Unforgettable: Multifactor Fuzzy Key Derivation Function

Oleksandr Kurbatov*, Dmytro Zakharov*‡, Lasha Antadze†, Victor Mashtalyar*,
Roman Skovron*, Volodymyr Dubinin*
*Distributed Lab †Rarimo ‡Kyiv School of Economics

***Abstract*—Secure storage of private keys is a long-standing challenge. While centralized infrastructure commonly relies on passwords or multi-factor authentication, decentralized applications rely almost exclusively on mnemonic-based key recovery schemes, most notably the BIP-39 standard for seed phrases. Although this approach improves memorability, users still frequently physically store mnemonics, largely defeating its intended usability and security benefits.**

We introduce Unforgettable — the multifactor fuzzy key derivation function that utilizes a diverse set of accessible sources (such as passwords, objects, facial data or fingerprints) to generate strong cryptographic keys. Since many of such sources are unstable, we employ fuzzy extractors to deterministically derive the same key despite small errors in several factors. Finally, we show that it is possible to achieve the false rejection rate of 1% with the 100-bit security level using passwords and two unstable factors.

1. Introduction

The secure and user-friendly key recovery have long been a major challenge in decentralized applications. Currently, the de-facto most practical solution is BIP-39 standard [34] introduced back in 2013. In its essence, BIP-39 proposed a way to convert a high-entropy randomness into a human-readable transcription of 12 words, which are surely more memorable and portable than a random 128-bit binary string. Despite its wide usage, the standard has a list of requirements and drawbacks: (i) the user cannot choose words from the dictionary as it reduces the entropy significantly, (ii) checksum detects only random errors and does not protect from the list of targeted attacks such as mnemonic substitution [35] or pre-computed by the malicious wallet seed phrases [28], (iii) key derivation function (KDF) is strongly defined to be `PBKDF2` [24] with the iteration parameter of 2048, which does not support the use of more secure KDFs [6] or increasing the iteration parameter to the level proposed by NIST [8].

The natural solution to improve user experience (UX) is to derive keys from more memorable data, such as passwords or biometrics. However, using one of these factors solely provides too weak security. For instance, a random password of 20 characters can provide the user with 128 bits of entropy, but keyword here is "random." Users do not generate passwords uniformly randomly, so to reach the same level of entropy, one needs to increase the password length to 50-60 characters. Brainwallets [43] is a similar interesting concept, which proposes selecting any data (a chapter in a book or a long sentence the user remembers) and passing it through the KDF function. This approach is better from the UX perspective, but security assumptions are much worse than for the randomly generated key.

What about biometrics, such as facial data, fingerprints, or voice? From a security perspective, biometric data is an even worse option: most key derivation protocols over biometric data can achieve at most 20–30 bits of security [3]. Moreover, biometric sources are *unstable*: upon receiving two biometric samples from the same person, they will not be precisely equal. This problem is well-researched in the prior literature: one can employ *fuzzy commitments* [22], *fuzzy vaults* [20], *fuzzy signatures* [25], [41], *asymetric fuzzy encapsulation* [14], and most notably, *fuzzy extractors* [11]. While such constructions make key derivation deterministic (as long as two biometric samples are close enough) and allow formal security analysis, they cannot increase the source entropy nor prevent a vast list of attacks, such as spoofing for impersonation [16].

Our Contribution. While passwords of biometrics solely cannot achieve a sufficient security level, we can expect to get a high-entropy key by combining several such factors. In this study, we present **Unforgettable** – the fuzzy key derivation function that from the diverse set of accessible sources (such as passwords, objects, or biometrics) generates strong cryptographic keys. Unforgettable Fuzzy Extractor is aiming to meet the following requirements:

- ✓ **Client-Side**: the whole recovery process can be conducted by the user without accessing any third-party services. Any auxilary data needed for recovery might be stored on blockchain and does not reduce security of derived keys.
- ✓ **Security**: entropy of the derived key should be approximately the sum of entropies of each individual source. With enough factors, the derived key is sufficiently strong.
- ✓ **Usability**: key generation and recovery flows are supposed to be much simpler and friendlier, even compared to the BIP-39 approach.

1.1. Previous Studies

While we have considered some of the single-factor derivation methods, we shall discuss previous studies on combining several factors.

MFKDF. Most notable study on this question is the Multi-Factor Key Derivation Function (MFKDF) [33] by Nair and Song that introduced the proper formalization and security analysis of merging multiple factors into a single key. Their construction defines the factor F as a

two-valued probability distribution (α, κ) with α being the public component and κ being the secret component. Each factor is equipped with the *factor construction* $\mathcal{F}$ that maps a factor witness W_i and public parameters α to a secret κ and new public parameters α': that is, $\mathcal{F} : (W_i, \alpha) \mapsto (\kappa_F, \alpha')$. Denote concatenation by $\odot$. Given a set of such factors $(F_i)_{i\in[N]}$, the MFKDF is constructed as follows:

(i) one derives individual factor materials $(\kappa_{F_i}, \alpha'_{F_i}) \leftarrow \mathcal{F}(W_{F_i}, \alpha_{F_i})$ for each factor F_i;
(ii) the resultant key K is derived from $(\kappa_{F_i})_{i\in[N]}$ (by essentially applying a KDF to $\bigodot_{i\in[N]} \kappa_{F_i}$);
(iii) public components α_{F_i} are set to newly created α'_{F_i}, if the factor F_i is *dynamic* (e.g., HOTP or TOTP).

Therefore, it is natural to embed the fuzzy extractor into such a theoretical framework: each biometric type would be a factor where α is a helper string that remains unchanged during the FactorUpdate procedure, while κ is the secret key derived through the fuzzy extractor. However, compared to the early draft of MFKDF2 [37], we claim that the fuzzy extractor scheme is incompatible with the construction of Nair and Song, since MFKDF requires *impossibility of guessing individual factors*. In such case, given that i^{th} factor's brute-force time complexity is proportional to 2^{λ_i}, we expect their combination to be brute-forced by adversary in time proportional to $\prod_{i\in[N]} 2^{\lambda_i}$. Fuzzy extractors fail to meet this criterion: given a helper string, the adversary always knows whether the decoded secret is valid. That said, instead of deriving the whole set $(\kappa_{F_i})_{i\in[N]}$ and passing it to the KDF, one can first brute-force individual κ_{F_i}'s and only after recovering each of them invoke key derivation. This drastically reduces brute-force complexity from $\prod_{i\in[N]} 2^{\lambda_i}$ to $\sum_{i\in[N]} 2^{\lambda_i}$, completely eliminating all the security benefits of merging several factors. Since this idea is crucial for the subsequent discussion, we illustrate it using the following example.

Example 1.1. Fix N password factors with λ-bit passwords $(\mathsf{p}_i)_{i\in[N]}$, secure hash function $H_0 : \{0,1\}^\lambda \to \{0,1\}^\ell$ and key-derivation function $\mathsf{KDF} : \{0,1\}^{\lambda N} \to \{0,1\}^\ell$ (with $\lambda N \le \ell$). Consider two derivation schemes:

- *Case A:* Each password factor has a public component $\alpha_i = H_0(\mathsf{p}_i)$ and the factor material is $\kappa_i = \mathsf{p}_i$. The combined key is $K := \mathsf{KDF}(\bigodot_{i\in[N]} \kappa_i)$.
- *Case B:* Each password factor has no public component: $\alpha_i = (\perp)$, and the factor material is $\kappa_i = \mathsf{p}_i$. As before, $K = \mathsf{KDF}(\bigodot_{i\in[N]} \kappa_i)$.

Now note that in *Case A*, the adversary can recover each κ_i separately using α_i with complexity 2^λ, resulting in the total complexity of $N \cdot 2^\lambda$. In the *Case B*, the best adversary can do is to brute-force over *tuples* $(\kappa_1, \ldots, \kappa_N)$, which lifts up the complexity to $2^{N\lambda}$. Therefore, *Case B* is a significantly stronger derivation scheme than *Case A*.

The same logic applies for fuzzy extractors, where α is the helper data while κ is the secret value.

2. Technical Preliminaries

2.1. Basic Notation

Throughout all sections, we are going to work with various *metric spaces*. We equip spaces of form Σ^n over the finite alphabet Σ with the *Hamming metric*. Namely, for $\mathbf{x}, \mathbf{y} \in \Sigma^n$, define the *Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$ as $\Delta(\mathbf{x}, \mathbf{y}) \triangleq \#\{i \in [n] : x_i \neq y_i\}$, while $\delta(\mathbf{x}, \mathbf{y}) \triangleq \frac{1}{n}\Delta(\mathbf{x}, \mathbf{y})$ to be the *normalized Hamming distance* (here and after, by $[n]$ we denote the set $\{1, \ldots, n\}$ and by (n) the set $\{0, \ldots, n-1\}$). We call the Hamming distance from $\mathbf{x} \in \Sigma^n$ to the zero string by *Hamming weight* of $\mathbf{x}$ and denote by $\|\mathbf{x}\|_0 \triangleq \Delta(\mathbf{x}, \mathbf{0})$.

In turn, when working over vector spaces of form $\mathbb{R}^n$, we typically use L^p distance, defined as $\|\mathbf{x} - \mathbf{y}\|_p$ with $\|\mathbf{x}\|_p \triangleq (\sum_{i\in[n]} |x_i|^p)^{1/p}$. When writing $d(\mathbf{x}, \mathbf{y})$, we assume that we equip $\mathbb{R}^n$ with the Euclidean metric (i.e., we use L^2 distance), if not stated otherwise.

Finally, as noted earlier, $\odot$ denotes concatenation of two strings while $\oplus$ denotes the usual XOR operation.

2.2. Error-Correction Codes

We denote the error correction code (ECC) over finite alphabet Σ by $\mathcal{C}$ and call it an (n, k, d) code if (1) $\mathcal{C} \subseteq \Sigma^n$, (2) messages are elements of Σ^k, and (3) distance of the code is d, where the distance of the code is defined as usual: $d(\mathcal{C}) \triangleq \min_{\mathbf{x},\mathbf{y}\in\mathcal{C}, \mathbf{x}\neq\mathbf{y}} \Delta(\mathbf{x}, \mathbf{y})$. We call $\mathcal{C}$ a *(q-ary) linear error-correction code* if $\Sigma = \mathbb{F}_q$ and $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$ with $\dim \mathcal{C} = k$. Finally, we always assume $d = 2t+1$ where t is the *unique decoding distance*. This allows for the unique correction of up to t errors.

Given the ECC $\mathcal{C}$ over alphabet Σ, we denote encoding of a message $\mathbf{m} \in \Sigma^k$ by $\mathcal{C}.\mathsf{Enc}(\mathbf{m})$ that outputs the valid codeword. Similarly, given a word $\mathbf{z} \in \{0,1\}^n$, we denote the error-correction procedure as $\mathcal{C}.\mathsf{Dec}(\mathbf{z})$.

An important class of ECCs that yields parameters $(2^m, 2^m - mt, 2t+1)$ and which we further extensively rely on is the *Goppa codes*. We select this ECC family for two reasons: *(i)* unlike Reed-Solomon codes, which are inherently non-binary (*i.e.*, defined over $\mathbb{F}_q$ with $q > 2$), Goppa codes provide a strong and well-suited ECC defined directly over $\mathbb{F}_2$, and *(ii)* parameter t is clearly determined by code parameters, unlike popular families like LDPC, where determining the true minimum distance is computationally infeasible. Let us introduce Goppa codes formally.

Definition 2.1. Consider the finite field $\mathbb{F}_{q^m}$ and a *locator set* $\mathcal{L} = \{\alpha_i\}_{i\in[n]} \subseteq \mathbb{F}_{q^m}$. Fix an irreducible polynomial $g \in \mathbb{F}_{q^m}[z]$ with $\deg g = t$. **Goppa code** Γ over $\mathcal{L}$ and g is defined as the set of all $\mathbf{c} \in \mathbb{F}_q^n$ such that:

$$\sum_{i\in[n]} \frac{c_i}{z - \alpha_i} = 0 \quad (\text{over } \mathbb{F}_{q^m}[z]/\langle g \rangle)$$

Further, always assume $q = 2$, and m is determined based on the factors size. To see more details on Goppa codes (such as decoding procedure), refer to Appendix A.

2.3. EMBLEM Error-Tolerant Encryption

As a modification of an existing LWE scheme, the EMBLEM key encapsulation mechanism is designed to handle better small secret distribution, while setting large standard deviation of the discrete Gaussian distribution. This makes range of possible outcomes from sampling larger standard deviation, and decreases the probability

of correctly obtaining the result, while using the classic LWE-based KEM schemes (such as Kyber). For this specific reason, EMBLEM KEM has it's own error-tolerant encryption to work in this setup effectively. We proceed to use it's simpler version of the encryption, which captures the main idea behind the method.

The Emblem error-tolerant encryption with parameters (d, q) and some message $\mathbf{m} \in \{0,1\}^n$ consists of these two procedures: $\mathsf{Encrypt}(\mathbf{m}) \to \mathbb{Z}_q^n$ and $\mathsf{Decrypt}(\mathbf{z} \in \mathbb{Z}_q^n) \to \{0,1\}^n$. For the sake of simplicity, we will denote the usage of this encryption by $\mathcal{E}$.

The encryption procedure is constructed as follows: for every of bit of the message $\mathbf{m}$ append $1||0^d$, where $d = \log_2(q) - 2$, and obtain the vector of $d+2$ bit numbers the size n.

The decryption procedure is the following: For every element of the encrypted message input $\mathbf{z} \in \mathbb{Z}_q^n$, we represent the number in bit form. Then find the first significant bit for every element of $\mathbf{z}$, and from them construct the message. The decryption will output in the correct result, iff $\|\mathbf{z}\|_\infty < 2^d$.

As we can see, the larger parameter d, the better the error-tolerance will be. Subsequently, this means that q must also be larger.

2.4. Image Recognition Neural Networks

Before defining fuzzy extractors, we shall discuss technical details on how one obtains witness, corresponding to the biometric data. We demonstrate this using an example of Face Recognition, but the idea used in other biometric data (such as fingerprints [13] of voice [2]) is essentially the same.

2.4.1. Face Recognition Model. The main problem encountered in the face recognition research is as follows: given two facial images, say, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{W \times H \times C}$, determine whether they correspond to the same person. Comparing two objects in the high-dimensional space (of dimensionality $W \cdot H \cdot C$) is vastly suboptimal. A more sufficient approach is therefore to train the feature extraction neural network $f : \mathbb{R}^{W \times H \times C} \to \mathbb{R}^n$ that projects the high-dimensional sample into a much lower-dimensional space $\mathbb{R}^n$, where n typically ranges from 128 to 1024. We further call the output of f an *embedding*.

What do we expect from such a projection? We want to obtain the following two properties: (1) given two images of the same person, say, $\mathbf{x}$ and $\mathbf{x}'$, the distance between embeddings $f(\mathbf{x})$ and $f(\mathbf{x}')$ is *small*, while (2) given two images of two different people, say, $\mathbf{x}$ and $\mathbf{y}$, we expect that distance between $f(\mathbf{x})$ and $f(\mathbf{y})$ would be *large*. Of course, due to the probabilistic and continuous nature of neural networks, we cannot expect that $f(\mathbf{x})$ and $f(\mathbf{x}')$ will match perfectly. Instead, we define the *threshold* $\varepsilon \in \mathbb{R}_{>0}$ and consider images $\mathbf{x}$ and $\mathbf{x}'$ to be of the same person iff $d(f(\mathbf{x}), f(\mathbf{x}')) \le \varepsilon$.

So how do we train f? Some early approaches [4], [42], such as *Eigenface* or *Fisherface*, proposed to use the Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) methods to learn the linear embedding $f(\mathbf{x}) = \Pi\mathbf{x} + \boldsymbol{\beta}$. However, better results, obtained e.g. by [9], [29], [38], [44], proposed to represent f as a neural network that outputs features on the unit d-dimensional hypersphere $\mathbf{S}^{n-1} \triangleq \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 = 1\}$. While all the mentioned papers employ different yet similar methods, the general idea is to construct a loss function that reduces the intra-class average distance while increasing the inter-class distance. We drop the specifics here, so further assume that we have trained the *feature extractor* $f : \mathbb{R}^{W \times H \times C} \to \mathbb{R}^n$.

2.4.2. Embeddings Quantization. As mentioned in the previous section, the neural network f outputs the n-dimensional vector on the hypersphere $\mathbf{S}^{n-1}$. Unfortunately, in cryptographic applications, operating with real-valued is troublesome. To resolve this issue, we map continuous values to the vector of elements from the finite alphabet Σ using what we call a *quantization method*. For simplicity, assume $\Sigma = \mathbb{Z}_r$ where r is the alphabet's size. Given the output of the neural network $\mathbf{v} = f(\mathbf{x}) \in \mathbf{S}^{n-1}$, we map it to $q(\mathbf{v})$ where $q : \mathbf{S}^{n-1} \to \mathbb{Z}_r^n$ is the *quantization function*. Below, we specify functions $q(\cdot)$ used to quantize the real-valued vector.

Binary Naive Embedding. This approach was used, in particular, in [27]. Here, the alphabet is binary, so $\Sigma = \mathbb{Z}_2$. Upon receiving the embedding $(v_1, \ldots, v_n) \in \mathbf{S}^{n-1}$, the output is the binary vector $(b_1, \ldots, b_n) \in \{0,1\}^n$ where $b_i = \mathbb{1}[v_i \ge 0]$ with $\mathbb{1}[\cdot]$ being the indicator function. A slight modification of this approach is to estimate the average values of each embedding component: say, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$, and form b_i as $b_i := \mathbb{1}[v_i \ge \mu_i]$.

Equal-probability quantization. This approach was used, in particular, in [12]. This method partitions the range of feature values into r bins such that each bin has approximately an equal probability mass under the empirical feature distribution. More specifically, assume that ξ_i is the random variable representing the i^{th} component of the neural network output $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ with the marginal distribution $p_i(x)$. Partition the interval $[-1, 1]$ into q bins $\{(\tau_{i,j}, \tau_{i,j+1})\}_{j \in [r]}$ with the property that $\int_{(\tau_{i,j}, \tau_{i,j+1})} dp_i = \frac{1}{r}$ for each bin $(\tau_{i,j}, \tau_{i,j+1})$. Upon receiving the feature vector $(v_1, \ldots, v_n) \in \mathbf{S}^{n-1}$, we convert it to the quantized vector $(\widehat{v}_1, \ldots, \widehat{v}_n) \in \mathbb{Z}_r^n$ such that $v_i \in (\tau_{i,\widehat{v}_i}, \tau_{i,\widehat{v}_i+1})$.

Random Projection. Another prominent approach for binary quantization ($r = 2$) thoroughly theoretically analyzed in [10], [45], [46] is surprisingly simple: quantize the feature vector $\boldsymbol{v} \in \mathbf{S}^{n-1}$ by computing $\widehat{\boldsymbol{v}} := \mathrm{sign}(\Pi\boldsymbol{v})$ where $\Pi \in \mathbb{R}^{m \times d}$ is a random projection matrix generated by sampling each matrix component from the standard normal distribution: $\Pi_{i,j} \sim \mathcal{N}(0,1)$. In particular, [45] proved the following theorem about such a mapping.

Theorem 2.1. *Consider an arbitrary finite set $K \subseteq \mathbf{S}^{n-1}$ of image recognition model outputs, and let the output dimensionality $m \ge (c/\varepsilon^2)|K|$. Let $\widehat{\cdot}$ be the quantized value using random projection $\widehat{\mathbf{x}} = \mathsf{sign}(\Pi\mathbf{x})$. Then, for all $\mathbf{x}, \mathbf{y} \in K$, we have:*

$$Pr\left[|\delta(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}) - d(\mathbf{x}, \mathbf{y})| \le \varepsilon\right] \ge 1 - 2\exp(-\varepsilon^2 m).$$

During our research, for each chosen face recognition model, we benchmark each of the quantization methods and select one that yields the smallest error. Additionally, to simplify notation, we always assume that the neural

network f outputs the element from $\mathbb{Z}_2^n$ as the result of model inference and the subsequent quantization process.

2.4.3. Neural Network Accuracy Estimation. To further model the security level of Unforgettable, we estimate the following parameters:

FRR & FAR (False Rejection Rate & False Acceptance Rate). Fix threshold $\varepsilon \in \mathbb{R}_{>0}$ and the neural network f. Assume ρ_{same} is the distribution of images of the same people, while ρ_{diff} is the distribution of images of different people. Then, we define FRR and FAR as the following quantities:

$$\text{FRR}^f(\varepsilon) = \Pr_{(\mathbf{x},\mathbf{x}')\sim\rho_{\text{same}}}[\Delta(f(\mathbf{x}), f(\mathbf{x}')) > \varepsilon],$$
$$\text{FAR}^f(\varepsilon) = \Pr_{(\mathbf{x},\mathbf{y})\sim\rho_{\text{diff}}}[\Delta(f(\mathbf{x}), f(\mathbf{y})) \le \varepsilon].$$

Empirically, we estimate these parameters as follows: fix the set of pairs of the same people from specifically chosen dataset $\mathcal{P}_{\text{same}} \subseteq (\mathbb{R}^{W\times H\times C})^2$ and similarly of different people $\mathcal{P}_{\text{diff}}$. Then,

$$\text{FRR}^f(\varepsilon) \approx \frac{1}{|\mathcal{P}_{\text{same}}|} \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}_{\text{same}}} \mathbb{1}[\Delta(f(\mathbf{x}), f(\mathbf{x}')) > \varepsilon],$$
$$\text{FAR}^f(\varepsilon) \approx \frac{1}{|\mathcal{P}_{\text{diff}}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{P}_{\text{diff}}} \mathbb{1}[\Delta(f(\mathbf{x}), f(\mathbf{y})) \le \varepsilon].$$

These two parameters are the crucial indicators of the neural network accuracy and security. The larger values of FRR result in a worse user experience, while larger FAR reduces the security level of the neural network. In fact, regardless of the further fuzzy extractor construction, the upper bound of the security of the single biometric source is roughly limited by $\log_2(1/\text{FAR}^f(\varepsilon))$ bits where ε is the threshold below which two feature vectors map to the same secret value. Indeed, the adversary might grind through approximately $\log_2(1/\text{FAR}^f(\varepsilon))$ biometric samples to get the matching feature vector. A slightly more rigorous justification is given in **??**. Based on modern studies and our experiments, currently it is possible to achieve up to ≈ 20 bits of FAR while still having a FRR in the range 0.1–0.2.

Average Separation. To understand how much two feature vectors differ, given they come from the same or different people, we introduce *average separation* values as follows:

$$d^+(f) := \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim\rho_{\text{same}}}[\Delta(f(\mathbf{x}), f(\mathbf{x}'))],$$
$$d^-(f) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\rho_{\text{diff}}}[\Delta(f(\mathbf{x}), f(\mathbf{y}))].$$

They are estimated in a similar manner. These two values will be used to model the likelihood that two individual elements in the feature vector differ, thereby understanding the robustness and security of the fuzzy extractor. For an "ideal" neural network, we expect d^+ to be as close as possible to 0 while d^- to be approximately $\frac{1}{2}$. While the latter condition is typically observed in modern neural networks, the value of d^+ is in practice close to $\frac{1}{4}$, as our experiments show. For concrete distribution of distances, refer to Figure 1. In particular, similar results were obtained in [27].

Datasets. For benchmarking models, we use LFW [18] and MS1M-V2 [9], [15], [30] datasets. These datasets are currently the most widely used in Face Recognition research and provide a comprehensive list of more than a million images with varying lighting conditions, races, or poses.

We provide benchmark results in Table 1.

3. Fuzzy Extractors

3.1. Entropies

Definition 3.1. Let $\mathcal{X}$ be the distribution defined over the set X. The **min-entropy** of $\mathcal{X}$ is defined as

$$\mathbf{H}_\infty(\mathcal{X}) \triangleq -\log_2\left(\max_{x\in X}\Pr[\mathcal{X}=x]\right).$$

Given the joint distribution $\mathcal{X}\times\mathcal{Y}$ over $X\times Y$, we define the **average min-entropy** of $\mathcal{X}$ given $\mathcal{Y}$ as:

$$\mathbf{H}_\infty(\mathcal{X}|\mathcal{Y}) \triangleq -\log_2\left(\mathbb{E}_{y\leftarrow\$\mathcal{Y}}\left[\max_{x\in X}\Pr[\mathcal{X}=x|\mathcal{Y}=y]\right]\right)$$

3.2. Sources and Factors

As mentioned in § 1.1, the factor notion introduced in MFKDF cannot be directly implied to fuzzy extractor-based factors. Specifically, we cannot assume that each factor is naturally equipped with some factor construction. For this reason, we introduce the new notion of factors that we can use for formal security analysis. Further assume that $\mathcal{U}$ is the finite set of all users.

Definition 3.2. Source $\mathtt{S}$ over the set of possible sample values X is a tuple $(X, \Phi, (\Phi_U)_{U\in\mathcal{U}})$ where:

- (X, d_X) is the discrete metric space.
- Φ is the overall population distribution over X.
- Φ_U is the probability distribution of the biometric data of the given user U over X.

We say that Φ_U is the **factor** of user U over $\mathtt{S}$.

Example 3.1. Facial data source $\mathtt{S}_F$ is defined over quantized embeddings $X = \{0,1\}^n$. Then, the factor $\Phi_U^{(\mathtt{S}_F)}$ is the distribution of embeddings of the person U. Similarly, we can associate uniform n-bit password data source $\mathtt{S}_P$ with $\Phi^{(\mathtt{S}_P)}$ being a uniform distribution over $\{0,1\}^n$, and $\Phi_U^{(\mathtt{S}_P)}$ is the distribution with the support of a single element (corresponding to user's password). Note that in practice $\Phi^{(\mathtt{S}_P)}$ is not uniform as users do not generate passwords uniformly randomly.

Example 3.1 motivates us to distinguish *stable sources* (such as passwords or seed phrases) and *unstable sources* (such as biometric data).

Definition 3.3. We say that the source $(X, \Phi, (\Phi_U)_{U\in\mathcal{U}})$ is **stable** if $\text{supp}(\Phi_U) = \{x_U\}$ for some $x_U \in X$ for each $U \in \mathcal{U}$. Otherwise, the source is called **unstable**.

Remark. When saying "m-entropy source", we mean that the entropy of a global distribution Φ is m. Indeed, the difficulty of breaking password/biometric lies in how much entropy we expect from the global distribution. The entropy of the user distribution Φ_U is merely a measure of variance: e.g., entropy of a user distribution when the source is stable is always 0.

TABLE 1: Modern neural networks accuracy metrics results based on LFW [18] and MS1MV2 [9], [15], [30].

		Random Projection				Equal-probability			
Recognition Model f	**Dataset**	$d^+(f)$	$d^-(f)$	ε	$\mathrm{FAR}^{\mathsf{S}}(\varepsilon)$	$d^+(f)$	$d^-(f)$	ε	$\mathrm{FAR}^{\mathsf{S}}(\varepsilon)$
ArcFace [9]	LFW	0.2463	0.4969	0.1679	$2^{-22.72}$	0.2456	0.4970	0.1699	$2^{-22.98}$
MagFace [32]	LFW	**0.2230**	0.4829	0.1445	$2^{-22.72}$	**0.2202**	0.4835	0.1464	$2^{-22.72}$
AdaFace [26]	LFW	0.2423	0.4998	0.1601	$2^{-22.72}$	0.2526	0.4998	0.1718	$2^{-22.49}$
CurricularFace [19]	LFW	0.2266	0.4988	0.1386	$2^{-22.72}$	0.2259	0.4979	0.1367	$2^{-22.72}$
ArcFace [9]	MS1MV2	0.2676	0.4932	0.0898	$2^{-28.70}$	0.2678	0.4938	0.0898	$2^{-28.70}$
MagFace [32]	MS1MV2	0.2393	0.4879	0.0781	$2^{-27.12}$	0.2379	0.4881	0.0781	$2^{-27.70}$
AdaFace [26]	MS1MV2	0.2715	**0.4999**	0.0839	$2^{-27.70}$	0.2705	**0.4999**	0.0859	$2^{-27.70}$
CurricularFace [19]	MS1MV2	0.2384	0.4996	0.0761	$2^{-27.70}$	0.2370	0.4996	0.0761	$2^{-28.70}$

Definition 3.4. We say that two sources (and factors, respectively) are **independent** if the corresponding global and user distributions are independent.

Definition 3.5. With the source S, similarly to § 2.4.2, we associate **false acceptance** and **false rejection** rates depending on threshold $\varepsilon \in \mathbb{R}_{>0}$ as follows[1]:

$$\mathrm{FAR}^{\mathsf{S}}(\varepsilon) = \Pr_{(\mathbf{w},\mathbf{w}')\sim\Phi_U\times\Phi_{U'}}[d_X(\mathbf{w},\mathbf{w}') \le \varepsilon],$$
$$\mathrm{FRR}^{\mathsf{S}}(\varepsilon) = \Pr_{\mathbf{w},\mathbf{w}'\sim\Phi_U}[d_X(\mathbf{w},\mathbf{w}') > \varepsilon].$$

3.3. Fuzzy Extractors

The vast majority of cryptographic authentication algorithms, one way or another, rely on the following rule: if the user knows (can reproduce) the *exact* secret value w, they are granted access to the system. The notion of *fuzzy extractors*, first introduced by Dodis [11], allows us to define less strict rules where the user might know a close, yet not necessarily identical secret value w' such that if $\Delta(w, w')$ is small enough, the user can still access the system. As discussed previously, the secret value w is the embedding vector extracted from a source (e.g., a face or physical identifiers). Now, we define the *fuzzy extractor scheme* more formally.

Definition 3.6. The **Fuzzy Extractor Scheme** Π_{FE} over unstable source $\mathsf{S} = (X, \Phi, (\Phi_U)_{U\in\mathcal{U}})$ is a tuple $(\mathsf{Gen}, \mathsf{Rep})$ that satisfies the following properties:

- $\mathsf{Gen}(1^\lambda, \mathbf{w}) \to (\mathsf{hs}, \mathsf{sk})$. Upon receiving the user's biometric sample $\mathbf{w} \in X$, the fuzzy extractor generates the *helper string* hs and *secret value* sk.
- $\mathsf{Rep}(\mathbf{w}', \mathsf{hs}) \to \mathsf{sk}$. Upon receiving the new user's biometric sample $\mathbf{w}' \in X$ and the helper data hs, the fuzzy extractor outputs the secret value sk.

We require the (δ, τ)**-correctness** property: for any biometric samples $\mathbf{w}, \mathbf{w}' \leftarrow\!\$\ \Phi_U$ sampled from some user's distribution, the probability of successfully recovering the secret key is:

$$\Pr\left[\mathsf{Rep}(\mathbf{w}', \mathsf{hs}) = \mathsf{sk} \;\middle|\; \begin{matrix}(\mathsf{hs}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}) \\ (\mathbf{w}, \mathbf{w}') \leftarrow\!\$\ \Phi_U\end{matrix}\right] \ge 1 - \delta$$

Additionally, the probability of reproducing the same key using two biometric samples of different people: $(\mathbf{w}, \mathbf{w}') \leftarrow\!\$\ \Phi_U \times \Phi_{U'}$, $U \ne U'$, is bounded above by another value which we call τ:

$$\Pr\left[\mathsf{Rep}(\mathbf{w}', \mathsf{hs}) = \mathsf{sk} \;\middle|\; \begin{matrix}(\mathsf{hs}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}) \\ (\mathbf{w}, \mathbf{w}') \leftarrow\!\$\ \Phi_U \times \Phi_{U'} \\ U \ne U'\end{matrix}\right] \le \tau$$

Definition 3.7 (Secure Fuzzy Extractor)**.** For adversary algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, define the following advantage value for any user U:

$$\mathrm{Adv}^{\Pi_{\mathrm{FE}},\mathrm{Sec}}_{\mathcal{A}}(\lambda) \triangleq 2\cdot\left|\Pr\left[b = \widehat{b} \;\middle|\; \begin{matrix}\mathbf{w} \leftarrow\!\$\ \Phi_U \\ (\mathsf{hs}, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}) \\ \mathsf{sk}_1 \leftarrow\!\$\ \{0,1\}^\ell, b \leftarrow\!\$\ \{0,1\} \\ \widehat{b} \leftarrow \mathcal{A}_1(\mathsf{hs}, \mathsf{sk}_b)\end{matrix}\right] - \frac{1}{2}\right|$$

We say that the Fuzzy Extractor Π is **ε-secure** against secret value recovery if for any efficient adversary $\mathcal{A}$, one has $\mathrm{SAdv}[\mathcal{A}, \Pi] \le \varepsilon$. Intuitively, the helper data hs reveals a small amount of information (namely, $\log_2 \frac{1}{\varepsilon}$ bits) about the secret sk. In particular, if $\varepsilon = \mu(\lambda)$ is negligible in λ, the Fuzzy Extractor scheme Π is computationally secure.

Beside mentioned properties, we also need to define the security of the fuzzy extractor when the secret key sk is revealed with the helper string hs, and how much information it will reveal about the secret values that will be created from the same feature vector source. In practice, we want the secret keys to be almost uniform and do not reveal the structure of the feature vector $\mathbf{w}$.

This property called **reusability**, and we define it in the terms of the game $\mathcal{G}$, such that: there are the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$, and they interact in the following way:

- $\mathcal{C}$ generates the pair $(\mathsf{hs}_0, \mathsf{sk}_0)$ from $\mathbf{w}_0 \leftarrow\!\$\ \Phi_u$, and sends it to the adversary hs_0.
- Then the adversary creates Q adaptive queries and sends $\delta_i \in \{0,1\}^n : \Delta(\delta_i, 0) \le t$ to the challenger to generate Q responses of pairs $(\mathsf{hs}_i, \mathsf{sk}_i)_{i\in[Q]}$.
- Finally, the challenger $\mathcal{C}$ randomly samples the value $b \in \{0,1\}$. If the result is $b = 1$, then $\mathcal{C}$ send to $\mathcal{A}$ the value sk_0, or if $b = 0$, then the challenger outputs $u \in \{0,1\}^\mu$, which is uniform.
- The adversary uses given values to deduce if the final key is from the extractor or a random uniform value, with the guess b'. If $b' = b$ — the adversary wins.

Then we can define the **reusability** of the fuzzy extractor from the game $\mathcal{G}$: the fuzzy extractor is a ε_{r}-reusable if:

1. Formally, we should additionally write that users U and U' are sampled randomly from $\mathcal{U}$. To shorten the notation, we will not write this explicitly, but this is always automatically assumed.

$$\left|\Pr\left[b = b'\right] - \frac{1}{2}\right| \leq \varepsilon_{\mathrm{r}}$$

Another property used for define the security of the fuzzy extractor is the **robustness**. The robustness property guarantees that the tampering of the helper string will be detected, and the procedure of restoration will return $\perp$. In the case of KDF, we don't need the robustness property, because fuzzy extractor must to output something as the result of restoration.

Summary. This way, the fuzzy extractor scheme Π properties are determined by four parameters: $(\delta, \tau, \varepsilon, \varepsilon_r)$, where the first parameter δ measures the false rejection rate, the second τ — false acceptance rate, and the rest ε and ε_r — security of the scheme Π_{FE}.

3.4. Threat Model

We shall now discuss how to estimate the aforementioned Fuzzy Extractor parameters ((δ, τ)-*correctness* and ε-*security*) practically. Compared to security analysis of most classical cryptography primitives, where derivations are based on certain assumptions (such as the Discrete Log or Computational Diffie-Hellman assumptions), we will introduce several attack scenarios and test our definitions against these models. We propose the following three models:

Threat Model #1. The PPT adversary $\mathcal{A}_1$ views the helper string $\mathsf{hs} = (I, V)$ and tries to derive the correct secret entropy β corresponding to hs. The probability of doing this successfully gives the value of ε-security. However, during attack, $\mathcal{A}_1$ *does not access neural networks or any other input data (e.g., biometric)*. This is the weakest assumption for basic protocol testing, and we typically expect ε to be negligible in this setting (or at least of the value of FAR).

Threat Model #2. The PPT adversary $\mathcal{A}_2$ views the helper string hs and without knowing the user's input, tries to achieve the same goals as in the Threat Model #1, but they are free to sample images from the distribution (e.g., take facial images from the dataset) and run the neural network over them. This is a quite natural and reasonable model, which provides one of the most optimal attack vectors.

Threat Model #3. The PPT adversary $\mathcal{A}_3$ is the same as in Threat Model #2, but they additionally know the identity of user possessing the helper string hs (*e.g.*, they know the user's facial photo).

Finally, we need to model the capabilities of the adversary to break our system. When working with stable cryptographic primitives such as digital signatures or hash functions, we either pass the verification or not. In case of biometric authentication, we have: (a) different probabilities of achieving proximity to the "correct" feature value, (b) different probabilities of passing verification with the given proximity. For that reason, we introduce Assumption 3.1, which states the capabilities of the adversary.

Proposition 3.1. *Assume, given the neural network f, the user's U biometrics distribution Φ_U, and any biometric sample $\mathbf{w} \leftarrow\!\$\ \Phi_U$ from this distribution, the adversary $\mathcal{A}$ can output the ε-close biometric sample $\mathbf{w}'$ (for $\varepsilon = k/n$ with $0 \leq k \leq n$) by calling the oracle $\mathcal{O}^f(\cdot)$ with probability $\rho^-(\varepsilon)$. More concretely,*

$$Pr\left[\Delta(\mathbf{w}, \mathbf{w}') = \varepsilon \;\middle|\; \begin{array}{c} \mathbf{w} \leftarrow\!\$\ \Phi_U \\ \mathbf{w}' \leftarrow \mathcal{O}^f(\cdot) \end{array}\right] = \rho^-(\varepsilon)$$

In particular, cumulative distribution function (CDF) of $\rho^-(\varepsilon)$ is given by FAR$^f(\varepsilon)$.

This assumption is reasonable: if FAR$^f(\varepsilon)$ of the neural network f is, say, 2^{-20} (corresponding to modern face recognition neural networks), then by sampling 2^{20} biometric samples, the adversary is very likely to sample $\mathbf{w}'$ such that $\Delta(\mathbf{w}', \mathbf{w}) \leq \varepsilon$. In particular, it is reasonable to expect that the cumulative probability of getting distance below ε is $\sum_{k=0}^{\varepsilon n} \rho^-(k/n) \approx \mathrm{FAR}^f(\varepsilon)$.

In turn, for the honest user, we have the following assumption.

Proposition 3.2. *Assume, given the neural network f, the user's U biometric distribution Φ_U, and any two biometric samples $\mathbf{w}, \mathbf{w}' \leftarrow\!\$\ \Phi_U$ from this distribution, the probability of them being ε-close is determined with probability $\rho^+(\varepsilon)$:*

$$Pr[\Delta(\mathbf{w}, \mathbf{w}') = \varepsilon \mid \mathbf{w}, \mathbf{w}' \leftarrow\!\$\ \Phi_U] = \rho^+(\varepsilon).$$

In particular, CDF of $\rho^+(\varepsilon)$ is given by TAR$^f(\varepsilon)$ (true acceptance rate).

These two distributions estimated on the real data are depicted in Figure 1.

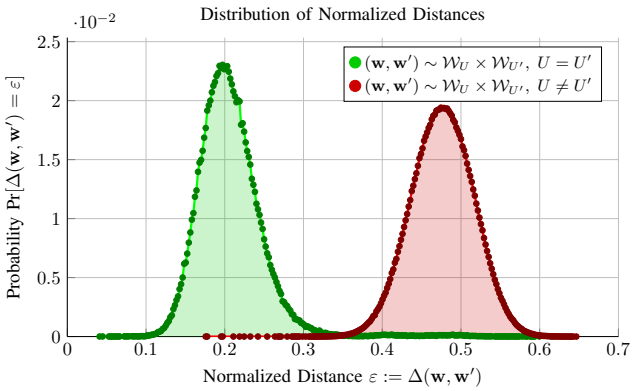

Figure 1: Distribution of distances: one distribution $\rho^+(\varepsilon)$ (marked in green) corresponds to probability of getting given distances between *same* people, while the other $\rho^-(\varepsilon)$ (marked in red) corresponds to distances between different people. As can be seen, proximity parameters (expected values of each distribution) are approximately $d^+(f) \approx 0.20$ and $d^-(f) \approx 0.47$, respectively.

Based on the aforementioned assumptions and propositions, we show the following property of the original X-Lock construction.

3.5. Previous Constructions

3.5.1. Code-Based Fuzzy Extractor. Since their introduction, multiple constructions and variations of fuzzy extractors have been proposed. One interesting class of fuzzy extractors relies on error-correcting codes [17], [23], such as Reed-Solomon codes [36], BCH [7], or Goppa

codes considered in § 2.2. As an example of the concrete fuzzy extractor construction, consider the *code-offset construction*, originally introduced in [11].

Definition 3.8 (Code-Offset Fuzzy Extractor)**.** Fix (n, k, d) linear error-correction code $\mathcal{C} \subseteq \{0,1\}^n$ and a random oracle $\mathcal{H} : \{0,1\}^n \to \{0,1\}^\mu$. Then, the fuzzy extractor $\Pi_\mathcal{C}$ is constructed as follows:

- $\mathbf{Gen}(1^\lambda, \mathbf{w} \in \{0,1\}^n) \to (\mathsf{hs} \in \{0,1\}^n, \mathsf{sk} \in \{0,1\}^\mu)$. Generate a random message $\mathbf{m} \leftarrow\!\$\ \{0,1\}^k$ and encode it to get a random codeword $\mathbf{c} \leftarrow \mathcal{C}.\mathsf{Enc}(\mathbf{m})$. Form the helper data as $\mathsf{hs} := \mathbf{w} \oplus \mathbf{c}$ while the secret data is $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{c})$. Output $(\mathsf{hs}, \mathsf{sk})$.
- $\mathbf{Rep}(\mathbf{w}' \in \{0,1\}^n, \mathsf{hs} \in \{0,1\}^n) \to \mathsf{sk}$. Compute the corrupted codeword $\mathbf{c}' \leftarrow \mathsf{hs} \oplus \mathbf{w}'$. Decode the codeword to get $\mathbf{c}$ and output $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{c})$.

Let us now consider some of the properties of such a construction. First, it is easy to see that if the biometric samples $\mathbf{w}, \mathbf{w}'$ are close enough (namely, less than t), we can always reproduce the valid key:

Theorem 3.1. *The function* $\Pi_\mathcal{C}.\mathsf{Rep}(\mathbf{w}')$ *always reproduces the valid key if* $\Delta(\mathbf{w}, \mathbf{w}') \le t$.

Reasoning. Notice that $\mathbf{c}' = \mathsf{hs} \oplus \mathbf{w}' = \mathbf{c} \oplus (\mathbf{w} \oplus \mathbf{w}')$. Therefore, $\Delta(\mathbf{c}', \mathbf{c}) = \Delta(\mathbf{w}, \mathbf{w}')$. Since by the condition in lemma we have $\Delta(\mathbf{w}, \mathbf{w}') \le t$, then $\Delta(\mathbf{c}', \mathbf{c}) \le t$, so a decoding of $\mathbf{c}'$ would yield $\mathbf{c}$.

However, for $(\mathbf{w}, \mathbf{w}') > t$, the decoding procedure becomes exponentially difficult. Consider the following theorem to illustrate this.

Theorem 3.2. *Probability that* $\mathcal{C}.\mathsf{Dec}(\mathbf{c}')$ *would succeed in decoding to some valid codeword is negligible (both in values of* t *and* n*)*

Proof. The code $\mathcal{C}$ contains 2^k valid codewords. The decoding procedure for $\mathbf{c}'$ succeeds if and only if $\Delta(\mathbf{c}, \mathbf{c}') \le t$ for some valid $\mathbf{c} \in \mathcal{C}$. Since there are 2^k valid codewords, the decoding would succeed only if $\mathbf{c}'$ happens to be in the t-ball of some codeword. The total volume of these balls is $2^k \cdot V(n,t)$ where $V(n,t)$ is the volume of the Hamming ball of radius t. Thus, the searched probability is:

$$\frac{2^k \cdot V(n,t)}{2^n} \approx \frac{t \cdot 2^k}{t!} \cdot 2^{-n} n^t = \mathrm{negl}(t,n)$$

Code-offset construction is highly dependent on the source's min-entropy. In our case, we require more than $\frac{n}{2}$ bits of the min-entropy, but in reality, the upper bound on min-entropy from the facial features is less than 40 bits. Moreover, the constructions that use the secure sketch will inevitably have information leakage, because the parity-check matrix is public, and consists of $n-k$ equations with n variables. At first sight, we have the number of solutions equal to k. However, including the min-entropy of the source, we have a small number of distinct feature vectors, which eventually reduces the dimensionality of the system of equations from the parity-check matrix. The remaining min-entropy from the scheme m' can be bounded by starting nim-entropy m from the feature distribution and the leakage $n-k$: $m' = m - (n-k)$.

3.5.2. Fuzzy Vault. Another interesting approach, so-called **Fuzzy Vault**, was proposed by Juels and Sudan [21]. The fuzzy vault is based on Shamir's secret sharing scheme [39] and was originally designed for unordered biometric feature sets. In this scheme, a secret is encoded as an evaluation of a polynomial of degree $t-1$. Polynomial is evaluated in all n features. The user who can reproduce t-of-n features can find the correct evaluations and interpolate the polynomial. Randomly generated garbage points obscure the vault against brute-force attacks. Specifically, the scheme is constructed as follows.

Definition 3.9 (Fuzzy Vault)**.** Fix parameters (n, k, r, t) where n is the biometric sample size, r is the size of helper data, k is the secret size, and $t-1$ is the polynomial degree. Then, the fuzzy extractor $\Pi_\mathcal{V}$ over the Reed-Solomon code $\mathsf{RS}[\mathbb{F}_q, n, t]$ is constructed as follows.

- $\mathbf{Gen}(1^\lambda, \mathbf{w} \in \mathbb{F}_q^n) \to (\mathsf{hs} \in \mathbb{F}_q^r, \mathsf{sk} \in \mathbb{F}_q^k)$. Generate a random polynomial $f(X) \in \mathbb{F}_q^{<t}[X]$. Set secret value to $\mathsf{sk} \leftarrow \{f(\alpha)\}_{\alpha \in \mathcal{L}}$ where $\mathcal{L} \subseteq \mathbb{F}_q$ is the set of distinct field elements of size k (evaluation domain). Form the helper data as the set of points $\mathsf{hs} = \{(x_i, f(x_i))\}_{i \in [r]}$ where $x_i = w_i$ for $i \in [n]$ and $x_i \leftarrow\!\$\ \mathbb{F}_q$ for all remaining indices $i \in [r] \setminus [n]$. Shuffle array hs.
- $\mathbf{Rep}(\mathbf{w}' \in \mathbb{F}_q^n, \mathsf{hs} \in \mathbb{F}_q^r) \to \mathsf{sk}$. Parse the received helper data hs as $\{(x_i, y_i)\}_{i \in [r]}$. Form an array of points $H := \{(x_i, y_i) \in \mathsf{hs} : x_i \approx w'_j$ for some $j \in [n]\}$[2]. Run the Reed-Solomon decoding algorithm over H to get polynomial $g \in \mathbb{F}^{<t}[X]$ and output $\mathsf{sk} \leftarrow \{g(\alpha)\}_{\alpha \in \mathcal{L}}$.

Despite the promise of this approach, it introduces a significant challenge related to transforming features into field elements. While in § 2.4.2 we described possible ways to construct such a map, the Fuzzy Vault construction must have a relatively large q (at least of characteristic n).

Other notable candidates. Other approaches include metric embedding techniques [11] and specialized adaptations for biometric authentication [40] and physical unclonable functions (PUFs) [31]. These architectures have broadened the applicability of fuzzy extractors across secure storage, authentication, and key generation.

3.5.3. LWE Code-Based Fuzzy Extractor. As described in the section § 3.5.1, fuzzy extractors that are built on the secure sketch paradigm require a high min-entropy input, which we could not afford because most of the available to us fuzzy sources have low min-entropy, which makes the scheme broken. Even having 128 bits of min-entropy from the combined feature vectors, we still need a large parameter t to correct the errors. This results in the negative entropy of the scheme.

For this specific reason, Fuller, Meng, and Reyzin [5] showed that there is no computational secure sketch that is more efficient than information-theoretical, which is based on ECC. They also proposed a new construction, which is based on the computational hardness of a random linear code, and has way less leakage than the constructions based on the secure sketch. But their construction has low error correction capability, which isn't practical for our purposes.

2. Here, $x_i \approx w'_j$ can be either interpreted as the precise equality or, say, if $|x_i - w'_j|/q < \delta$ for some $\delta \in [0,1]$.

Another fuzzy extractor based on the LWE problem described in [**?**]. This construction utilizes another approach to correcting errors using ECC with leakage-resistance by adding the codeword to an LWE instance with a secret given by the feature vector and fresh noise, instead of masking the feature vector in the LWE instance. This construction gives us the ability to correct more errors, while relying on the hardness of the LWE problem (more accurately — Non-uniform LWE). The construction is defined as follows:

Definition 3.10 (NLWE Code-Offset Fuzzy Extractor)**.** Consider (m, k, d) linear error-correction code $\mathcal{C} \subseteq \{0,1\}^m$, (q, h) error-tolerant encryption *EMBLEM* $\mathcal{E} \subseteq \mathbb{Z}_q^m$ and a random oracle $\mathcal{H} : \{0,1\}^m \to \{0,1\}^\mu$. Let $\eta = \left[-\frac{\sqrt{q}}{2}; \frac{\sqrt{q}}{2}\right]$, where q is a modulus of an NLWE instance. Then, the fuzzy extractor $\Pi_{\mathcal{C}_{\text{NLWE}}}$ is constructed as follows:

- $\mathsf{Gen}(1^\lambda, \mathbf{w} \in \mathbb{Z}_q^n) \to (\mathsf{hs} = (A, b) \in \eta^{m\times n} \times \mathbb{Z}_q^m, \mathsf{sk} \in \{0,1\}^\mu)$. Sample a matrix $A \hookleftarrow\$\ \eta^{m\times n}$ and the secret error term $\mathbf{e} \in \mathbb{Z}_q^m$ from discrete Gaussian distribution. Generate a random value (message) $\mathbf{r} \hookleftarrow\$\ \{0,1\}^k$ and encode it to get a random codeword in $\mathbb{Z}_q^m$: $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{r}) = \mathcal{E}.\mathsf{Encrypt}(\mathcal{C}.\mathsf{Enc}(\mathbf{r}))$. Evaluate $b = A\mathbf{w} + \mathbf{e} + \mathbf{c} \in \mathbb{Z}_q^m$. Form the helper data as $\mathsf{hs} = (A, b)$, while the secret data is $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{r})$. Output $(\mathsf{hs}, \mathsf{sk})$.
- $\mathsf{Rep}(\mathbf{w}' \in \mathbb{Z}_q^n, \mathsf{hs} = (A, b) \in \eta^{m\times n} \times \mathbb{Z}_q^m) \to \mathsf{sk}$. Evaluate $\mathbf{c}' = b - A\mathbf{w}'$. Decode the corrupted codeword to get $\mathbf{r} \leftarrow \mathsf{Dec}(\mathbf{c}') = \mathcal{C}.\mathsf{Dec}(\mathcal{E}.\mathsf{Decrypt}(\mathbf{c}'))$, and output $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{r})$.

3.6. Unforgettable Fuzzy Extractor

3.6.1. Keyed LWE-based Fuzzy Extractor. After analyzing current suitable fuzzy extractor constructions, we adapt and modify the NLWE code-offset construction to use multiple sources (such as stable and unstable factors) to achieve a secure and efficient construction, which requires less min-entropy from all factors combined. As discussed in § 1.1, we want to use all factors directly in the fuzzy extractor construction. At first, we will describe the Keyed LWE fuzzy extractor construction, which we will modify to the Unforgettable fuzzy extractor. The Keyed LWE construction is defined as follows:

Definition 3.11 (Keyed LWE Code-Offset Fuzzy Extractor)**.** Consider the same setup as in the section Definition 3.10. Additionally, we define the distribution of secret keys, which has at least 8 symbols as $\mathcal{K}$ from some dictionary Σ with min-entropy $\lambda_\mathcal{K}$. Then, the fuzzy extractor $\Pi_{\mathcal{C}_\text{U}}$ is constructed as follows:

- $\mathsf{Gen}(1^\lambda, \mathbf{w} \in \mathbb{Z}_q^n, \mathbf{p} \in \mathcal{P}) \to (\mathsf{hs} = (A_1, A_2, b, \mathsf{h}) \in \mathbb{Z}_q^{m\times n} \times \eta^{m\times n} \times \mathbb{Z}_q^m \times \{0,1\}^*, \mathsf{sk} \in \{0,1\}^\mu)$. Sample matrices $A_1 \hookleftarrow\$\ \mathbb{Z}_q^{m\times n}$ $A_2 \hookleftarrow\$\ \eta^{m\times n}$ and the secret error term $\mathbf{e} \in \mathbb{Z}_q^m$ from Gaussian distribution. From the secret key distribution $\mathbf{k}$ obtain the uniform in $\mathbb{Z}_q^n$ the secret value $\mathbf{s} \leftarrow \mathrm{PRF}(\mathcal{H}(\mathbf{k}||\mathsf{h}))$. Generate a random value (message) $\mathbf{r} \hookleftarrow\$\ \{0,1\}^k$ and encode it to get a random codeword in $\mathbb{Z}_q^m$: $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{r}) = \mathcal{E}.\mathsf{Encrypt}(\mathcal{C}.\mathsf{Enc}(\mathbf{r}))$. Evaluate $b = A_1\mathbf{s} + A_2\mathbf{w} + \mathbf{e} + \mathbf{c} \in \mathbb{Z}_q^m$. Form the helper data as $\mathsf{hs} = (A, b)$, while the secret data is $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{r})$. Output $(\mathsf{hs}, \mathsf{sk})$.
- $\mathsf{Rep}(\mathbf{w}' \in \mathbb{Z}_q^n, \mathbf{k} \in \mathcal{K}, \mathsf{hs} = (A_1, A_2, b, \mathsf{h}) \in \mathbb{Z}_q^{m\times n} \times \eta^{m\times n} \times \mathbb{Z}_q^m \times \{0,1\}^*) \to \mathsf{sk}$. Obtain the secret value $\mathbf{s} \in \mathbb{Z}_q^n$. Evaluate $\mathbf{c}' = b - A_1\mathbf{s} - A_2\mathbf{w}'$. Decode the corrupted codeword to get $\mathbf{r} \leftarrow \mathsf{Dec}(\mathbf{c}') = \mathcal{C}.\mathsf{Dec}(\mathcal{E}.\mathsf{Decrypt}(\mathbf{c}'))$ and output $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{r})$.

This construction utilizes the key $\mathbf{k}$ to use the LWE setting, instead of the NLWE, which is more convenient for security analysis and overall security of the scheme. The adversary must somehow obtain *both* the correct key $\mathbf{k}$ *and* the feature vector $\mathbf{w}'$ to successfully recover the secret value $\mathbf{r}$.

References

[1] S2 geometry.

[2] Zhongxin Bai and Xiao-Lei Zhang. Speaker recognition based on deep learning: An overview. *Neural Networks*, 140:65–99, 2021.

[3] Lucas Ballard, Seny Kamara, and Michael K Reiter. The practical subtleties of biometric key generation. In *USENIX Security Symposium*, pages 61–74, 2008.

[4] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[5] Xianrui Meng Benjamin Fuller and Leonid Reyzin. Computational fuzzy extractors. In *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 174–193. Springer, 2013. Also available as Cryptology ePrint Archive, Report 2013/416.

[6] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: The Memory-Hard Function for Password Hashing and Other Applications. *IACR Transactions on Symmetric Cryptology*, 2017(4):43–57, 2017.

[7] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

[8] William E. Burr, Donna F. Dodson, and Elaine M. Barker. Recommendation for password-based key derivation: Part 1: Storage applications. Technical Report NIST SP 800-132, National Institute of Standards and Technology (NIST), December 2010. NIST Special Publication 800-132.

[9] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):5962–5979, October 2022.

[10] Sjoerd Dirksen and Alexander Stollenwerk. Fast binary embeddings with gaussian circulant matrices: improved bounds. *Discrete & Computational Geometry*, 60(3):599–626, 2018.

[11] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology – EUROCRYPT 2004*, pages 523–540. Springer, 2004.

[12] Xingbo Dong, Hui Zhang, Yen Lung Lai, Zhe Jin, Junduan Huang, Wenxiong Kang, and Andrew Beng Jin Teoh. Wifakey: Generating cryptographic keys from face in the wild. *IEEE Transactions on Instrumentation and Measurement*, 2024.

[13] Joshua J Engelsma, Kai Cao, and Anil K Jain. Learning a fixed-length fingerprint representation. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):1981–1997, 2019.

[14] Andreas Erwig, Julia Hesse, Maximilian Orlt, and Siavash Riahi. Fuzzy asymmetric password-authenticated key exchange. In *Advances in Cryptology – ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II*, page 761–784, Berlin, Heidelberg, 2020. Springer-Verlag.

[15] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition, 2016.

[16] Abdenour Hadid. Face biometrics under spoofing attacks: Vulnerabilities, countermeasures, open issues, and research directions. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 113–118, 2014.

[17] Feng Hao, Ross Anderson, and John Daugman. Combining crypto with biometrics effectively. *IEEE Transactions on Computers*, 55(9):1081–1088, 2006.

[18] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[19] Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. Curricularface: Adaptive curriculum learning loss for deep face recognition, 2020.

[20] A. Juels and M. Sudan. A fuzzy vault scheme. In *Proceedings IEEE International Symposium on Information Theory,*, pages 408–, 2002.

[21] Ari Juels and Madhu Sudan. A fuzzy vault scheme. In *IEEE International Symposium on Information Theory (ISIT)*, page 408. IEEE, 2002.

[22] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. *Proceedings of the ACM Conference on Computer and Communications Security*, 1, 12 1999.

[23] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS)*, pages 28–36. ACM, 1999.

[24] Burt Kaliski. Pkcs #5: Password-based cryptography specification version 2.0. RFC RFC 2898, RSA Laboratories, 2000. Request for Comments 2898.

[25] Shuichi Katsumata, Takahiro Matsuda, Wataru Nakamura, Kazuma Ohara, and Kenta Takahashi. Revisiting fuzzy signatures: Towards a more risk-free cryptographic authentication system based on biometrics. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 2046–2065. ACM, November 2021.

[26] Minchul Kim, Anil K. Jain, and Xiaoming Liu. Adaface: Quality adaptive margin for face recognition, 2023.

[27] Oleksandr Kuznetsov, Dmytro Zakharov, and Emanuele Frontoni. Deep learning-based biometric cryptographic key generation with post-quantum security. *Multimedia Tools and Applications*, 83(19):56909–56938, 2024.

[28] Ledger Donjon. Funds of every wallet created with the trust wallet browser extension could have been stolen, April 2023. Ledger blog, accessed 20 August 2025.

[29] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[30] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[31] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. A soft decision helper data algorithm for sram pufs. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2101–2105. IEEE, 2008.

[32] Qiang Meng, Shichao Zhao, Zhida Huang, and Feng Zhou. Magface: A universal representation for face recognition and quality assessment, 2021.

[33] Vivek Nair and Dawn Song. Multi-factor key derivation function (mfkdf) for fast, flexible, secure, & practical key management. In *Proceedings of the 32nd USENIX Conference on Security Symposium*, SEC '23, USA, 2023. USENIX Association.

[34] Marek Palatinus, Pavol Rusnak, Aaron Voisine, and Sean Bowe. Bip-39: Mnemonic code for generating deterministic keys. https://github.com/bitcoin/bips/blob/master/bip-0039. mediawiki, 2013. Accessed: 2025-08-20.

[35] Michael Folkson (question) and Kalle Rosenbaum (accepted answer). Should the bip 39 mnemonic sentence checksum be eliminated from the standard? does it do more harm than good?, December 2020. Bitcoin Stack Exchange, Question #100376, accessed 20 August 2025.

[36] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[37] Colin Roberts, Vivek Nair, and Dawn Song. Wrangling entropy: Next-generation multi-factor key derivation, credential hashing, and credential generation functions, 2025.

[38] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815–823. IEEE, June 2015.

[39] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[40] Y. Sutcu, Qiming Li, and Nasir Memon. Protecting biometric templates with sketch: Theory and practice. *IEEE Transactions on Information Forensics and Security*, 2(3):503–512, 2007.

[41] Kenta Takahashi, Takahiro Matsuda, Takao Murakami, Goichiro Hanaoka, and Masakatsu Nishigaki. Signature schemes with a fuzzy private key. *International Journal of Information Security*, 18(5):581–617, 2019.

[42] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[43] Marie Vasek, Joseph Bonneau, Ryan Castellucci, Cameron Keith, and Tyler Moore. The bitcoin brain drain: Examining the use and abuse of bitcoin brain wallets. pages 609–618, 05 2017.

[44] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018.

[45] Xinyang Yi, Constantine Caramanis, and Eric Price. Binary embedding: Fundamental limits and fast algorithm. In *International Conference on Machine Learning*, pages 2162–2170. PMLR, 2015.

[46] Jinjie Zhang and Rayan Saab. Faster binary embeddings for preserving euclidean distances. In *International Conference on Learning Representations*, 2021.

Appendix

1. Goppa Codes

$\mathbf{y} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in \Gamma(\mathcal{L}, g)$ and $\|\mathbf{e}\|_0 \leq t$. To correct $\mathbf{y}$, we need to locate position of errors, say, $\mathcal{B} := \{i \in [n] : e_i \neq 0\}$, and determine the corresponding $\{e_i\}_{i\in\mathcal{B}}$. Define the *error-locator polynomial* $\sigma(z)$ and *error-evaluator polynomial* $w(z)$ as follows:

$$\sigma(z) \triangleq \prod_{i\in\mathcal{B}}(z-\alpha_i), \quad w(z) \triangleq \sum_{i\in\mathcal{B}} e_i \prod_{j\in\mathcal{B}\setminus\{i\}} (z-\alpha_j).$$

Finally, define the *syndrome* $S(\mathbf{y}) \triangleq \sum_{i\in[n]} \frac{y_i}{z-\alpha_i} \equiv \sum_{i\in\mathcal{B}} \frac{e_i}{z-\alpha_i} \pmod{g(z)}$.

Proposition A.1. *Suppose* $\mathbf{y} = \mathbf{c} + \mathbf{e}$ *with* $\mathbf{c} \in \Gamma(\mathcal{L}, g)$ *and* $r := \|\mathbf{e}\|_0 \leq t$. *Then,*

- *Degrees of* σ *and* w *are as follows:* $\deg\sigma = r$, $\deg w \leq r-1$.
- σ *and* w *are relatively prime:* $gcd(\sigma(z), w(z)) = 1$.
- *Error bits can be found as* $e_k = w(\alpha_k)/\sigma'(\alpha_k)$ *where* $k \in \mathcal{B}$ *and* σ' *is the formal derivative of* $\sigma(z)$.
- *One has* $\sigma(z)S(\mathbf{y}) = w(z)$ *modulo* $g(z)$.

The Proposition A.1 gives rise to the following correction procedure.

Goppa Code Decoding Procedure Γ.Dec

Input: $\mathbf{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_{q^m}$ with Goppa code $\Gamma(\mathcal{L} = \{\alpha_i\}_{i\in[n]}, g(z))$.

Algorithm:

1) Compute the syndrome $S(\mathbf{y}) = \sum_{i\in[n]} \frac{y_i}{z-\alpha_i}$.
2) Solve the equation $\sigma(z)S(\mathbf{y}) = w(z) \pmod{g(z)}$ with respect to coefficients of w and σ: namely, $\sigma(z) = z^r + \sum_{i\in[r]} \sigma_i z^i$ and $w(z) = \sum_{i\in[n]} w_i z^i$. If $q = 2$, take $w(z) := \sigma'(z)$.
3) Compute $\mathcal{B} = \{i \in [n] : \sigma(\alpha_i) = 0\}$ and compute errors $e_i = w(\alpha_i)/\sigma'(\alpha_i)$ for $i \in \mathcal{B}$.

Output: $\mathbf{c} = \mathbf{y} - \mathbf{e} \in \Gamma(\mathcal{L}, g)$.

Remark. While the aforementioned construction works over fields of arbitrary finite characteristic, we further always assume $q = 2$.

2. Geoposition factor

2.1. Converting to stable factor. One of the sources of key derivation could be geopositioning. It could be the current user's geoposition or any memorable place. However, geoposition is not precise, it possible to make it stable with the chosen precision, so geoposition will be treated in Unforgettable as a stable factor.

In order to convert geoposition into a stable value, we will use the S2Geometry [1]. This method projects the entire globe into a cube. Then, each face of the cube is divided into squares of the same size as $\frac{1}{2^{2\cdot d}}$ of the original face size, where d is the depth level that determines the cell size.

When the cube is divided into cells using the Hilbert curve, all the cells are numbered by a certain ID. The Hilbert curve is built recursively, starting from the $d = 0$, where we have only one square. Then, the starting square is divided into four smaller squares that are connected in a U-shape. Then each smaller square is divided into 4 additional squares, which are connected together in a U-shape or a rotated U-shape. All of the U-shapes are connected together, forming a single line. Then, such a process continues until we reach the required depth level. This operation is performed for each of the cube faces, and together they can construct a continuous, closed line.

When each cell has it's own ID, we can convert the geoposition into the cell with the needed precision. So now, the cell ID will be the unique identifier of the geolocation that the user has chosen.

In order to ensure that the user's geoposition is connected to only one cell, the geoposition should be taken with a precision of less than the minimum cell size. However, there is still a problem when the users geoposition could be chosen in the edge of two neighboring cells. To solve this, we propose recovering the key for the cells that are neighbors to the cell extracted from the geoposition cell. Usually, there are tools in S2Geometry libraries for extracting neighboring cells.

2.2. Security analysis. The most obvious way is to calculate the number of cells $6 \cdot 2^{2\cdot d}$, reverse it $\frac{1}{6\cdot 2^{2\cdot d}}$, and state that it is the probability of randomly guessing the cell among all the others. In this way, we already have $2 \cdot d + 2$ bits of security. However, it is less likely that person a will choose geolocation in the ocean or in areas where nobody lives.

In order to calculate such a probability, we need the area of the Earth without oceans $S_l \approx 150 \cdot 10^{12}\mathsf{m}^2$ and the actual area of the Earth $S \approx 510,1 \cdot 10^{12}\mathsf{m}^2$. Then, cell area $S_c \approx \frac{S}{6\cdot 2^{2\cdot d}}$. So, the number of cells that are on the land is $\frac{S_l}{S_c} \approx \frac{S_l \cdot 6 \cdot 2^{2\cdot d}}{S}$. So the probability is $\approx \frac{S}{S_l \cdot 6 \cdot 2^{2\cdot d}} \approx \frac{0.56}{2^{2\cdot d}}$, and the number of bits of security is $2 \cdot d$.

However, assuming a uniform distribution over the land S_l is still too optimistic. In reality, population density is highly uneven. Studies show that roughly 95% of the population inhabits only about 10% of the land area. Let us define this effective inhabited area as $S_{eff} \approx 0.1 \cdot S_l \approx 15 \cdot 10^{12}\mathsf{m}^2$.

Compared to the actual area of the Earth $S \approx 510 \cdot 10^{12}\mathsf{m}^2$, the ratio is $\frac{S}{S_{eff}} \approx 34$. Then, the number of effective cells is $\frac{S_{eff}}{S_c} \approx \frac{S_{eff} \cdot 6 \cdot 2^{2\cdot d}}{S}$. Consequently, the probability of guessing the cell is $\approx \frac{S}{S_{eff} \cdot 6 \cdot 2^{2\cdot d}} \approx \frac{34}{6\cdot 2^{2\cdot d}} \approx \frac{5.66}{2^{2\cdot d}}$.

So the entropy is $\approx -\log_2\left(\frac{5.66}{2^{2\cdot d}}\right) = 2 \cdot d - \log_2(5.66) \approx 2 \cdot d - 2.5$. Thus, the number of bits of security is $2 \cdot d - 3$.

3. Voice factor

One of the sources of key derivation could be voice biometrics. It could be the current user's voice recording or a specific passphrase. However, raw audio is not precise; it is susceptible to background noise, intonation variability, and channel distortion. To make it stable with the chosen precision, voice must be treated in Unforgettable as a stable factor.

In order to convert voice into a stable value, we will use the **ECAPA-TDNN** model. This method projects the variable-length audio signal into a fixed-dimensional embedding space $\mathbb{R}^{192}$. To produce the input for the model, the continuous audio signal is sampled at 16 kHz. Then, the signal is divided into short, overlapping frames to capture temporal stationarity. This operation is performed using a sliding window technique with a frame length of 25 ms and a stride of 10 ms. To minimize spectral leakage at the boundaries, a Hamming window is applied to each frame.

When the frames are prepared, a Short-Time Fourier Transform (STFT) is computed to obtain the power spectrum. This converts the signal from the time domain to the frequency domain. Then, the linear frequency spectrum is mapped to the **Mel scale**, which mimics the non-linear frequency resolution of the human ear. We apply a filterbank of 80 triangular filters spaced linearly on the Mel scale to aggregate the spectral energy. Finally, a logarithmic operation is applied to the filterbank energies to replicate the human perception of loudness. The resulting matrix serves as the input to the neural network.

When the features are extracted, the ECAPA-TDNN projects them into the embedding space. However, there is still a problem with transient noise in single utterances. To solve this, we propose recovering the centroid vector for the user by applying a windowed averaging technique. The input audio is processed in overlapping segments, and

the resulting vectors are aggregated. Finally, the vector $\mathbf{w}$ is L_2-normalized to lie on the surface of the unit hypersphere.

After receiving the feature vector from the neural network, we need to binarize it using the random projection method. Then binarized samplescould be compared to verify if the voice samples are the same using hamming distance, so we can put them directly into the fuzzy extractor.

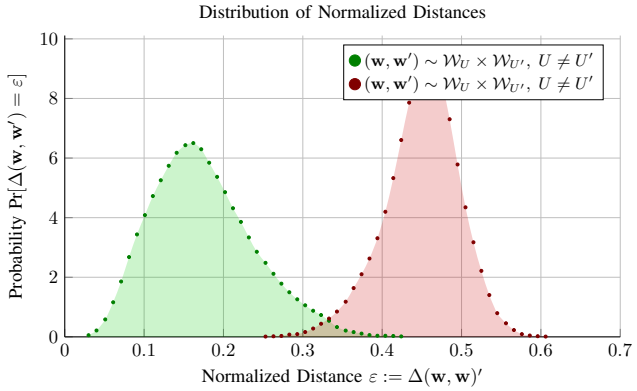

Figure 2: Distribution of distances: one distribution $\rho^+(\varepsilon)$ (marked in green) corresponds to probability of getting given distances between *same* people, while the other $\rho^-(\varepsilon)$ (marked in red) corresponds to distances between different people. As can be seen, proximity parameters (expected values of each distribution) are approximately $d^+ \approx 0.17$, and $d^- \approx 0.45$ respectively.