LifeXP$^+$: Secure, Usable and Reliable Key Recovery for Web3 Applications

Panagiotis Chatzigiannis, Suvradip Chakraborty, and Shimaa Ahmed

Visa Research
{pchatzig,suvchakr,shiahmed}@visa.com

Abstract. In the Web2 world, users control their accounts using credentials such as usernames and passwords, which can be reset or recovered by centralized servers if the user loses them. In the decentralized Web3 world however, users control their accounts through cryptographic private-public key pairs which are much more complex to manage securely. In addition, the decentralized nature of Web3 makes account recovery impossible in the absence of predetermined recovery mechanisms. With the proliferation of blockchains and cryptocurrencies over the last years, it is crucial to provide users secure, usable and reliable ways to recover their accounts and assets. However, up to this day, no Web3 recovery method has adequately achieved all three of the above required properties. For instance, conventional "mnemonic" backups which can deterministically reconstruct a private key require verbatim recall of a fixed word list, creating an unpleasant usability/security trade-off.

In this work, we present a fully-offline protocol called LifeXP$^+$, that allows a user to reconstruct a cryptographically-secure private key from a natural-language story, which a user always remembers, such an memorable life event. To ensure usability of our protocol, key reconstruction can work even when the story is later retold with different wording or grammar, only requiring to preserve the semantics. The protocol combines pre-trained sentence embeddings to capture semantics, locality-sensitive hashing to quantize embeddings into stable bit strings, a cryptographic fuzzy extractor that corrects bit errors caused by paraphrasing, and a biometric factor that is fused with the linguistic factor to boost entropy and enhance security. In our paper we describe the design, show that the protocol achieves the required properties, and provide an evaluation based on publicly-available datasets which runs completely offline on commodity hardware, showcasing its feasibility.

1 Introduction

1.1 Problem Statement

Public-key cryptocurrencies are based on cryptographic protocols to authenticate user transactions. While the cryptographic strength of the private key ensures that no malicious entity can perform unauthorized transactions in a brute-force manner, the burden of secure key custody is placed on the end-user. The user

needs to make sure that the private key is kept secret, as a key leakage to an adversary would result in theft of funds, with no authority being able to reverse it due to the decentralized nature of the blockchain. Therefore, it is generally recommended that the user minimizes the attack surface by avoiding storing replicas of the key in different mediums or locations [8, 23]. At the same time however, if access to the private key is lost for any reason (such as hardware failure, user locked out of the device or its software, device loss etc.), the blockchain account and its associated funds are irrecoverable. No authority can help with recovery here either, again due to the decentralized nature of the blockchain. This has led to many users permanently losing access to their funds, with some prominent cases involving millions of dollars [2].

The recovery problem has been acknowledged by the cryptocurrency world, with popular wallets attempting to mitigate this risk using various recovery methods, such as predetermined entities with the power to help with recovery [1, 19] ("guardians"), secret-sharing or encrypting keys across different entities [1, 3], or sophisticated smart contract-based recovery protocols [4], aiming to strike a balance between usability, security and reliability (or availability). Still, one of the most commonly-used methods remains the recovery phrase, a 12-word random seed from a 2^{11} dictionary size, generated during wallet creation which can deterministically reconstruct the key if needed (in Bitcoin, this is referenced as the BIP-39 standard [16]). While this is intended to provide an easy and more usable way to represent a key in human-readable format, the seed phrase must still be stored verbatim; forgetting/mistyping a single word or not preserving their exact ordering produces a completely different seed due to the avalanche effect of cryptographic hash functions. Essentially, this method has not offered much, as humans tend to remember *stories*, not random words with no meaning. Also, workarounds such as writing these words on a piece of paper or storing them in different mediums again increase the attack surface and the risk of account compromise. Other workarounds such as adding numbers to the words which would reduce the total number of words needed still requires a high amount of mental workload which is not yet acceptable to everyday, non tech-savvy users [6].

To address the problem we propose LifeXP$^+$, a new protocol which can deterministically reconstruct cryptographic keys out of natural language stories chosen by the user. While the story needs to have greater length than just 12 words to ensure sufficient entropy (and therefore preserve security), LifeXP$^+$ is much more user-friendly; it is designed to tolerate linguistic variation while mapping similar inputs to the *same* key, still guaranteeing high min-entropy (with the aid of biometrics) so that an attacker cannot brute-force the key space by enumerating stories. The only requirement is that the user picks the story according to some basic guidelines; besides its length and uniqueness (trivial stories easily guessable need to be avoided), the user needs to pick a story that not only is embedded in his/her memory (e.g. a childhood story or some other memorable experience from his/her life), but is also uniquely known to the user. To model security and adversarial attempts to guess the story and therefore reconstruct the key, we consider two types of adversaries: a) A *weak*

adversary which has no relation to the user, resembling random attackers in the wild attempting to recover blockchain keys b) A *strong* adversary, resembling attackers related to the user (such as close friends or relatives) who have better knowledge of the user's experiences and therefore have an advantage in guessing the story.

1.2 Description

At a high level, LifeXP$^+$ aims to deterministically map any semantically equivalent telling of a user's personal story to the same cryptographic key, while rejecting dissimilar inputs and enforcing cryptographic entropy. The core idea is to treat a natural–language narrative as a noisy secret: small linguistic variations are expected and must be corrected, but the underlying "meaning" should anchor to a stable point in a discrete key space. The pipeline comprises of 4 stages that together bridge continuous semantics to uniform keys and provide error tolerance.

1. Semantic embedding (meaning capture). A local sentence/paragraph encoder converts the input story into a fixed–dimensional vector in a semantic space where paraphrases are close and unrelated texts are far apart. This step suppresses irrelevant lexical and syntactic variation (e.g., word order, morphology, synonyms) and preserves the gist of the narrative.
2. Device–salted quantization (continuous→discrete). We project the embedding onto a device–seeded family of random hyperplanes and record only the signs, yielding a compact binary "semantic fingerprint". This locality–sensitive quantization preserves "neighbourhoods" (nearby embeddings map to bit strings with small Hamming distance), while the secret salt thwarts precomputation and ties the discretization to the user's device without external trust.
3. Biometric fusion. To increase effective min–entropy and provide two–factor protection, the protocol also receives the user's biometrics as input (e.g., a fingerprint). The biometric is independently processed through the same pipeline to yield a stable biometric key; we then combine the linguistic and biometric keys to form the final secret. This forces an attacker to compromise both a private narrative and a physical trait.
4. Error tolerance and uniformity via fuzzy extraction. Because even close paraphrases can flip some bits, we enroll the binary fingerprint with a fuzzy extractor instantiated by an error–correcting code (ECC). Enrollment produces public helper data and a uniformly random key; recovery uses a new story rendition to reconstruct the exact same key as long as the new fingerprint is within the correction radius. By applying a randomness extractor after error correction, the final key is information–theoretically close to uniform.

Operationally, the system supports two modes: enrollment and recovery. During enrollment, the client computes a semantic embedding of the autobiographical story and a biometric embedding. These vectors are normalized and quantized into an n-bit pre-extraction code using a locality-sensitive hashing scheme. An ECC encodes this binary string to produce public helper data, and a key is

derived from the corrected code. During recovery, the user provides a paraphrase and biometrics; the client repeats the embedding, fusion, and quantization, uses the stored helper data to correct bit errors, and reproduces the exact key. All steps are executed locally within less than a second on commodity CPUs. This architecture simultaneously provides usability (free-form recall), determinism (exact key reproduction from paraphrases), security, and deployability, with no external server dependency.

1.3 Related Work

Key recovery in blockchain settings. The key recovery problem in blockchains was recognized from the early days of bitcoin, which led to the implementation of BIP-39 [16]. Although this remains the most widely deployed recovery approach by the vast majority of wallets, it still suffers from usability and memorability limitations, and it is not considered to have helped into the mainstream adoption of blockchain/web3 wallets. Several systems in the blockchain space have explored alternative mechanisms for private key recovery. For instance, Coinbase MPC wallet [13] employs a threshold multi-party computation protocol to split signing authority across devices and servers, enabling recovery without direct exposure of the private key. Similarly, some commercial wallets [1] implement a social recovery model based on smart contracts, where a user's "guardians" can collectively approve key rotation. KELP [4] is a more advanced a complex key recovery protocol using smart contracts, without needing to rely on external guardians, but rather requiring collateral deposits locking them for extensive periods of time. Recent works have examined the composability of different recovery methods [9], as well as providing systematic overviews of key management and recovery methods [8]. Finally, user studies have investigated the real-world usability, security trade-offs, and failure modes of recovery mechanisms in blockchain wallets [15], highlighting the gap between theoretical guarantees and user practices.

Semantic extraction. Near–duplicate document detection uses locality–sensitive hashing (LSH) such as SimHash [7]. LSH preserves Hamming distance for lexically similar documents but fails for paraphrases. *Semantic hashing* with deep autoencoders [18] maps documents to binary codes that preserve meaning, yet practical deployments do not target cryptographic entropy. Fuzzy extractors [11] convert noisy biometric measurements into uniform keys; the biometric literature shows many deployments (e.g. fingerprints, voice, EEG) but hardly any with natural language. A recent work by Wu *et al.* [22] combines semantic extraction and fuzzy extraction to derive keys from sentences. However, their work is based on non-public datasets, covers only short phrases, does not integrate biometrics and doesn't have any formal security analysis or an end-to-end offline implementation. Finally, Life Experience Passwords (LEPs) [21] propose using personal life experiences as high-entropy passwords, where users answer a series of autobiographical questions during registration and must provide the same answers during authentication. Their focus is on replacing traditional passwords with memorable, user-specific knowledge that is difficult for attackers to guess. However, LEPs are primarily designed for *direct authentication*, rely solely on

textual autobiographical data and assume consistent recall of exact answers, whereas our scheme targets *cryptographic key recovery and binding* for blockchain wallets, fusing *multiple modalities* (personal narratives and biometrics) to increase entropy and security and evaluate against *strong* and *weak* adversary scenarios, and incorporates fuzzy extraction and error correction to handle natural variability in retelling memories. These differences make our approach more suitable for secure, repeatable key recovery in high-stakes cryptographic applications where both usability and adversary resistance are critical.

2 Preliminaries

Sentence Embedding. A sentence embedding is a mapping from a variable-length sequence of words (or tokens) into a fixed-dimensional vector space, such that semantically similar sequences are located close to each other under a chosen similarity metric (e.g., cosine similarity). Formally, for a sentence $S = (w_1, w_2, \ldots, w_n)$, the embedding function is $f : S \to \mathbb{R}^d$ where d is the embedding dimension and f is typically parameterized by a neural encoder (e.g., transformer-based). When applied to narratives or stories, f encodes not only lexical meaning but also contextual and relational information across the narrative, enabling downstream tasks such as semantic similarity measurement, clustering, or retrieval.

Locality–Sensitive Hashing. LSH is a family of probabilistic hashing methods designed to increase the likelihood that similar items in a high-dimensional space are mapped to the same hash "bucket", while dissimilar items are mapped to different buckets. Formally, given a metric space $(\mathcal{M}, D)$ with distance function $D : \mathcal{M} \times \mathcal{M} \to \mathbb{R}\geq 0$, an LSH family $\mathcal{H}$ is defined such that for $h \in \mathcal{H}$ and $p, q \in \mathcal{M} : P_{h\in\mathcal{H}}(h(p) = h(q))$ is high if $D(p, q) \leq r_1$ and is low if $D(p, q) \geq r_2$ where $r_1 < r_2$ and (r_1, r_2) are distance thresholds.

Thus, LSH maps embeddings of semantically similar stories, even when expressed with different wording, to the same hash bucket with high probability. This consistent mapping allows the system to group paraphrased narratives as equivalent, while still separating them from semantically unrelated stories.

Fuzzy Extractors. A fuzzy extractor derives a stable, uniformly random key from noisy but similar inputs. It consists of two procedures: $\mathsf{Gen}(c^\star) \to (R, P)$, which on enrollment outputs a key R and public helper data P from an input $c^\star$, and $\mathsf{Rec}(c', P) \to R$, which reproduces R for any c' within Hamming distance t of $c^\star$, i.e. $\mathrm{dist}(c', c^\star) \leq t$.

Biometric Hash. A biometric hash is a fixed-length binary string derived from a biometric sample such that samples from the same user map to the same value with high probability, while samples from different users rarely collide. Formally, given a biometric sample B, a feature extractor Φ maps B to a binary template $T \in \{0, 1\}^n$, which is then transformed via $H : \{0, 1\}^n \to \{0, 1\}^k$.

3 Our LifeXP$^+$ Protocol Description

This section formalizes the end–to–end protocol, specifies interfaces and parameters, and presents enrollment and recovery algorithms. In our construction, the biometric factor is fused at the feature level immediately after story embedding.

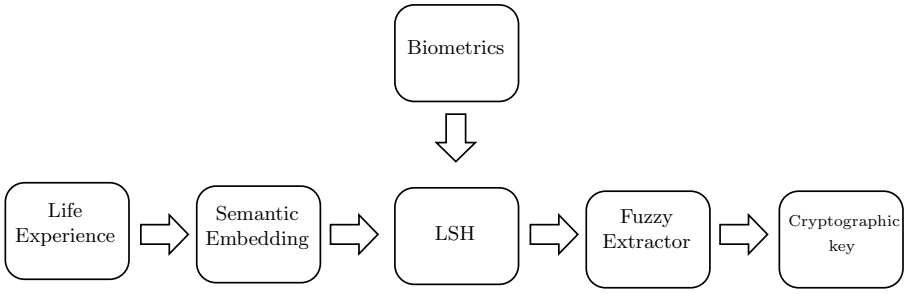

Fig. 1. LifeXP$^+$ system flow

3.1 Interfaces and Notation

We denote by $\mathcal{E}_s : \mathcal{S} \to \mathbb{R}^{d_s}$ a local semantic encoder for sentences or paragraphs, and by $\mathcal{E}_b : \mathcal{B} \to \mathbb{R}^{d_b}$ a local biometric feature extractor. A locality-sensitive hashing map $\mathsf{LSH} : \mathbb{R}^d \times \mathcal{R}_{\mathsf{LSH}} \to \{0,1\}^m$ is parameterized by randomness $r \in \mathcal{R}_{\mathsf{LSH}}$ and later instantiated via PRF-based random hyperplanes. We write $(\mathsf{Gen}, \mathsf{Rep})$ for a generic fuzzy extractor, where $\mathsf{Gen} : \mathcal{W} \times \mathcal{R}_{\mathsf{FE}} \to \{0,1\}^{\kappa(\lambda)} \times \mathcal{P}$ outputs a key and public helper data from input W, and $\mathsf{Rep} : \mathcal{W} \times \mathcal{P} \to \{0,1\}^{\kappa(\lambda)}$ reconstructs the key; this will be instantiated with a code-based construction. We use KDF for a cryptographic key-derivation function, $\|$ for concatenation, and Hdist for Hamming distance.

User Input and Output. In the enrollment phase, the input consists of the user's personal story $s \in \mathcal{S}$ and a biometric sample $b \in \mathcal{B}$. In the recovery phase, the input is a paraphrased story s', a biometric sample b', and public metadata M produced during enrollment; M can be stored openly and is used to assist in recovery. In both modes, the output is a uniformly random key $R \in \{0,1\}^\kappa$, where $\kappa \in \{128, 256\}$.

Setup and Parameters. The text embedding dimension is $d_t = 768$, and the biometric embedding dimension $d_b \in \{192, 256, 512\}$ depends on the implementation; the fused vector has dimension $d = d_t + d_b$. Locality-sensitive hashing produces $m = 256$ sign bits (configurable). The fuzzy extractor is instantiated with a Reed–Solomon error-correcting code chosen to match the empirically

observed bit-flip budget, enabling reproducible key generation from noisy inputs. The security parameter is $\lambda = 256$ bits, and the extracted key length is $\kappa \in \{128, 256\}$.

3.2 LifeXP$^+$ Protocol flow

Before presenting the protocol in detail, we provide an overview of the key derivation pipeline. The user supplies a personal story and biometric data, which are processed through feature extraction, LSH, clustering, and a fuzzy extractor to produce a reproducible key. For a visual summary, see Figure 1, which illustrates the complete flow from input to final cryptographic key, and refer to Algorithm 1 for the detailed step-by-step procedure.

Algorithm 1 User-Derived Cryptographic Key Generation

Require: Personal story s, biometric input b, LSH seed $r_{\mathrm{LSH}} \in \mathcal{R}_{\mathsf{LSH}}$, seed $\rho \in \{0,1\}^{256}$ of the fuzzy extractor.

Ensure: Public metadata M, derived key $K \in \{0,1\}^{\ell(\lambda)}$

1: $v_s \leftarrow \mathcal{E}_s(s)$, $v_b \leftarrow \mathcal{E}_b(b)$

2: $v \leftarrow (v_s/\|v_s\|) \parallel (v_b/\|v_b\|)$ ▷ Concatenate normalized features

3: $u \leftarrow \mathsf{LSH}(v; r_{\mathrm{LSH}})$ ▷ Compact binary sketch of v; r_{LSH} is the PRF seed controlling hyperplane generation (see Algorithm 3); fixing it ensures reproducibility, varying it enables per-enrollment unlinkability

4: $W \leftarrow Q(C(u))$ ▷ Clustering and quantization of LSH sketch

5: $(R, P) \leftarrow \mathsf{Gen}(W; \rho)$ ▷ Fuzzy extractor generates helper data P and outputs a reproducible key statistically close to uniform

6: $K \leftarrow \mathsf{KDF}(R)$ ▷ Final cryptographic key for blockchain wallets

7: $M \leftarrow (P, \text{LSH parameters}, \text{cluster info}, \dots)$

8: **return** (M, K)

Algorithm 2 User-Derived Key Recovery

Require: Noisy story s', biometric b', public metadata $M = (P, \dots)$, LSH seed r_{LSH}, seed $\rho \in \{0,1\}^{256}$ of the fuzzy extractor.

Ensure: Reconstructed key $K' \in \{0,1\}^{\ell(\lambda)}$

1: $v'_s \leftarrow \mathsf{SentenceEncoder}(S')$, $v'_b \leftarrow \mathsf{BiometricEncoder}(B')$

2: $v' \leftarrow (v'_s/\|v'_s\|) \parallel (v'_b/\|v'_b\|)$ ▷ Concatenate normalized features

3: $u' \leftarrow \mathsf{LSH}(v'; r_{\text{LSH}})$ ▷ Compact binary sketch of v'; r_{LSH} ensures the same hyperplanes as in enrollment

4: $W' \leftarrow Q(C(u'))$ ▷ Clustering and quantization of LSH sketch

5: $R' \leftarrow \mathsf{Rep}(W', P)$ ▷ Re-derive fuzzy extractor output using helper data P from metadata M

6: $K' \leftarrow \mathsf{KDF}(R')$ ▷ Final cryptographic key reconstruction

7: **return** K'

— **Feature Fusion and Quantization.** Let $\mathcal{E}_s : \mathcal{S} \to \mathbb{R}^{d_s}$ and $\mathcal{E}_b : \mathcal{B} \to \mathbb{R}^{d_b}$ be deterministic embedding functions or encoders as defined above. Compute the embeddings:

$$v_s = \mathcal{E}_s(s), \qquad v_b = \mathcal{E}_b(b).$$

Normalization. Normalize both embeddings to unit vectors:

$$\hat{v}_s = \frac{v_s}{\|v_s\|} \in \mathbb{S}^{d_s-1}, \qquad \hat{v}_b = \frac{v_b}{\|v_b\|} \in \mathbb{S}^{d_b-1}.$$

Concatenate to form the combined feature vector $v = (\hat{v}_s, \hat{v}_b) \in \mathbb{R}^d$,
where $d = d_s + d_b$.

— **Locality-Sensitive Hashing (PRF-based random hyperplanes).** Let $\mathsf{LSH} : \mathbb{R}^d \to \{0,1\}^m$ be a random hyperplane LSH map, where the hyperplanes are derived deterministically from a PRF. We compute the LSH output as follows:

Let $v \in \mathbb{R}^d$ be the normalized feature vector ($\|v\| = 1$) from the above step.

We construct a random hyperplane LSH map $\mathsf{LSH} : \mathbb{R}^d \to \{0,1\}^m$ as shown in Algorithm 3.

Remarks. The choice of PRF output width b is flexible: 32 bits per component suffice for practical floating-point precision, while larger widths improve numerical reproducibility. Using the PRF allows the hyperplanes to be deterministically reconstructed from the seed via the fuzzy extractor. Normalization ensures that the sign test is equivalent to computing cosine similarity between vectors. Finally, small variations in v may flip some bits, but subsequent clustering C and quantization Q mitigate these errors.

Algorithm 3 PRF-based Random Hyperplane LSH

Require: Normalized input $v \in \mathbb{R}^d$ ($\|v\| = 1$), number of hyperplanes m, PRF $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^b$, seed $\mathsf{seed} \in \{0,1\}^\lambda$
Ensure: Binary LSH sketch $u \in \{0,1\}^m$

for $j = 1$ to m **do**
  for $i = 1$ to d **do**
    $r_{j,i} \leftarrow \mathsf{PRF}(\mathsf{seed}, (j,i))$ ▷ Pseudorandom b-bit output
    Map first 32 bits of $r_{j,i}$ to $x \in [-1,1]$ via $x = 2 \cdot u/2^{32} - 1$, where u is the unsigned integer
  end for
  $r_j \leftarrow (r_{j,1}, \ldots, r_{j,d})$
  $\hat{r}_j \leftarrow r_j / \|r_j\|$ ▷ Normalize hyperplane
  $u_j \leftarrow \mathbf{1}\{\langle v, \hat{r}_j \rangle \geq 0\}$ ▷ LSH bit
end for
$u \leftarrow (u_1, \ldots, u_m)$
return u

— **Feature stabilization via clustering and quantization.** After applying the locality-sensitive hash to get $u = \mathsf{LSH}(v) \in \{0,1\}^m$, we do the following:.

Clustering: Define a clustering function

$$C : \{0,1\}^m \to \mathcal{C} \subseteq \{0,1\}^m,$$

where $\mathcal{C}$ is a set of representative cluster centers (e.g., the codewords of an error-correcting code). For any $u \in \{0,1\}^m$, $C(u)$ is defined as the closest representative in $\mathcal{C}$ under Hamming distance. This step canonicalizes noisy LSH outputs to stable representatives.

Quantization: Define a quantizer

$$Q : \mathcal{C} \to \{0,1\}^n,$$

which maps the cluster representative to a shorter, robust binary string. The quantizer may, for example, (i) discard unreliable bit positions, or (ii) map projections into multi-bit bins to increase stability.

Robust feature string: The stabilized string given to the fuzzy extractor is

$$w = Q(C(u)) \in \{0,1\}^n.$$

— **Key derivation.** Finally, the fuzzy extractor and key derivation function are applied:

$$(R, P) \leftarrow \mathsf{Gen}(w), \quad K = \mathsf{KDF}(R).$$

where P is public helper data and $R \in \{0,1\}^\kappa$ is the output of the fuzzy extractor that will be post-processed via a standard KDF to derive the final secret key to be used in the wallet:

$$K_{\mathrm{BIP32}} = \mathsf{HKDF\_SHA256}(R,\ \text{info:“bip32-seed”}).$$

The same interface supports other consumers (TLS keys, disk encryption), preserving offline operation.

The public metadata M consists of all public information (e.g., P, and optionally u or published cluster ids etc).

Correctness of our Protocol. Our protocol is correct in the sense that, for honest users providing inputs within the tolerated noise bounds, the reconstructed key matches the original key with high probability. A formal proof of correctness follows.

$$\begin{aligned} v &= \mathsf{Feature}(S, B), & u &= \mathsf{LSH}(v; r_{\mathsf{LSH}}), \\ u_C &= C(u), & w &= Q(u_C), \end{aligned}$$

and the recovery pipeline outputs

$$\begin{aligned} v' &= \mathsf{Feature}(S', B'), & u' &= \mathsf{LSH}(v'; r_{\mathsf{LSH}}), \\ u'_C &= C(u'), & w' &= Q(u'_C). \end{aligned}$$

Here $\mathsf{Feature}$ is the composition of the sentence and biometric encoders followed by concatenation of the normalized feature vectors.

At enrolment, we compute

$$\begin{aligned} (R, P) &\leftarrow \mathsf{Gen}(w; \rho), \\ K &= \mathsf{KDF}(R), \end{aligned}$$

and at recovery we compute

$$\begin{aligned} R' &= \mathsf{Rep}(w', P), \\ K' &= \mathsf{KDF}(R'), \end{aligned}$$

where K' is the reconstructed key when Rep succeeds.

We assume the following per-step correctness/stability bounds (parameters are functions of the security parameter λ, omitted for brevity):

- **Feature stability:** the encoder produces a stable representation with probability
$$\Pr\left[\mathrm{dist}_v(v, v') \le \Delta_v\right] \ge 1 - \delta_{\mathrm{feat}},$$
where $\mathrm{dist}_v(\cdot, \cdot)$ is an appropriate distance on feature vectors (e.g., angle or Euclidean) and Δ_v is a small tolerance.
- **LSH stability:** conditioned on $\mathrm{dist}_v(v, v') \le \Delta_v$,
$$\Pr\left[\mathsf{Hdist}(u, u') \le h \mid \mathrm{dist}_v(v, v') \le \Delta_v\right] \ge 1 - \delta_{\mathsf{LSH}},$$
where $\mathsf{Hdist}(\cdot, \cdot)$ is Hamming distance and h is a chosen Hamming-tolerance. Random-hyperplane LSH preserves neighbourhoods in expectation, so the induced bit flip count $X = \mathsf{Hdist}(u, u')$ concentrates around $k \cdot \theta/\pi$, where θ is the angle between u and u'. For paraphrases from the same user and stable biometric captures, θ is small and $X \le h$ with high probability.

– **Clustering/quantization stability:** if $\mathsf{Hdist}(u, u') \leq h$ then with probability at least $1 - \delta_C$ we have

$$Q(C(u)) = Q(C(u')).$$

Equivalently, clustering and quantization map u and u' to the same discrete string w except with probability δ_C. In other words:

$$\mathsf{Hdist}(w, w') \leq h + \Delta_C.$$

– **Fuzzy-extractor correctness:** the fuzzy extractor recovers the same corrected value provided the input strings are within allowed distance τ. Let $\tau = h + \Delta_C$. Then:

$$\Pr\left[\mathsf{Rep}(w', P) = R \mid \mathsf{Hdist}(w, w') \leq \tau\right] \geq 1 - \delta_{\mathsf{FE}},$$

– **Extractor/KDF determinism:** KDF is deterministic; if Rep outputs the same R as in enrollment then $K' = K$ holds deterministically.

Hence, by the union bound, the probability that at least one of the above failure events occurs is at most

$$\delta_{\text{feat}} + \delta_{\mathsf{LSH}} + \delta_C + \delta_{\mathsf{FE}} = \delta_{\text{tot}},$$

so $\Pr[K' = K] \geq 1 - \delta_{\text{tot}}$, proving the correctness of the protocol.

4 Security Analysis of our LifeXP$^+$ Protocol.

In this section, we formalize the security properties of our protocol under standard cryptographic notions. We begin by specifying the overall security goal and adversarial threat model before proceeding to precise definitions and proofs.

4.1 Security Goal and Threat Model

In order to reason about the secrecy of cryptographic keys derived from user inputs, we must formally specify both the source of randomness (the user's life experiences and biometric data) and the adversarial capabilities. Our framework models the user input as a high-entropy distribution conditioned on potential side information available to an adversary. The goal of the system is to extract nearly uniform keys from this distribution, despite the leakage induced by helper data and other public metadata.

We begin by defining the user distribution and entropy assumptions, followed by the adversary classes (weak vs. strong), and finally the indistinguishability-based key secrecy definition.

User Distribution Assumption. We model the user's input as a random variable $(S, B, Z) \sim \mathcal{D}$, where

– S is the user's personal story or natural language input,
– B is the user's biometric feature vector, and
– Z is adversarial side information correlated with (S, B) (empty for weak adversaries).

Entropy requirement. We assume that the distribution $\mathcal{D}$ satisfies a prior conditional min-entropy requirement:

$$\tilde{H}_\infty(S, B \mid Z) := -\log\Big(\max_{s,b,z} \Pr[(S, B) = (s, b) \mid Z = z]\Big) \geq k_0.$$

This ensures that, for every possible adversarial side information z, the most likely value of (S, B) has probability at most 2^{-k_0}, guaranteeing sufficient unpredictability in the user's input before any public metadata is published.

Adversary classes. We consider passive, polynomial-time adversaries who observe public metadata M and may hold side information Z.

– *Weak adversary* $\mathcal{A}_{\mathsf{weak}}$: $Z = \varnothing$ (or only population priors).
– *Strong adversary* $\mathcal{A}_{\mathsf{strong}}$: holds auxiliary side information Z that is jointly distributed with (S, B) (e.g., social posts, friend knowledge).

In the *weak-adversary case*, Z is independent of (S, B) (or empty), so the requirement reduces to $H_\infty(S, B) \geq k$, i.e., the user's story and biometrics together contain at least k bits of unpredictability with respect to general population priors. In the *strong-adversary case*, Z may capture partial information about the user's life or biometric (e.g., information known to friends/relatives). The requirement $\tilde{H}_\infty(S, B \mid Z) \geq k$ enforces that even given this side information, there remains at least k bits of unpredictability in the joint input.

Key secrecy with helper-data privacy and leakage accounting.

Setup and notation. Let $(X, Z) \sim \mathcal{D}$ be jointly distributed random variables, where

– X is the user's secret source (the full secret datum to be used to derive keys; e.g. a combined representation of life-experience and biometric), and
– Z is adversarial side information correlated with X (empty for the weak adversary model).

All randomness used by algorithms below is independent of (X, Z) unless stated.

Let Enroll be an arbitrary (possibly randomized) enrollment algorithm that takes X and produces a tuple of public metadata and secret material:

$$(M, R) \leftarrow \mathsf{Enroll}(X),$$

where M denotes all *public* metadata published by the enrollment procedure (helper data, public seeds, buckets, published sketches, salts, etc.), and R denotes

internal randomness or a raw secret string (kept private by the challenger; a KDF may be applied to R to yield the final key).

Entropy thresholds and leakage accounting. Fix two entropy thresholds k_0 and k_1 with $k_1 \leq k_0$:

- k_0 is the *prior* entropy requirement on the source: a design assumption about (X, Z) before enrolment.
- k_1 is the required *residual* min-entropy of X after publishing M; it encodes a leakage budget and rules out trivial attacks.

More formally, let k_0 be the *prior* conditional-min-entropy guarantee required from the user source such that:

$$\tilde{H}_\infty(X \mid Z) \;\geq\; k_0.$$

Let $\mathrm{leak}(M)$ denote a conservative bound on the number of bits of information about X revealed by the public metadata M (this includes helper-data leakage due to P, any published LSH buckets, seeds, salts, etc.). We then define a *post-publication* entropy threshold

$$k_1 \;:=\; k_0 - \mathrm{leak}(M).$$

Intuitively, k_1 is the minimum residual entropy we require *after* publication of M to rule out trivial attacks.

Since public metadata M may leak information about the source X, not all executions retain enough entropy to ensure secrecy. We therefore restrict attention to *non-trivial* executions where the residual entropy remains above a threshold.

Good execution condition (non-triviality). Define the good (non-trivial) event as:

$$\mathsf{Good} \;:=\; \big\{\tilde{H}_\infty(X \mid M, Z) \geq k_1\big\}.$$

If $\neg\mathsf{Good}$ then the helper-data leakage is deemed too large: an adversary that observes (M, Z) can guess X (and hence K) with probability significantly larger than 2^{-k_1}; we call such a situation a *trivial attack* (the system is obviously insecure in this case).

Security statements below are therefore made *conditioned* on the event Good, i.e., we restrict attention to non-trivial (meaningful) executions where helper-data privacy is not catastrophically violated.

We next define the indistinguishability experiment that models an adversary attempting to distinguish the real key from random.

Key secrecy (indistinguishability) experiment. Fix a security parameter λ and let the target key length be $\kappa = \kappa(\lambda)$ bits and a distinguishing bound $\varepsilon = \varepsilon(\lambda) > 0$. Let $\mathcal{A}$ be a PPT adversary. The experiment $\mathsf{Exp}^{\mathsf{ind}}_{\mathsf{Enroll},\mathcal{A}}(k_0, k_1, \kappa)$ proceeds:

- *Source sampling.* Sample $(X, Z) \sim \mathcal{D}$.

– *Enrollment.* Run $(M, R) \leftarrow \mathsf{Enroll}(X)$ and set $K = \mathsf{KDF}(R) \in \{0,1\}^\kappa$.
– *Goodness check.* Evaluate (or upper-bound) $\tilde{H}_\infty(X \mid M, Z)$. If

$$\tilde{H}_\infty(X \mid M, Z) < k_1,$$

then the challenger *aborts* and returns the special symbol $\perp_{\text{trivial}}$ (the execution is labeled trivial/insecure).
– *Challenge.* Otherwise (i.e., Good holds), give (M, Z) to $\mathcal{A}$. Sample a uniform bit $b \in \{0,1\}$. If $b = 1$ give $\mathcal{A}$ the real key K; if $b = 0$ give $\mathcal{A}$ a uniformly random string $U_\kappa \in \{0,1\}^\kappa$.
– *Output.* $\mathcal{A}$ outputs a bit b'. The experiment outputs 1 if $b' = b$ and 0 otherwise.

Define the adversary's distinguishing advantage conditioned on non-triviality:

$$\mathrm{Adv}_{\mathcal{A}} \;=\; \Big|\Pr\left[\mathsf{Exp}^{\mathsf{ind}}_{\mathsf{Enroll},\mathcal{A}}(k_0, k_1, \kappa) = 1 \,\Big|\, \mathsf{Good}\right] - \tfrac{1}{2}\Big|.$$

We now formalize the security guarantees of an enrollment scheme. The definition captures both prior entropy assumptions and protection against trivial attacks caused by helper-data leakage.

Security definition. The enrollment scheme $(\mathsf{Enroll}, \mathsf{KDF})$ is $(\kappa, \varepsilon; k_0, k_1)$-*secure* against an adversary class $\mathfrak{A}$ if the following hold:

1. (*Prior entropy.*) The source satisfies the prior entropy requirement:

$$\tilde{H}_\infty(X \mid Z) \;\geq\; k_0.$$

2. (*Non-trivial indistinguishability.*) For every adversary $\mathcal{A} \in \mathfrak{A}$, conditioned on Good the distinguishing advantage is bounded:

$$\mathrm{Adv}_{\mathcal{A}}(\lambda) \;\leq\; \varepsilon(\lambda).$$

If an execution yields $\neg\mathsf{Good}$ the challenger reports the execution as $\perp_{\text{trivial}}$ and the scheme makes no secrecy claim for that run.

Remarks. In practice $\tilde{H}_\infty(W \mid M, Z)$ will be estimated empirically (or conservatively bounded analytically). The leakage budget $\mathrm{leak}(M)$ should include all published values (helper data P, LSH seeds, published buckets, etc.). If desired, one can replace the abort with a weaker notion: report the adversary's advantage *and* a leakage metric; however for clean cryptographic guarantees it is preferable to condition security on the non-trivial event.

Unlinkability. This ensures that an adversary cannot determine whether two key-generation or enrolment events correspond to the same user, even if they observe all public metadata. In particular, given helper data (P_1, M_1) and (P_2, M_2) from two enrollments, unlinkability requires that no polynomial-time adversary can distinguish whether the two tuples were produced by the same user or two

independent users with advantage greater than $\varepsilon_{\text{link}}$. Formally, we model the adversary as attempting to distinguish whether two enrollments come from the same underlying source X or from independent sources drawn from the same distribution $\mathcal{D}$. We exclude full device compromise at enrollment and recovery time (e.g., malware exfiltrating raw factors), physical sensor hijacking, and side-channel attacks on hardware beyond standard software countermeasures. These can be mitigated with hardened runtimes and secure UI but are orthogonal to protocol design. We now formally define our unlinkability game as follows:
Setup. Let $(X, Z) \sim \mathcal{D}$ be the user source and adversarial side information, as in the key-secrecy experiment. Let Enroll be a generic enrollment algorithm producing public metadata M and secret material R, and KDF derive the final key:

$$(M, R) \leftarrow \mathsf{Enroll}(X), \quad K = \mathsf{KDF}(R).$$

Fix a security parameter λ, target key length $\kappa(\lambda)$, and a distinguishing advantage bound $\varepsilon_{\text{link}}(\lambda)$.

Unlinkability experiment. The unlinkability experiment $\mathsf{Exp}^{\mathsf{unlink}}_{\mathsf{Enroll},\mathcal{A}}(\lambda)$ proceeds as follows:

1. Challenger samples a bit $b \in \{0, 1\}$ uniformly at random.
2. If $b = 1$, sample a single user source $X \sim \mathcal{D}$ and generate two enrollments:
$$(M_1, R_1) \leftarrow \mathsf{Enroll}(X), \quad (M_2, R_2) \leftarrow \mathsf{Enroll}(X).$$
3. If $b = 0$, sample two independent user sources $X_1, X_2 \sim \mathcal{D}$ and generate two enrollments:
$$(M_1, R_1) \leftarrow \mathsf{Enroll}(X_1), \quad (M_2, R_2) \leftarrow \mathsf{Enroll}(X_2).$$
4. Give the adversary $\mathcal{A}$ the public metadata (M_1, M_2) and any side information (Z_1, Z_2) correlated with X_1, X_2.
5. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.
6. The experiment outputs 1 if $b' = b$, and 0 otherwise.

Good execution condition (non-triviality). As in the key secrecy experiment, define a residual-entropy threshold $k_1(\lambda)$ and let

$$\mathsf{Good} := \big\{\tilde{H}_\infty(X_i \mid M_i, Z_i) \geq k_1(\lambda) \text{ for } i = 1, 2\big\}.$$

Executions where $\neg\mathsf{Good}$ are considered trivial (helper-data leakage may trivially reveal linkage) and the experiment aborts.
The advantage of the adversary is given by:

$$\mathrm{Adv}^{\mathsf{unlink}}_{\mathcal{A}}(\lambda) := \big|\Pr[b' = b \mid \mathsf{Good}] - 1/2\big|.$$

Definition 1 (Unlinkability security). *A generic enrolment scheme* $(\mathsf{Enroll}, \mathsf{KDF})$ *is* $(\kappa(\lambda), \varepsilon_{\text{link}}(\lambda); k_1(\lambda))$-unlinkable *against an adversary class* $\mathfrak{A}$ *if for all* $\mathcal{A} \in \mathfrak{A}$*:*

$$\mathrm{Adv}^{\mathsf{unlink}}_{\mathcal{A}}(\lambda) \leq \varepsilon_{link}(\lambda)$$

conditioned on the non-trivial event Good*.*

4.2 Security Guarantees of LifeXP$^+$

We analyze the main security properties of our LifeXP$^+$ protocol: the secrecy of the derived keys and the unlinkability of multiple enrollments. These guarantees ensure that adversaries cannot distinguish keys from uniform or link different enrollments, even with access to public metadata and auxiliary information.

Theorem 1 (Key Secrecy of LifeXP$^+$). *Let* $(\mathsf{Gen}, \mathsf{Rep})$ *be a fuzzy extractor with extraction/privacy error* $\varepsilon_{\mathsf{FE}}$ *(i.e., when its input has min-entropy* $\geq k$ *the extractor output is* $\varepsilon_{\mathsf{FE}}$*-close to uniform conditioned on helper data). Assume the protocol's enrollment procedure* Enroll *produces public metadata* M *and a pre-extraction string* W *that (on the non-trivial executions) satisfies the extractor's entropy requirement* $\tilde{H}_\infty(W \mid M, Z) \geq k$*. Let* K *be the final key output by the protocol (the result of applying the fuzzy-extractor* Gen *and downstream KDF).*

Then, conditioned on the non-trivial (Good*) event* $\tilde{H}_\infty(W \mid M, Z) \geq k$*, for every PPT distinguisher* $\mathcal{A}$

$$\Big| \Pr\left[\mathcal{A}(K, M, Z) = 1 \mid \mathsf{Good}\right] - \Pr\left[\mathcal{A}(U_\kappa, M, Z) = 1 \mid \mathsf{Good}\right] \Big| \;\leq\; \varepsilon_{\mathsf{FE}}.$$

where U_κ *is uniform on* $\{0,1\}^\kappa$*. In particular,* K *is* $\varepsilon_{\mathsf{FE}}$*-indistinguishable from uniform given* (M, Z)*.*

Proof. We show a formal reduction from a key distinguisher to a fuzzy-extractor breaker.

Let $(\mathsf{Gen}, \mathsf{Rep})$ be a fuzzy extractor with privacy error $\varepsilon_{\mathsf{FE}}$ and correctness failure probability δ_{corr} when the input has min-entropy $\geq k$. Suppose a PPT adversary $\mathcal{A}$, given public metadata M and side-information Z, distinguishes the protocol key K from uniform with advantage

$$\mathrm{Adv}_{\mathcal{A}} = \Big| \Pr[\mathcal{A}(K, M, Z) = 1 \mid \mathsf{Good}] - \Pr[\mathcal{A}(U_\kappa, M, Z) = 1 \mid \mathsf{Good}] \Big| = \eta.$$

Then there exists a PPT algorithm $\mathcal{B}$ that breaks the fuzzy-extractor privacy guarantee with advantage at least $\eta - \delta_{\mathrm{corr}}$. Consequently, $\eta \leq \varepsilon_{\mathsf{FE}} + \delta_{\mathrm{corr}}$.

Reduction construction. Build $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance sampled from either the real distribution or the ideal distribution for the fuzzy-extractor privacy game:

$$\text{Challenge } \Xi := \begin{cases} (K^*, P, M, Z) & \text{(real)} \\ (U_\kappa, P, M, Z) & \text{(ideal)} \end{cases}$$

(Here P, M, Z are provided to $\mathcal{B}$ by the challenger; the bit determining real vs. ideal is unknown to $\mathcal{B}$.) The goal of $\mathcal{B}$ is to output a bit guessing whether Ξ is real or ideal.

Algorithm $\mathcal{B}$:

1. Receive challenge tuple $\Xi = (\tilde{K}, P, M, Z)$ from its challenger (either real or ideal).
2. Run $\mathcal{A}$ by giving it $(\tilde{K}, M, Z)$ as its view (i.e., treat $\tilde{K}$ as the protocol key and present the rest of the public metadata and side information unchanged).
3. When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs b' as its own guess (that the challenge was real if $b' = 1$ and ideal if $b' = 0$).

Analysis. Let $\Pr_{\text{real}}[\cdot]$ denote probability over the experiment where $\Xi = (K, P, M, Z)$ is drawn from the real distribution arising from running $\mathsf{Gen}(W)$ (under the Good event), and let $\Pr_{\text{ideal}}[\cdot]$ denote probability over the experiment where $\Xi = (U_\kappa, P, M, Z)$.

By construction of $\mathcal{B}$,

$$\Pr[\mathcal{B} \text{ outputs } 1 \mid \text{real}] = \Pr_{\text{real}}[\mathcal{A}(K, M, Z) = 1],$$

$$\Pr[\mathcal{B} \text{ outputs } 1 \mid \text{ideal}] = \Pr_{\text{ideal}}[\mathcal{A}(U_\kappa, M, Z) = 1].$$

Thus $\mathcal{B}$'s distinguishing advantage equals

$$\mathrm{Adv}_{\mathcal{B}} \;=\; \Big| \Pr_{\text{real}}[\mathcal{A}(K, M, Z) = 1] - \Pr_{\text{ideal}}[\mathcal{A}(U_\kappa, M, Z) = 1] \Big|.$$

Now relate $\mathrm{Adv}_{\mathcal{B}}$ to $\mathrm{Adv}_{\mathcal{A}} = \eta$. The subtlety is correctness failure: with probability at most δ_{corr}, Rep may fail and the real experiment's reconstructed key used in the protocol may differ from the idealized key value that Gen intended to bind to W. Concretely, condition on the Good event (min-entropy $\geq k$) we partition the real experiment outcomes by whether Rep would succeed for the corresponding noisy witness:

$$\begin{aligned}\Pr_{\text{real}}[\mathcal{A}(K, M, Z) = 1] = \Pr[&\mathsf{Rep} \text{ succeeds}] \\ &\cdot \Pr\left[\mathcal{A}(K, M, Z) = 1 \mid \mathsf{Rep} \text{ succeeds}\right] \\ &+ \Pr[\mathsf{Rep} \text{ fails}] \\ &\cdot \Pr\left[\mathcal{A}(K, M, Z) = 1 \mid \mathsf{Rep} \text{ fails}\right]\end{aligned}$$

Since $\Pr[\mathsf{Rep} \text{ fails}] \leq \delta_{\text{corr}}$, the difference between the real experiment conditioned on successful Rep and the unconditioned real experiment is at most δ_{corr} in statistical distance; hence

$$\begin{aligned}\Big| \Pr_{\text{real}}[\mathcal{A}(K, M, Z) = 1] &- \Pr[\mathcal{A}(K, M, Z) = 1 \mid \mathsf{Rep} \text{ succeeds}] \Big| \\ &\leq \delta_{\text{corr}}.\end{aligned}$$

By the definition of $\eta = \mathrm{Adv}_{\mathcal{A}}$ (which we take to be measured with respect to the conditioned-on-Good-and-successful-Rep experiment), we therefore have

$$\begin{aligned}\mathrm{Adv}_{\mathcal{B}} \;\geq\; \big|\, &\Pr[\mathcal{A}(K, M, Z) = 1 \mid \mathsf{Rep} \text{ succeeds}] \\ &- \Pr[\mathcal{A}(U_\kappa, M, Z) = 1]\big| \;-\; \delta_{\mathrm{corr}} \;=\; \eta - \delta_{\mathrm{corr}}.\end{aligned}$$

(Here we used that $\Pr_{\mathrm{ideal}}[\mathcal{A}(U_\kappa, M, Z) = 1] = \Pr[\mathcal{A}(U_\kappa, M, Z) = 1]$ and that η was defined relative to the conditioned-on-success experiment; if η was defined unconditionally an identical argument with slightly different bookkeeping yields the same bound up to δ_{corr}.)

Conclusion. If $\mathcal{A}$ has advantage η then $\mathcal{B}$ has advantage at least $\eta - \delta_{\mathrm{corr}}$ in breaking the fuzzy-extractor privacy guarantee. By the assumed security of the fuzzy extractor we have $\mathrm{Adv}_{\mathcal{B}} \leq \varepsilon_{\mathsf{FE}}$, hence

$$\eta \leq \varepsilon_{\mathsf{FE}} + \delta_{\mathrm{corr}}.$$

This completes the formal reduction.
Below, we give a heuristic argument why the Good event is plausible and achievable in practice.

- **Source composition:** the pre-image (S, B) is formed from two (largely) different modalities: a personal story S (semantic/textual variability) and a biometric B (physical/high-dimensional uniqueness). Let

  $$k_S := \tilde{H}_\infty(S \mid Z), \qquad k_B := \tilde{H}_\infty(B \mid Z).$$

  Absent extreme side-information Z, these contributions are (approximately) additive: the joint source yields roughly $k_S + k_B$ bits of uncertainty from the adversary's perspective.
- **Encoders preserve entropy directionally:** The sentence and biometric encoders map inputs to high-dimensional vectors. Provided the encoders do not collapse many distinct users to the same vector (an empirical design requirement), directional variation in the embedding space preserves a large fraction of the original entropy.
- **LSH / clustering / quantization tradeoff:** these steps intentionally reduce intra-user variability while keeping inter-user distinctions. They do reduce entropy, but this reduction can be bounded and controlled by parameters (number of hyperplanes m, quantization resolution, cluster-codebook size). Let $\mathrm{leak}_{\mathsf{LSH}}$ denote the conservative bit-budget of information lost due to these coarsening steps (including any published sketches/seeds).
- **Helper-data leakage accounting:** the fuzzy extractor helper data P leaks at most the ECC redundancy (or a known analytic bound). Let leak_P denote that leakage. Define total conservative leakage

  $$\mathrm{leak}(M) := \mathrm{leak}_{\mathsf{LSH}} + \mathrm{leak}_P + (\text{other published bits}).$$

- **Resulting residual entropy estimate:** combining the above heuristics yields the rough lower bound

  $$\tilde{H}_\infty(W \mid M, Z) \;\gtrsim\; k_S + k_B - \mathrm{leak}(M).$$

Thus, by designing parameters so that $k_S + k_B - \mathrm{leak}(M) \geq k$ (the fuzzy extractor's required min-entropy), Good will hold.

□

Remarks.

- The reduction is tight up to the additive correctness term δ_{corr}; this is inherent because failure of Rep can make the real experiment's key distribution differ from the ideal one independent of extractor privacy.
- If one wishes to include offline guessing in the reduction explicitly: an adversary that succeeds by enumerating candidate W values (making q guesses) can be converted into an extractor breaker only if those guesses imply a statistical distinction between (K, P, M, Z) and (U_κ, P, M, Z). The simple bound $p_{\mathrm{succ}}(q) \leq q \cdot 2^{-k}$ (from min-entropy k) shows that enumeration-based attackers cannot achieve advantage larger than $q \cdot 2^{-k}$ unless the extractor's security is violated; this is compatible with the reduction above which binds any distinguisher's advantage to $\varepsilon_{\mathsf{FE}} + \delta_{\mathrm{corr}}$.

Theorem 2 (User-Key Unlinkability). *Let the user-derived key protocol be instantiated with a fuzzy extractor* $(\mathsf{Gen}, \mathsf{Rep})$ *that is* $(\kappa, \varepsilon_{\mathsf{FE}})$*-secure. Let the per-enrollment randomness* r_{LSH} *be independently chosen for each enrollment. Then, for any PPT adversary* $\mathcal{A}$ *attempting to link two different enrollments of users (either the same or different) based on the public metadata* M_1, M_2 *and side information* Z*, the advantage of* $\mathcal{A}$ *in distinguishing "same user" vs "different user" is at most*

$$\mathrm{Adv}^{\mathsf{unlink}}_{\mathcal{A}}(\lambda) \;\leq\; \varepsilon_{\mathsf{FE}} + 2\delta_{\mathrm{corr}}.$$

Proof. Now we provide the proof sketch of Theorem 2.

Proof sketch of Theorem 2. The key idea is that unlinkability relies on two sources of randomness:

- *Per-enrollment randomness:* By choosing independent r_{LSH} for each enrollment, the LSH/quantization output (and hence the fuzzy-extractor input W) is decorrelated across enrollments even for the same user.
- *Fuzzy extractor security:* The fuzzy extractor guarantees that the extracted key K is statistically close to uniform given the public helper data P. Hence, for independently chosen r_{LSH}, two enrollments produce keys and public data that are nearly independent and indistinguishable from the ideal uniform distribution.

Conditioning on the Good event where the inputs have sufficient min-entropy, the residual correlation between two enrollments is negligible. Correctness failure in Rep contributes at most δ_{corr} per enrollment. Hence, the total linking advantage is bounded by $\varepsilon_{\mathsf{FE}} + 2\delta_{\mathrm{corr}}$.

Intuitively, even if the same user enrolls twice, the independent randomization ensures that the public metadata and the derived keys reveal almost no information linking the two instances. Similarly, different users' enrollments remain statistically independent, satisfying the unlinkability property.

Remark. While unlinkability is not the core security requirement of our protocol, it remains a desirable privacy property. In our design, the per-enrollment randomness r_{LSH} used for locality-sensitive hashing is device-specific; losing it would prevent recovery and defeat the purpose of the scheme. However, r_{LSH} can be safely replicated to private user cloud or backup locations without harming key secrecy. Even if r_{LSH} were compromised, the worst an adversary could do is to break unlinkability between enrollments; the secrecy of the derived key remains intact.

5 Implementation

We implemented the protocol in ∼300 lines of Python code. The pipeline supports fused (text+biometrics) operation and works offline in its entirety.

To showcase that our protocol's properties hold in real-word scenarios, we employ the following two publicly available, independent datasets:

- **Hippocorpus V3** [20]: a set of short personal narratives collected under controlled prompts. For each *pair ID* there are three variants:
  - *Recalled*: participant recounts a real personal event from memory.
  - *Retold*: the same participant retells the same event in slightly different words.
  - *Imagined*: the participant invents a fictional event but with the same theme as the corresponding recalled story (i.e., belonging to the same pair ID).

  These enable our three evaluation regimes: *honest recovery* (recalled → retold, same subject), *strong adversary* (imagined, same subject), and *weak adversary* (a random story from a different subject).
- **FVC2002-DB1** [14]: a benchmark fingerprint dataset with multiple impressions per finger. We pair impressions to the same/other subjects according to the above honest/strong/weak regimes.

Architecture. Given a story and a fingerprint image, the pipeline proceeds as follows:

1. **Text embedding:** We use a locally stored Sentence-BERT model to embed the story into a 768-dimensional vector.
2. **Biometric embedding:** A DeepPrint extractor [12] produces a fixed-length texture embedding from the fingerprint image.
3. **Fusion:** In fusion mode, the L2-normalized text and biometric embeddings are concatenated with a tunable weight α on the biometric component, and the result is re-normalized.
4. **Quantization:** We apply random-hyperplane LSH (SimHash) to project the embedding to a binary string of length n (typically $n \in \{32, 48, 64, 80\}$).
5. **Error correction:** A Reed–Solomon error-correcting code encodes the enrollment codeword and produces a public helper string. At recovery, the helper is used to decode and correct bit errors up to the configured parity limit.

6. **Key derivation:** The corrected bitstring is hashed with SHA-256; matching hashes indicate successful recovery.

Environment and Performance. All models (Sentence-BERT and DeepPrint) are stored locally and executed entirely on-device. Experiments were run on a Macbook M1 Pro laptop with 32 GB RAM. On the above CPU, average runtimes per sample are: story embedding 209 ms, fingerprint embedding 168 ms, SimHash projection 0.7 ms, Reed–Solomon decoding under 0.1 ms, and end-to-end enrollment or recovery 377.9 ms. All timings are for single-threaded CPU execution.

Accuracy. We evaluate over 100 random honest, strong, and weak trials per parameter configuration. The *honest success rate* is the proportion of genuine retold or recalled pairs that are successfully recovered. The *strong adversary success rate* is the proportion of imagined or same-subject impostor attempts that are incorrectly accepted. The *weak adversary success rate* is the proportion of different-subject impostor attempts that are incorrectly accepted. In our parameter search, we identify configurations where the weak adversary success rate is **exactly 0%** and honest–strong separation is maximized. For example, for $n = 32$ bits, $\alpha = 3.0$, and RS parity $= 4$, we achieve honest success $\approx 15\%$, weak $= 0\%$, and strong–honest Hamming separation of ≈ 3.9 bits.

Note: Although the honest recovery success rate under these strict security parameters may appear low, this is not a major usability issue. In a real-world deployment, a legitimate user can attempt recovery multiple times, re-telling the same memory with different wording until successful, meaning the success rate can be easily amplified (especially given our pipeline executes in less than a second on commodity hardware). By contrast, we intentionally choose parameters that keep the weak adversary success rate at **zero**, as this is critical for the security of the scheme, even by many adversarial retries.

Accuracy under Multiple Attempts. A primary design goal of our system is to amplify the honest success rate by allowing users multiple recovery attempts (this is in line with the decentralized web3/blockchain setting, where no limit of "key guesses" exists). To evaluate this amplification effect, we needed to simulate a user recalling their memory several times. However, the standard Hippocorpus dataset, while valuable for single-attempt analysis, is limited in this regard as it provides only one recalled version for each original story. To overcome this limitation, we created a synthetic dataset to model a realistic multi-attempt scenario. For each of the stories in the Hippocorpus dataset, we prompted the GPT-4o language model to generate six distinct, paraphrased retellings. This synthetic data allows us to measure the cumulative success rate, $S(k)$, as a user makes up to $k = 4$ attempts. Using our baseline "High-Security" configuration ($\alpha = 3.0$, $n = 32$ bits, 'rs_parity' $= 4$), which guarantees a 0% weak adversary success rate, the success rate amplifies to 33.8%. Crucially, for the users who do succeed, the process is extremely efficient. Therefore, the expected number of attempts until success, $E[k|\text{success}]$, is only 1.67, indicating a fast and frictionless experience.

Entropy Analysis. For autobiographical secrets, "guessing" entropy [17] (the expected number of guesses an optimal adversary requires) is a more meaningful security metric than strict min-entropy. It captures the full distribution of possible responses rather than only the single most likely value, and provides a realistic measure of the effort needed for an offline guessing attack. However, obtaining accurate guessing entropy estimates for our approach would require a large-scale user study to model human retelling variability, which is out of scope for this work. Despite this limitation, LifeXP$^+$ even in the most conservative scenario can operate solely on a high-entropy biometrics such as the iris, which as analyzed by Daugman [10], exhibit approximately 249 bits of effective degrees of freedom for a single eye, and have been deployed at scale in systems such as Worldcoin [5]. Therefore, LifeXP$^+$ by combining the biometric with the cognitive component can comfortably exceed the min-entropy target typically required for blockchain private keys, providing a substantial security margin.

Limitations. In principle, a more valid evaluation would require a user study in which participants repeatedly retell the same memory over long periods of time. To our knowledge, however, no dataset of naturally occurring multiple retellings exists, and constructing one would impose a significant logistical burden. Genuine memory drift typically emerges over weeks or months, not minutes, meaning that participants would need to return for many spaced recall sessions. Conducting such a longitudinal study at scale would therefore require high participant retention, extended timelines, and substantial resources. Given these practical constraints, we use GPT-4o–generated paraphrases as a stand-in for the natural variability that would arise from multiple human retellings of the same memory.

6 Conclusion

We presented LifeXP$^+$, a key recovery protocol for blockchain settings that combines autobiographical user memory with biometric features, enabling secure binding of cryptographic keys to human-verifiable secrets. Our design intentionally prioritises security by selecting parameters that achieve a *zero* weak-adversary success rate, even at the cost of a lower honest recovery rate, which can be amplified in practice by allowing multiple retelling attempts. We implemented and evaluated the system on public story and fingerprint datasets, and quantified the trade-offs between honest success, adversary resistance, and the honest–strong separation. Looking ahead, we believe that such human-centric, multi-modal recovery mechanisms are well-suited to become the next generation of BIP-39 in the era of generative AI, where traditional mnemonic phrases face increasing risks from large-scale language modelling and social engineering.

Disclaimers

Case studies, comparisons, statistics, research and recommendations are provided "AS IS" and intended for informational purposes only and should not be relied upon for operational, marketing, legal, technical, tax, financial or other advice. Visa Inc.

neither makes any warranty or representation as to the completeness or accuracy of the information within this document, nor assumes any liability or responsibility that may result from reliance on such information. The Information contained herein is not intended as investment or legal advice, and readers are encouraged to seek the advice of a competent professional where such advice is required. These materials and best practice recommendations are provided for informational purposes only and should not be relied upon for marketing, legal, regulatory or other advice. Recommended marketing materials should be independently evaluated in light of your specific business needs and any applicable laws and regulations. Visa is not responsible for your use of the marketing materials, best practice recommendations, or other information, including errors of any kind, contained in this document. All trademarks are the property of their respective owners, are used for identification purposes only, and do not necessarily imply product endorsement or affiliation with Visa.

References

1. Argent vault security and recovery, `https://support.argent.xyz/hc/en-us/articles/360007338877-How-to-recover-my-wallet-with-guardians-complete-guide`
2. A man who says he threw away a hard drive loaded with 7,500 bitcoins in 2013 is offering his city $70 million to dig it up from the dump, `https://www.businessinsider.com/man-offers-council-70-million-dig-up-bitcoin-hard-drive-2021-1`
3. Zengo wallet recovery, `https://zengo.com/understand-the-difference-zengo-wallet-recovery-vs-ledger-recover/`
4. Blackshear, S., Chalkias, K., Chatzigiannis, P., Faizullabhoy, R., Khaburzaniya, I., Kokoris-Kogias, E., Lind, J., Wong, D., Zakian, T.: Reactive key-loss protection in blockchains. In: Bernhard, M., Bracciali, A., Gudgeon, L., Haines, T., Klages-Mundt, A., Matsuo, S., Perez, D., Sala, M., Werner, S. (eds.) FC 2021 Workshops. LNCS, vol. 12676, pp. 431–450. Springer, Berlin, Heidelberg (Mar 2021). https://doi.org/10.1007/978-3-662-63958-0_34
5. Bloemen, R., Kales, D., Sippl, P., Walch, R.: Large-scale MPC: Scaling private iris code uniqueness checks to millions of users. Cryptology ePrint Archive, Report 2024/705 (2024), `https://eprint.iacr.org/2024/705`
6. Chalkias, K.: Compressed mnemonic format that reduces from 12 to 8 words without loss of entropy. `https://x.com/kostascrypto/status/1717281228800630857` (Oct 2023), accessed September 2025
7. Charikar, M.: Similarity estimation techniques from rounding algorithms. In: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. pp. 380–388. ACM (2002)
8. Chatzigiannis, P., Chalkias, K., Kate, A., Mangipudi, E.V., Minaei, M., Mondal, M.: SoK: Web3 recovery mechanisms. Cryptology ePrint Archive, Report 2023/1575 (2023), `https://eprint.iacr.org/2023/1575`
9. Chatzigiannis, P., Wang, K.C., Arora, S.S., Minaei, M.: A composability analysis framework for Web3 wallet recovery mechanisms. In: Blanton, M., Enck, W., Nita-Rotaru, C. (eds.) 2025 IEEE Symposium on Security and Privacy. pp. 1531–1546. IEEE Computer Society Press (May 2025). https://doi.org/10.1109/SP61157.2025.00158

10. Daugman, J.G.: How iris recognition works. IEEE Transactions on Circuits and Systems for Video Technology **14**(1), 21–30 (2004)
11. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Berlin, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_31
12. Engelsma, J.J., Cao, K., Jain, A.K.: Learning a fixed-length fingerprint representation (2019), https://arxiv.org/abs/1909.09901
13. Lindell, Y.: Cryptography and mpc in the coinbase prime web3 wallet (2023), https://coinbase.bynder.com/m/68b5c8bedf49a9f9/original/CB-Prime-W3W-MPC-White-Paper.pdf
14. Maio, D., Maltoni, D., Cappelli, R., Wayman, J., Jain, A.: Fvc2002: Second fingerprint verification competition. In: 2002 International Conference on Pattern Recognition. vol. 3, pp. 811–814 vol.3 (2002). https://doi.org/10.1109/ICPR.2002.1048144
15. Mangipudi, E.V., Chatzigiannis, P., Chalkias, K., Kate, A., Minaei, M., Mondal, M.: Web3 recovery mechanisms and user preferences. Cryptology ePrint Archive, Paper 2025/1687 (2025), https://eprint.iacr.org/2025/1687
16. Marek, P., Pavol, R., Aaron, V., Sean, B.: Mnemonic code for generating deterministic keys. https://github.com/bitcoin/bips/tree/master/bip-0039 (2013), [Online; accessed September 2025]
17. Massey, J.: Guessing and entropy. In: Proceedings of 1994 IEEE International Symposium on Information Theory. pp. 204– (1994). https://doi.org/10.1109/ISIT.1994.394764
18. Salakhutdinov, R., Hinton, G.E.: Semantic hashing. International Journal of Approximate Reasoning **50**(7), 969–978 (2009). https://doi.org/10.1016/j.ijar.2008.11.006
19. Schmidt, R., Mota, M., Buterin, V., naxe: Ethereum EIP2492 - secret multisig recovery (2019), https://gitlab.com/status-im/docs/EIPs/blob/secret-multisig-recovery/EIPS/eip-2429.md
20. W. Bos, M., Horvitz, E., Choi, Y., A. Smith, N., W. Pennebaker, J.: Recollection versus imagination: Exploring human memory and cognition via neural language models. In: ACL 2020 (February 2020), https://www.microsoft.com/en-us/research/publication/recollection-versus-imagination-exploring-human-memory-and-cognition-via-neural-language-models/
21. Woo, S., Kaiser, E., Artstein, R., Mirkovic, J.: Life-experience passwords (leps). In: Proceedings of the 32nd Annual Conference on Computer Security Applications. p. 113–126. ACSAC '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2991079.2991107, https://doi.org/10.1145/2991079.2991107
22. Wu, Z., Kang, J., Jiang, Q.: Semantic key generation based on natural language. International Journal of Intelligent Systems **37**(7), 4041–4064 (2022). https://doi.org/https://doi.org/10.1002/int.22711, https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22711
23. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview. Tech. rep. (2018-10-03 2018). https://doi.org/https://doi.org/10.6028/NIST.IR.8202