap_{j\in J} A_j$, $A^*_J := A_J \cap (\bigcap_{j\in G-J} A^\complement_j)$ for any subset J of the index set $G := \{1, 2, \dots , m\}$, and denote by $\mathcal{G}_\eta$ the family of all η-element subsets of G for $\eta = 1, \dots , m$.
For $\eta = \tau, \dots , m$, and $J = \{g_1, \dots , g_\eta\} \in \mathcal{G}_\eta$, we have

$$\Pr(A_J) = \Pr(A_{g_1} \cap \ldots \cap A_{g_\eta}) = \frac{{}_{n-\eta d}C_t}{{}_nC_t},$$

since $A_{g_1} \cap \ldots \cap A_{g_\eta}$ is equal to the set of events in which all t of the indices j such that $w_j \neq w^*_j$ belong to the $m - \eta$ boxes in $\{B_g | g \notin \{g_1, \dots , g_\eta\}\}$. Also, as a consequence of the definitions of A_J and A^*_Q, and that all of the set of events A^*_Q are exclusive, we have

$$\Pr(A_J) = \sum_{J\subseteq Q\subseteq G} \Pr(A^*_Q).$$

From our use of the (τ , m) - threshold scheme, the set of events in which $r^*_i = r_i$ is the union of all A_J for which $J \subset G$ and $|J| \geq \tau$, which in turn is the disjoint union of all A_Q for which $Q \subset G$ and $|Q| \geq \tau$, and thus

$$q = \sum_{\substack{Q\subset G\\ |Q|\geq\tau}} \Pr(A^*_Q).$$

From the symmetry between elements of $\mathcal{G}_\eta$, the cardinality of which is $|\mathcal{G}_\eta| =_m C_\eta$, we can deduce that

$$\sum_{J\in\mathcal{G}_\eta} \Pr(A_J) =_m C_\eta \cdot \frac{{}_{n-\eta d}C_t}{{}_nC_t}.$$

Combining this with

$$\frac{\tau \cdot_m C_\tau \times_{m-\tau} C_{\eta-\tau}}{\eta} =_{\eta-1} C_{\tau-1} \times_m C_\eta \text{ for } \eta = \tau, \dots , m,$$

we can observe that the rightmost side of the equation in the lemma statement is equal to

$$\begin{aligned}
\frac{1}{{}_nC_t} \sum_{\eta=\tau}^{m} (-1)^{\eta-\tau} \cdot_{\eta-1} C_{\tau-1} \times_m C_\eta \times_{n-\eta d} C_t &= \sum_{\eta=\tau}^{m} [(-1)^{\eta-\tau} \cdot_{\eta-1} C_{\tau-1} \cdot \sum_{J\in\mathcal{G}_\eta} \Pr(A_J)] \\
&= \sum_{\eta=\tau}^{m} (-1)^{\eta-\tau} \cdot_{\eta-1} C_{\tau-1} \cdot \sum_{J\in\mathcal{G}_\eta} \sum_{J\subset Q} \Pr(A^*_Q) \\
&= \sum_{\substack{Q\subset G\\ |Q|\geq\tau}} \left(\sum_{\substack{J\subset Q\\ |J|\geq\tau}} (-1)^{|J|-\tau} \cdot_{|J|-1} C_{\tau-1} \right) \Pr(A^*_Q) \\
&= \sum_{\substack{Q\subset G\\ |Q|\geq\tau}} \left(\sum_{\eta=\tau}^{|Q|} (-1)^{\eta-\tau} \cdot_{\eta-1} C_{\tau-1} \times_{|Q|} C_\eta \right) \Pr(A^*_Q),
\end{aligned}$$

the third equation following from simple double counting. Thus it follows from the identity of the following Lemma A.1

$$\sum_{\eta=\tau}^{x} (-1)^{\eta-\tau} \cdot_{\eta-1} C_{\tau-1} \times_x C_\eta = 1 \text{ for } x = \tau, \tau + 1, \dots , m$$

that the coefficients of $\Pr(A^*_Q)$ in the previous equation are all 1, and therefore that the rightmost side of the equation is indeed equal to $q = \sum_{\substack{Q\subset G\\ |Q|\geq\tau}} \Pr(A^*_Q)$. □

Lemma A.1 *For any integer $x \geq \tau$, we have*

$$\sum_{\eta=\tau}^{x} (-1)^{\eta-\tau} \cdot_{\eta-1} C_{\tau-1} \times_x C_\eta = 1.$$

Proof. Simple computations regarding the binomial coefficients in the left hand side give us

$$\sum_{\eta=\tau}^{x} (-1)^{\eta-\tau} \cdot_{\eta-1} C_{\tau-1} \cdot_x C_\eta = x \times_{x-1} C_{\tau-1} \cdot \sum_{\eta=\tau}^{x} \frac{(-1)^{\eta-\tau}}{\eta} \cdot_{x-\tau} C_{\eta-\tau}.$$

Integrating the generating function $f(y) = y^{\tau-1}(1-y)^{x-\tau} = \sum_{\eta=\tau}^{x} (-1)^{\eta-\tau} \cdot_{x-\tau} C_{\eta-\tau} \cdot y^{\eta-1}$, we have

$$\int_0^y f(z)dz = \sum_{\eta=\tau}^{x} \frac{(-1)^{\eta-\tau}}{\eta} \cdot_{x-\tau} C_{\eta-\tau} \cdot y^{\eta},$$

and taking $y = 1$, we arrive at

$$\sum_{\eta=\tau}^{x} \frac{(-1)^{\eta-\tau}}{\eta} \cdot_{x-\tau} C_{\eta-\tau} = \int_0^1 f(z)dz.$$

Using integration by parts results in

$$\int_0^1 y^{\tau-1}(1-y)^{x-\tau}dy = \frac{\tau-1}{x-\tau+1} \int_0^1 y^{\tau-2}(1-y)^{x-\tau+1}dy,$$

the repeated use of which leads to

$$\int_0^1 y^{\tau-1}(1-y)^{x-\tau}dy = \ldots = \frac{(\tau-1)\cdots 1}{(x-\tau+1)\cdots(x-1)} \int_0^1 (1-y)^{x-1}dy = \frac{1}{x \times_{x-1} C_{\tau-1}},$$

which in turn implies our desired result. □

Theorem 5 *Let λ be a security parameter and $\mathcal{W}$ be a family of sources with α-entropy k-samples*[15] *over $\mathcal{Z}^n$ where $\alpha = \omega(\log \lambda)$. Then for any $s_{sec} = \mathsf{poly}(\lambda)$ there exists an $\epsilon_{sec} = \mathsf{ngl}(\lambda)$ such that Construction is a $(\mathcal{Z}^n, \mathcal{W}, \kappa, t)$- computational fuzzy extractor that is $(\epsilon_{sec}, s_{sec})$-hard with error $\delta = (1-q)^N + mN \cdot \gamma$, where the formula for q is given in 4.*

Proof. Correctness follows from Lemma 4.

The security proof is almost the same as the proof of Canetti*et al.* 's [9], except that we additionally consider lock' and recovery algorithm (RA) of the threshold scheme. We assume that RA is given to the distinguisher and simulator though we don't denote it explicitly.

Let R, P be the random variables on r, and the public helper value, respectively. U is a uniformly random variable over $\{0,1\}^\kappa$. For the proof, we will show that for all $s_{sec} = \mathsf{poly}(\lambda)$, there exists $\epsilon_{sec} = \mathsf{ngl}(\lambda)$ such that $\delta^{D_{s_{sec}}}((R,P),(U,P)) \leq \epsilon_{sec}$. In other words, we need to bound $|\mathbb{E}[D(R,P)] - \mathbb{E}[D(U,P)]|$ by a negligible function $\frac{1}{p(\lambda)}$ for some polynomial $p(\cdot)$. We will denote $D = D_{s_{sec}}$ from now on.

First, we can substitute the distinguisher D by an unbounded simulator S with the security of digital lockers (Definition 3). That is, there is a polynomial $q(\cdot)$ and an unbounded time simulator S, which makes at most $q(\lambda)$ queries to the oracles $\mathsf{idealUnlock}(s_i, r)_{i=1}^N$, such that

$$\left| \mathbb{E}[D(R,P)] - \mathbb{E}\left[S^{\mathsf{idealUnlock}(s_i,r)_{i=1}^N} \left(R, \{\mathsf{lock}'_{i,j}|_{j=1}^m, \mathcal{P}_i\}_{i=1}^N, k, \kappa\right)\right] \right| \leq \frac{1}{3p(\lambda)},$$

where $\mathsf{lock}'_{i,j} = \mathsf{lock}'(v_{i,j}, s_{i,j})$. Note that this is also true for R substituted by U.

[15] We set D, m_p such that $k \leq D(m_p - 1)$, and take $k = D(m_p - 1)$.

Now, we claim that

$$\Big| \mathbb{E}\left[S^{\mathsf{idealUnlock}(s_i,r)_{i=1}^N}\left(R, \{\mathsf{lock}'_{i,j}|_{j=1}^m, \mathcal{P}_i\}_{i=1}^N, k, \kappa\right)\right] - \mathbb{E}\left[S^{\mathsf{idealUnlock}(s_i,r)_{i=1}^N}\left(U, \{\mathsf{lock}'_{i,j}|_{j=1}^m, \mathcal{P}_i\}_{i=1}^N, k, \kappa\right)\right]\Big| \leq \frac{1}{3p(\lambda)}.$$

Given idealUnlock, the only way of S to distinguish $r \in R$ and $u \in U$ is to query idealUnlock and get a non-$\perp$ response, which is equivalent to correctly guessing s_i of $\mathsf{idealUnlock}(\mathsf{s_i}, \mathsf{r})$. Unbounded simulator S can guess s_i directly or by guessing the share $s_{i,j}$ of s_i with $\mathsf{lock}'_{d,k,\mu}(v_{i,j}, s_{i,j})$. Note that we exploited *perfect* (τ, m)-threshold scheme so that $H(S_i|\, S_{i,j},\ j \in J) = H(S_i)$ if J contains elements less than τ. In other words, S must correctly guess τ $s_{i,j}$'s to find s_i using RA or S gets no advantage over randomly(uniformly random) guessing s_i. Guessing τ $s_{i,j}$'s with $\mathsf{lock}'_{d,k,\mu}(v_{i,j}, s_{i,j})$ is equivalent to guessing τ $v_{i,j}$'s. Note that our construction satisfies $d\tau \geq k$, and guessing τ of the $v_{i,j}$'s is at least harder than guessing k bits of the partial biometrics. The argument regarding the probability of guessing k bits of partial biometrics correctly when given $q(\lambda)$ queries on idealUnlock is the same as in the proof of Lemma 1 in [9].

First, we modify S slightly so that it quits immediately after getting a non-$\perp$ answer.[16] Then, the view of S on q or less queries have $q+1$ values. (q values of getting a non-$\perp$ answer on the i-th query, and 1 value of getting all $\perp$ answers.) By Lemma 2.2b in [6], $\tilde{H}_\infty(V_i|View(S), j_{i,j}) \geq \tilde{H}_\infty(V_i|j_{i,j}) - \log(q+1) \geq \alpha - \log(q+1)$. Therefore, at each query, the probability that S gets a non-$\perp$ answer is at most $(q+1)/2^\alpha$. Since there are q queries of S, the overall probability is bounded by $q(q+1)/2^\alpha$. Now, $\alpha = \omega(\log \lambda)$ gives $\frac{q(q+1)}{2^\alpha} \leq \frac{1}{3p(\lambda)}$, and the claim holds. □

[16] It has no advantage in distinguishing r and u. Refer to [9]