1 Introduction

Sources of reproducible uniformly distributed random strings are necessary for many cryptographic applications. Such string can be used to create a secret key or serve as a seed to generate a public/private key pair.

Biometric data, have been suggested as a potential source to enable strong, cryptographically-secure authentication of human users. The appeal of biometric data as cryptographic secret stems from their high entropy and the fact that biometric characteristics cannot be lost or forgotten. However, before such data can be used in existing cryptographic protocols, two issues must be addressed: first, biometric data are not uniformly distributed and hence do not offer provable security guarantees if used. Secondly, biometric data are not exactly reproducible, as two biometric readings of the same feature are rarely identical. For example, when using a fingerprint as an encryption key, we have the problem that if we read the fingerprint later again, we might get a slightly different key. Thus, the key cannot be used to decrypt. To utilize such data, it is required to solve these issues, in order to generate the same value in subsequent readings.

This paper explores how to use biometric data or other noisy data to securely derive cryptographic keys which could then be used, in particular, for the purposes of authentication. Two primitives are introduced: a secure sketch which allows recovery of the secret key given a close approximation thereof, and a fuzzy extractor which extracts a uniformly distributed string from this secret key in an error-tolerant way. Both primitives work by outputting a public string which is stored by the server and transmitted to the user. The primitives are designed so as to be secure even when an adversary learns the value of this public string. In addition, optimal constructions of both primitive for three different metrics are also derived. This work is based on paper [1]. The purpose of this report is to make the technical results in the original paper more accessible and to describe their main results. In general, Section 2 introduces two primitives and states the main definitions and related notions. In the following, Section 3 provides constructions of secure sketches and fuzzy extractor using the Hamming distance, set difference and edit distance metrics. In Section 4, an implementation of the fuzzy extractor for password authentication with biometric data is given.
2 Preliminaries

We review the definitions and notations from [1] using slightly different terminology.

2.1 Basic definitions

We denote random variables by capital letters and corresponding lowercase letters for their samples. Unless explicitly stated otherwise, all logarithms are base 2. We use $U_l$ to denote the uniform distribution on $l$-bit binary strings.

Let $M = \{0, 1\}^n$ be a finite dimensional metric space consisting of biometric inputs, with a distance function $\text{dis}: M \times M \rightarrow \mathbb{R}^+$, which calculates the distance between two given inputs based on the metric chosen. Let $l$ be the number of bits of the extracted output string $R$ from biometric input $\omega$, and $t$ be the error threshold value (i.e., for two inputs $\omega, \omega' \in M$ has $\text{dis}(\omega, \omega') \leq t$). Thus, given the same metric space $M$, the distance for different metrics will be defined as follows:

- **Hamming metric.** Given two inputs $\omega, \omega' \in M$, the $\text{dis}(\omega, \omega')$ is the number of positions in which they differ.
- **Set Difference metric.** Here $M$ consists of all $s$-element subsets in a universe $U = [n] = 1, ..., n$. The distance between two sets $A, B$ is the number of points in $A$ that are not in $B$. Since $A$ and $B$ have the same size, the distance is half of the size of their symmetric difference: $\text{dis}(A, B) = \frac{1}{2} |A \triangle B|$.
- **Edit metric.** Given two inputs $\omega, \omega' \in M$, the $\text{dis}(\omega, \omega')$ is one half of the smallest number of character insertions and deletions needed to transform $\omega$ into $\omega'$.

One way to measure the uniformity of a distribution is to use the concept of min-entropy. If the predictability of a random variable $A$ is $\max_a P[A = a]$, then the min-entropy is defined as:

$$H_{\infty}(A) = -\log(\max_a P[A = a]) \tag{1}$$

Let's consider a pair of random discrete variable $A$ and $B$. The average min-entropy of $A$ given $B$ is defined as:

$$H_{\infty}(A|B) = -\log \left( E_{\text{b} \rightarrow B} [2^{-H_{\infty}(A|B)}] \right) \tag{2}$$

Generally, min-entropy can be viewed as the worst-case entropy and the average min-entropy can be used to measure the average-case entropy of $A$ given $B$. For a random variable $X$, the entropy loss of $A$ given $B$ is defined as $L = H_{\infty}(A) - H_{\infty}(A|B)$. This definition is useful, since for any $l$-bit string $B$, we have $H_{\infty}(A|B) \geq H_{\infty}(A) - l$, which means that we can bound $L$ from above by $l$ without having to know the distributions of $A$ and $B$. Also, we use
\[ \text{SD}(A, B) \] to denote the statistical difference between two probability distributions \( A \) and \( B \) such that:

\[
SD(A, B) = \frac{1}{2} \sum_{v} \left| \Pr[A = v] - \Pr[B = v] \right|
\] (3)

Here we repeat the definition of strong randomness extractors given by Dodis et al. [1]. Intuitively, a strong randomness extractor is a randomized function that transforms its input from any biased distribution of sufficient min-entropy into an output that yields an almost uniform distribution. A randomness extractor can be constructed by so-called universal hash functions.

**Definition 1.** An efficient \((n, m, l, \epsilon)\)-strong extractor is a polynomial time probabilistic function \( \text{Ext} : \{0,1\}^n \rightarrow \{0,1\}^l \) such that for all min-entropy \( m \) distributions \( W \), we have \( \text{SD}(\text{Ext}(W; X), X, (U_l, X)) \geq \epsilon \), where \( \text{Ext}(W; X) \) stands for applying \( \text{Ext} \) to \( W \) using (uniformly distributed) randomness \( X \).

As shown in [1], the theoretical limit about the number of bits that strong extractors can extract, is given by \( l \leq m \cdot 2 \log(1/\epsilon) + O(1) \).

Many constructions utilize error-correcting codes. A code \( C \) is a subset of \( k \) elements \( \{\omega_1, \omega_2, ..., \omega_{k-1}\} \) of \( M \). The minimum distance of \( C \) is the smallest \( d \) such that \( \text{dis}(\omega_i, \omega_j) \geq d \) for all \( i \neq j \), which implies that the code can detect up to \((d-1)\) errors; and the error-correcting distance is \( t = \lfloor (d-1)/2 \rfloor \).

### 2.2 Secure Sketches and Fuzzy Extractors

Dodis et.al [1] introduce the concept of two primitives as follows:

**Definition 2.** An \((M, m, m', t)\)-secure sketch is a pair of randomized procedures \((\text{SS}, \text{Rec})\) such that the following holds:

- **Sketch:** \( \text{SS} \) takes an input \( \omega \in M \) and returns the sketch \( S \in \{0,1\}^* \). \( \text{Rec} \) takes as input \( \omega' \in M \) and \( S \in \{0,1\}^* \) and outputs the original \( \omega \).
- **Correctness:** For all pairs of points \( \omega, \omega' \in M \), if \( \text{dis}(\omega, \omega') \leq t \), then \( \text{Rec}(\omega', S) = \omega \).
- **Security:** The min-entropy \( m \) of \( W \) given \( S \) is high. Then, for any random variable \( W \) over \( M \), the average min-entropy \( m' \) is: \( H_\infty (W|S) \geq m' \).

The key notion of a secure sketch is that given a ”noisy” version \( \omega' \in M \), the original input \( \omega \in M \) would be recovered by a recovery procedure in such a way that the entropy of \( \omega \) remains high.

Security of a secure sketch is evaluated in terms of entropy of \( W \) before, \( m \), and after, \( m' \), releasing the string \( P \). Therefore, \((m - m')\) is called the entropy loss of a secure sketch associated with making \( S \) public. A secure sketch allows one to correct errors in \( W \), while giving up the smallest entropy about \( W \), which is exactly the entropy loss. It is easy to see by the definition of the average min-entropy, that one way to bound the entropy loss is to make the length of the sketch \( S \) as small as possible. Indeed, based on the entropy loss definition,
the size of the sketch $S$ can be easily obtained regardless of the distribution of $W$. Regarding the correctness property, we note that no guarantee is provided about the output of $Rec$ when $\text{dis}(\omega, \omega') > t$.

While secure sketches address the issue of error tolerance, they do not address the issue of the possible non-uniformity of $W$. Fuzzy extractors, on the other hand, address both.

Fuzzy extractors allow one to extract randomness $R$ from $\omega$ and later reproduce it exactly using $\omega'$ close to the original $\omega$. In other words, a fuzzy extractor does not necessarily recover the original input.

**Definition 3.** An $(M, m, m', l, t, \epsilon)$-fuzzy extractor is a pair of randomized procedures $(Gen, Rep)$:

- **Operation:** $Gen$ outputs a string $R \in \{0, 1\}^l$ and a helper string $P$ from $\omega$. $Rep$ takes as input $\omega' \in M$ and $P \in \{0, 1\}^l$, in order to recover $R$.
- **Correctness:** If $\text{dis}(\omega, \omega') \leq t$ and $(R, P) \leftarrow Gen(\omega)$, then $Rep(\omega', P) = R$.
- **Security:** The string $R$ is nearly uniform even given $P$. Therefore, if $(R, P) \leftarrow Gen(\omega)$, it holds that $SD((R, P), (U_l, P)) \leq \epsilon$

A secure sketch and a fuzzy extractor are considered to be efficient, if the corresponding procedures are run in expected polynomial time.

The security requirement for the fuzzy extractor is that, for any $W$ of min-entropy $m$, the statistical distance between the distribution of $R$ and the uniform distribution of strings of the same length is no greater than $\epsilon$, even after observing $P$.

We note that in the above definitions of secure sketches (and fuzzy extractor), the errors are selected before knowing $S$ ($P$ for fuzzy extractor). However, if the error between $\omega$ and $\omega'$ depends on the output of $SS$ ($Gen$ for fuzzy extractors), then there is no guarantee whether the correctness property is achieved. Also, we note that the fuzzy extractor definition make no guarantee when $\text{dis}(\omega, \omega') > t$. Although, $Rep$ procedure may not output an incorrect string $R$, it might not terminate. Therefore, the behavior of $Rep$ is not clear when $\text{dis}(\omega, \omega') > t$.

As shown in Lemma 1 [1] using slightly different terminology, a fuzzy extractor can be constructed on top of a secure sketch by applying a strong extractor $Ext$. The construction of the fuzzy extractor from sketches is depicted in Fig. 1.

**Lemma 1.** Assume $SS$ is a $(M, m, m, t)$-secure sketch with recovery procedure $Rec$, and let $Ext$ be the $(n, m, l, \epsilon)$-strong extractor based on pairwise-independent hashing (in particular, $l = m' - 2\log 1/\epsilon$). Then the following $(Gen, Rep)$ is a $(M, m, l, t, \epsilon)$-fuzzy extractor where:

- $Gen(\omega, i)$: set $P = (SS(\omega), i)$, $R = Ext(\omega, i)$, output $(R, P)$ (where we choose a random $i$ in $Gen$).
- $Rep(\omega', (s, i))$: recover $\omega = Rec(\omega', s)$ and output $R = Ext(\omega; i)$.

The correctness property of the above construction of a $(M, m, l, t, \epsilon)$-fuzzy extractor is guaranteed by the correctness property of the used $(M, m, m, t)$-
secure sketch. The security property is achieved by using a \((n, m, l, \epsilon)\)-strong extractor. In other words, the strong extractor compresses the remaining entropy, after publishing \(P\), into a random output \(R\) which can be used as a secret key.

### 3 Construction of secure sketches and fuzzy extractor

Dodis et al.\[1\] proposed constructions of secure sketches for three different metrics: Hamming distance, set difference and edit distance. However, only the Hamming distance construction is explained in details in the following sections.

#### 3.1 The Hamming distance construction

In this section, we will consider two constructions of secure sketches for the Hamming distance metric. Both constructions are based on the use of binary error-correcting linear block code. Hamming distance (i.e., the number of bits positions that differ between \(\omega\) and \(\omega'\)) is perhaps the most natural metric to consider when constructing sketches and fuzzy extractors.

Let \(C\) be an \([n, k, t]\) error-correcting linear block. We shall consider \(G_{k \times n}\) as its generator matrix and \(H_{(n-k) \times n}\) as its corresponding parity-check matrix with \(G \cdot H^T = 0\). Then, we use this code to construct a \((M, m, m + k - n, t)\)-secure sketch over the Hamming space \(M = \{0, 1\}^n\), as it is demonstrated by the following two constructions:

**The Code-Offset construction:**

- **\(SS(y) \rightarrow v\):** the sketching procedure generates a helper data string \(v = y \oplus c\), where \(c \in C\).
- **\(Rec(y', v) \rightarrow y''\):** the recovery procedure computes a noisy codeword \(c' = y' \oplus v = (y' \oplus y) \oplus c\). Then, it applies the error-correcting procedure on \(c'\) to get the new correcting codeword \(c''\). As a result, the recovered value is: \(y'' = v \oplus c'' = y \oplus (c \oplus c'')\).
- **Correctness property:** If \(\text{dis}(y, y') \leq t\), then \(\text{dis}(c, c') \leq t\), and the correction procedure can correct \(c'' = c\) and hence, \(y'' = y\).
- **Security property:** This property follows from the fact that, \(v\) discloses at most \(n\) bits of which \(k\) are independent from \(y\). We have that \(k\), such that \(k < n\). Knowing this fact, we have that \(H_{\infty}(Y | V) \geq H_{\infty}(Y) - (n - k) = m - n + k\).
The Syndrome construction:

- **$SS(y) \rightarrow v$:** the sketching procedure generates a helper data string $v = y \cdot H^T$.
- **$Rec(y', v) \rightarrow y''$:** the recovery procedure computes the syndrome $s$ of $C$ as: $y' \cdot H^T \oplus v = (y' \oplus y) \cdot H^T$. The syndrome decoding procedure finds the error $e$ by computing: $s = e \cdot H^T$. As a result, the recovered value is: $y'' \rightarrow y' \oplus e$.

**Correctness property:** If $\text{dis}(y, y') \leq t$, then $e = y \oplus y'$ and $y'' = y$.

**Security property:** This property follows by applying Lemma 3 described in the paper[1].

By using the construction in Lemma 1 with one of the two constructions of secure sketches above, we immediately obtain a $(M, m, l, t, \epsilon)$-fuzzy extractor $(Gen, Rep)$ where $l = m + k - 2 - 2\log(1/\epsilon)$ and $t$ is the Hamming distance.

### 3.2 Set difference construction

Dodis et al.[1] has proposed several constructions under this metric. The entropy loss by all these schemes are roughly the same. They differ in the sizes of the sketches, decoding efficiency and also the degree of ease in practical implementation. We would only consider the construction of secure sketches for large universe size. This syndrome-based construction is called Pinsketch.

**Pinsketch construction:**

- Let $\omega = \{\omega_1, \omega_1', ..., \omega_s\}$ be a given biometric template.
- **$SS(\omega) = syn(\omega)$** is the sketching procedure that generates the helper data $(s_1, s_2, ..., s_{2t-1})$, where $\omega_i = \sum_{j=1}^{s} \omega_{ij}$ and $t$ is the error-tolerance.
- Given a query $\omega' = \{\omega'_1, \omega'_1', ..., \omega'_s\}$, the sketch generates the syndrome of $\omega'$ as: $syn(\omega') = (s'_1, s'_2, ..., s'_{2t-1})$.
- **$Rec(\omega', SS(\omega)) = supp(v) \Delta \omega'$**, which is used to retrieve the template $\omega$. Here, $supp(v) \Delta \omega' = supp(v) \cup \omega' - supp(v) \cap \Delta \omega'$ and $supp(v)$ is defined as the positions in which $v$ is nonzero and it can be computed through syndrome: $syn(v) = (s'_1 - s_1, s'_2 - s_2, ..., s'_{2t-1} - s_{2t-1})$.
- The construction guarantees that $Rec(\omega', SS(\omega)) = \omega$, if $\text{dis}(\omega, \omega') \leq t$.

### 3.3 Edit distance construction

Here, $\omega$ and $\omega'$ are strings of arbitrary lengths over some alphabet and the distance between $\omega$ and $\omega'$ is defined as the smallest number of character insertions and deletions that transform $\omega$ into $\omega'$.

There are two alternative ways proposed by [1] to realize the secure sketch construction for the edit distance metrics. The first consists of applying an existing low-distortion embedding (that does not significantly change the distance between two biometrics after the mapping) of the edit distance into the Hamming distance and then using a syndrome construction for the Hamming distance to produce the public data. The second includes the application of a specially
designed embedding of the edit distance into the set difference metric using so-called c-shinglings.

After the embedding of a biometric is performed, the third construction represented in set difference construction is applied to the resulting representation to compute the sketch. Note that in both of the above cases, linear error-correcting codes are used.

4 Implementation of a Fuzzy Extractor with Smart Card

In this section, we describe a remote biometric authentication scheme using fuzzy extractor with smart card, which is proposed by Zhang et al. [2].

This scheme, which is mutual authentication scheme, contains four phases: registration phase, login phase, authentication and key agreement phase and the password and biometrics update. We use this scheme to illustrate how could a fuzzy extractor implemented for secure authentication.

We start describing the scheme by giving firstly, the following notations:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>Remote server</td>
<td>$h(g)$</td>
<td>One-way hash function</td>
</tr>
<tr>
<td>$U_i$</td>
<td>$i$-th user</td>
<td>$PW_i$</td>
<td>$U_i$'s Password</td>
</tr>
<tr>
<td>$ID_i$</td>
<td>$U_i$'s identity</td>
<td>$PW_i'$</td>
<td>New password</td>
</tr>
<tr>
<td>$B_i$</td>
<td>Registered biometrics</td>
<td>$p,q$</td>
<td>Two large coprime numbers</td>
</tr>
<tr>
<td>$B_i^*$</td>
<td>Imprinted biometrics</td>
<td>$Z_p^<em>Z_q^</em>$</td>
<td>$p$-order or $q$-order group</td>
</tr>
<tr>
<td>$B_i^*$</td>
<td>Updated biometrics</td>
<td>$\oplus$</td>
<td>Bitwise XOR computation</td>
</tr>
<tr>
<td>$x$</td>
<td>Secret value of $S$</td>
<td>$P$</td>
<td>Concatenation operation</td>
</tr>
</tbody>
</table>

Fig. 2. Notations [2]

1. The registration phase:
Before the legitimate user $U_i$ starts login and authentication phase, $U_i$ and the remote server $S$ should perform the registration phase as follows. Fig 2 describes the detailed steps of registration phase.

(1) $U_i$ chooses user’s $ID_i$ and $PW_i$. And then, $U_i$ imprints his biometric information $B_i$ to the fuzzy extractor and the generation procedure $Gen$ computes $Gen(B_i) \rightarrow (R, P)$. Smart card computes $h(R)$ and $h(R||PW_i)$. $U_i$ sends the user’s $ID_i$ and the hashed values calculated by the smart card to the remove server $S$ through a secure communication channel.

(2) $S$ uses user’s message and secret key $x$, to compute $S_1, S_2, S_3$, each of one has the corresponding value shown in the Fig 2. Then it stores the message
\{S_3, h(\cdot), p, q\} into the smart card and after that, it sends this message to the user \( U_i \) through the channel.

![Diagram](image)

**Fig. 3.** Proposed scheme: registratrion phase [2]

(3) After the user receives the smart card, \( U_i \) inputs \{\( S_3, h(\cdot), h(R), p, q\)\} into the smart card.

Note how the generation procedure \( Gen \) on a given biometric input \( B \) outputs the biometric key \( R \) and the helper data \( P \).

2. The login phase:

(1) User \( U_i \) inserts his smart card into reader and inputs \( ID_i \) and \( PW_i \). And then, \( U_i \) imprints \( B_i^* \) at specific device with fuzzy extractor and computes \( Rep(B_i^*, P) \rightarrow R' \).

(2) Then, it will check if \( h(R) = h(R') \). If it is equal, then the smart card will compute \( S_2' \) and \( S_1 \), as it is shown in Fig 3. Otherwise, if the checking fails, then the smart card would stop the session.

(3) The smart card will generate a random number \( N_1 \) and compute \( M_1 \) and \( a \) (shown in details in Fig 3).

(4) Lastly, \( U_i \) sends as a message the computed values \{\( ID_i, M_1, a\)\} to the remote server \( S \) through a public channel.

3. The authentication and key agreement phase:

\( S \) receives the users request message and then, \( S \) and \( U_i \) start to perform the steps to authenticate each other and establish session key using key agreement step. Fig 3 describes the detail steps of this phase.

4. The password and biometrics update phase:
The idea of this phase is that when \( U_i \) wants to change users \( ID_i \) and \( PW_i \) with a new password \( PW'_i \) new, the user can easily change users password. And also, \( U_i \) can be finished the password and biometrics update phase without assistance of the server \( S \). For further details refer to the original paper[2].
5 Conclusion

In this paper, we show that it is possible to generate secret keys from biometric data and other noisy data, in particular for the purposes of authentication. Roughly speaking, two primitives and their constructions are introduced in our work.

References
