Unlinkable Minutiae-Based Fuzzy Vault for Multiple Fingerprints

Benjamin Tams

Institute for Mathematical Stochastics
University of Goettingen
Goldschmidtstr. 7
D-37077, Goettingen
btams@math.uni-goettingen.de

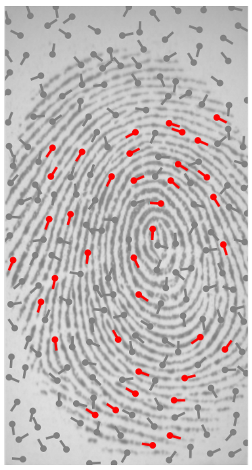
April 14, 2015

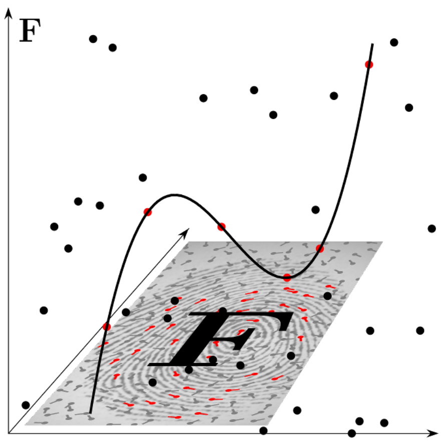
Abstract

The *fuzzy vault scheme* is a cryptographic primitive being considered for storing fingerprint minutiae protected. A well-known problem of the fuzzy vault scheme is its vulnerability against *correlation attack*-based cross-matching thereby conflicting with the *unlinkability requirement* and *irreversibility requirement* of effective biometric information protection. Yet, it has been demonstrated that in principle a minutiae-based fuzzy vault can be secured against the correlation attack by passing the to-be-protected minutiae through a quantization scheme. Unfortunately, single fingerprints seem not to be capable of providing an acceptable security level against offline attacks. To overcome the aforementioned security issues, this paper shows how an implementation for multiple fingerprints can be derived on base of the implementation for single finger thereby making use of a *Guruswami-Sudan algorithm*-based decoder for verification. The implementation, of which public C++ source code can be downloaded, is evaluated for single and various multi-finger settings using the MCYT-Fingerprint-100 database and provides security enhancing features such as the possibility of combination with password and a slow-down mechanism.

Keywords

fingerprint minutiae, fuzzy vault scheme, implementation, multi-finger, unlinkability


(a)



(b)

Figure 1: (a) Genuine (red) and chaff minutiae (grey); (b) each minutia is encoded on a vault point's abscissa where its ordinate binds the minutia to the secret polynomial

1 Introduction

At a glance switching from password to biometric-based authentication schemes can solve problems such as key management in which strong and secure passwords may be required to be stored on a chip card which however could possibly be stolen. Similar to passwords biometric information must be stored in a protected form to prevent them from being lost on data theft. This introduces new challenges due to the biometries' non-deterministic nature. Properties for effective biometric information protection, by means of *renewable biometric references*, are requested by international standards [1]. There exist several biometric modalities such as *iris*, *face*, *palm*, and *fingerprints* (see [2]) where each is related with individual challenges; these must be accounted when implementing extractors for renewable biometric reference for a specific modality or a fusion thereof. In this article, we focus on fingerprints [3] and the generation of renewable biometric references from them with a *fuzzy vault scheme* [4, 5].

1.1 Minutiae-Based Fuzzy Vault

In 2003, Clancy *et al.* [6] analysed the eligibility of the fuzzy vault scheme to protect fingerprint minutiae which resulted in a series of minutiae-based fuzzy vault implementations [7–11]. Basically, their functioning is as follows:

Enrolment

At most $t_{\max}$ minutiae are encoded as a subset $\mathbf{A}$ of a fixed finite field $\mathbf{F}$; then, a polynomial $f \in \mathbf{F}[X]$ of degree smaller than k is chosen at random and the set of *genuine pairs*/*genuine set*, $\mathbf{G} = \{\ (a, f(a)) \mid a \in \mathbf{A}\ \}$ is built; note that, if $|\mathbf{G}| \geq k$, the genuine pairs encode the minutiae as well as the secret polynomial

f. Next, a large set of *chaff pairs/chaff set*, $\mathbf{C} = \{\ (x, y)\ \}$ is generated such that the values of x look like an encoding of a genuine minutiae and the values of y are such that $y \neq f(x)$. In this way the vault $\mathbf{V} = \mathbf{G} \cup \mathbf{C}$ is built hiding the genuine pairs by dispersing them within the chaff pairs (see Figure 1 for a visualization of this process).

We shall note here that it may be necessary to store a cryptographic hash value $\mathrm{SHA}(f)$ along with $\mathbf{V}$ such that the pair $(\mathbf{V}, \mathrm{SHA}(f))$ builds the *vault record* as a candidate for a renewable biometric reference. In fact, in [6] no such hash value is employed but a mechanism for verifying the correctness of f is used in which it is assumed that any other polynomial hardly interpolates a sufficient number of vault pairs. Furthermore, in [8–11] a 16-bit *cyclic redundancy check code* is attached to the correct polynomial such that its correctness can be verified. We stress that a cryptographic hash value, as for example also used in [12, 13], can serve as a mechanism for verification, too, and may even be considered advantageous in view of *blended substitution attacks* the details of which we refer to [14].

Verification

Given $(\mathbf{V}, \mathrm{SHA}(f))$, query minutiae are used to identify those vault pairs from $\mathbf{V}$ of which *vault minutiae*, encoded on the abscissa, well agree with the query minutiae; thereby the *unlocking pairs/unlocking set* $\mathbf{U}$ is established. If we assume that the minutiae protected by the vault and the query minutiae stem from the same finger, then the unlocking set may contain a significant proportion of genuine pairs laying on the graph of the secret polynomial, *i.e.* (x, y) where $f(x) = y$. Certain Reed-Solomon decoders can then be used to recover f the correctness of which can be verified using the hash $\mathrm{SHA}(f)$.

Pre-Alignment

On genuine verification, the query minutiae must be aligned such that they well agree with the genuine vault minutiae. In a minutiae-based fuzzy vault [7, 9–11, 15] a common approach is to support this *pre-alignment step* using auxiliary alignment data publicly stored along with the records. It is important to note that public unprotected data does leak information about the protected fingerprint which could be exploited by an intruder attempting to forge the vault records in an attack. Attacks are discussed in the following in which we ignore (for simplicity) that an implementation has to provide a mechanism for automatic pre-alignment; but, the reader should be aware of the fact that security may be lower in presences of auxiliary alignment data.

Security

Brute-Force Security An intruder who has intercepted a vault $(\mathbf{V}, \mathrm{SHA}(f))$ can try to guess k vault pairs and hope that they are genuine; if they are, the interpolation polynomial f^* will be equals to the secret polynomial f the correctness of which can be verified using $\mathrm{SHA}(f)$. It is important to note that recovery of f is equivalent to recovery of the genuine minutiae set protected by the vault. The difficulty that a guess of k vault pairs yields to the correct polynomial is equals to

$$\binom{n}{k}\binom{t}{k}^{-1} \tag{1}$$

where $t = |\mathbf{G}|$ denotes the number of genuine pairs. This yields a notion of *brute-force security* [16] being often used as the mere measure to assess security in a fuzzy vault to fingerprint; this measure, however, tends to be quite low. For example, in the implementation by Nandakumar, Jain, and Pankanti [10] for the parameter configuration $(n, t_{\max}, k) = (224, 24, 11)$ at a genuine acceptance rate of 86% brute-force security evaluates as 2^{39}.

False-Accept Security It is important to note, that brute-force security tends to significantly overestimate the effective security of a minutia-based fuzzy vault system. A more realistic measure can be derived from the false acceptance rate FAR. An attacker can iteratively simulate impostor verification attempts until he successfully unlocks the vault thereby running a *false-accept attack*; within each simulated attempt, he may succeed with probability equals to FAR. More precisely, if the *average impostor decoding time* IDT is known, then the attacker can expect to unlock the vault with effort

$$\mathrm{IDT} \cdot \log(0.5)/\log(1 - \mathrm{FAR}) \tag{2}$$

yielding the notion of *false-accept security.*

For example, in [10] for $(n, t_{\max}, k) = (224, 24, 9)$ brute-force security results in 2^{31} while the false acceptance rate has been evaluated as FAR $= 0.01\%$. It has furthermore been reported that the unlocking sizes were such that on verification 33 candidate polynomials had to be tested in average. Thus, an intruder can expect to test $33 \cdot \log(0.5)/(1 - 0.01\%) \approx 2^{18}$ candidate polynomials before success. Compared to the brute-force security of 2^{31}, the false-accept attack clearly is the more efficient attack. Even if no false accept has been observed, the false acceptance rate still is non-zero. It may not be clear how to estimate the false acceptance rate except coarse upper bounds (*e.g.* with the *rule of three* [17]). Yet, brute-force security cannot be used for approximating the FAR as emphasized above.

One may argue that a false-accept attack is harder to conduct for an intruder since he has first to establish the attack database containing real fingerprints. However, brute-force attacks may also be accelerated by exploiting the distribution and dependency of the fingerprint features yielding statistical attacks. It is important to note that the false-accept attack does exploit the distribution and dependency of the biometric features and thus yields an upper bounds for the system's overall security.

At a glance it seems reasonable to reduce the false acceptance rate to improve security, *e.g.* by processing other features than mere minutiae [11,12]; but even if we can reduce the false acceptance rate to its half, which would be a tremendous improvement, false-accept security will increase only by a single bit. In view of this fact it seems hard to believe that significant security improvements can be achieved while sticking to a certain biometric modality. We may improve biometric security with the requested significance in combinations with PIN or password or by fusing multiple biometric instances, for example by developing fuzzy vault for multiple fingerprints, the latter being addressed in this work.

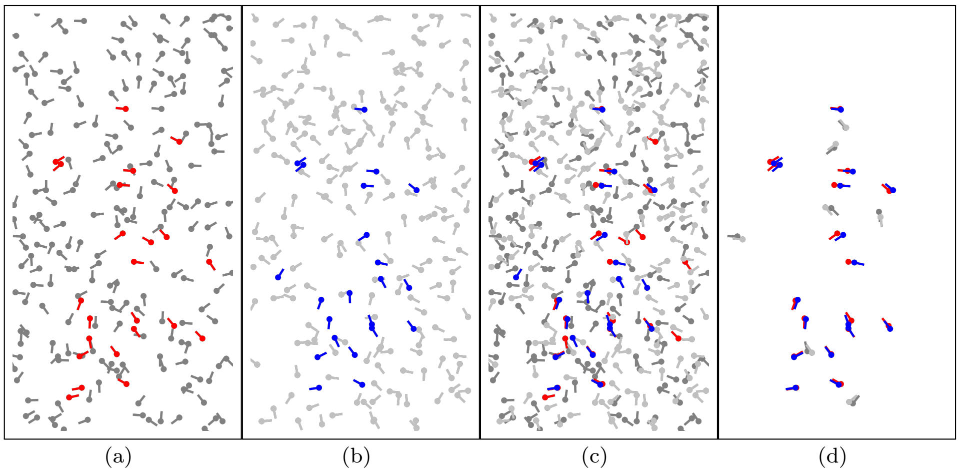


Figure 2: Visualization of the correlation attack: Two vaults (a), (b) with chaff minutiae (grey and light-grey) and genuine minutiae (red and blue). (c), (d) The genuine minutiae have a bias to be in agreement and can be identified to attack unlinkability and irreversibility of both vaults.

Correlation Attack There exists other risks than mere offline attacks. A well-known vulnerability of the fuzzy vault scheme poses the *correlation attack* conflicting with the unlinkability and even irreversibility requirement). Assume an attacker has intercepted two records that have been generated from the same finger. Then, we may observe that genuine minutiae correlate well as compared to chaff minutiae (Figure 2). This can be exploited by an attacker to decide whether two records are *related* or *non-related* which conflicts with the unlinkability requirement. Even worse, since the attacker may be able to identify a significant number of genuine vault pairs via correlation, he may be able to fully break the records. Kholmatov and Yanikoglu [18] demonstrated that in principle approximately 60% related vault records can be broken with the correlation attack.

A simple, yet effective, measure to avoid the correlation attack is to round minutiae to coordinates of a grid (*e.g.* rectangular or hexagonal); all unoccupied grid coordinates are used to encode chaff pairs. Then there is no correlation that can be exploited. It has been demonstrated in [19] that in principle the approach can work: A genuine acceptance rate of $\approx 80\%$ at no observed false accepts has been measured in combination with an automatic method for absolute fingerprint pre-alignment which removes the problem of information leakage from public auxiliary alignment data.

1.2 Contribution

In this paper, on base of the correlation attack-resistant implementation presented in [19] we design an implementation for multiple fingerprint. In particular, to make the verification process practical, we design a decoder based on a *Guruswami-Sudan algorithm* [20]. Furthermore, due to the fact that chaff features are not random, we can base our implementation on the *improved fuzzy vault scheme* by Dodis *et al.* [21, 22] with the positive effect of significantly more compact records. The implementation furthermore features measures for preventing other known record multiplicity attacks. Optionally, the implementation can be encrypted/decrypted with a user password and implements a configurable slow-down mechanism with which the verification time can be artificially increased in order to improve absolute security. We evaluate our implementation using the MCYT-Fingerprint-100 database [23] and provide a security estimation against a certain false-accept attack. For example, in a four-finger setting, our evaluation results in a genuine acceptance rate of 93% at an estimated false-accept security of 2^{65}.

It may be worth noting that our implementation can be downloaded in form of a C++ library called THIMBLE from

`http://www.stochastik.math.uni-goettingen.de/biometrics/thimble`.

The library is supplemented with two executable programs that make use of Digital Persona's FingerJetFX OSE minutiae extractor (`http://digitalpersona.com/fingerjetfx`). With these programs fuzzy vault protected data can be generated from single and multiple fingerprint images; furthermore the programs can be used to run the verification process with single and multiple fingerprint against the protected data previously generated. Using our C++ library THIMBLE, the generated records can be read (using the classes `ProtectedMinutiaeTemplate` and `ProtectedMinutiaeRecord`; also see the documentation of THIMBLE) and then processed allowing the community to run heavy cryptanalyses with our implementation.

1.3 Related Work

An implementation for multiple fingerprint has been presented by Merkle *et al.* in 2011 [13]. However, a performance evaluation for the implementation has not been given and no measures for preventing record multiplicity attacks have been implemented.

Nagar, Nandakumar and Jain in 2012 [24] considered different feature level fusions between the modalities *fingerprint*, *face* and *iris* using the fuzzy vault and *fuzzy commitment scheme* [25]. Even though verification performances were not the focus in [24], only a 75% genuine acceptance rate at a 53 bit security estimate has been achieved which can be significantly outperformed with our implementation.

2 Implementation for Single Fingerprint

Throughout, we assume that fingerprint minutiae are absolutely pre-aligned. An example of such a method has been presented in [19] and an implementation is contained in our open-source software library THIMBLE.

2.1 Minutia Quantization

We pass absolutely pre-aligned minutiae through a quantization scheme. Therefore, we use a hexagonal grid of which coordinates Λ_i are equidistantly spaced by ℓ pixels; the grid is centred in the region in which absolutely pre-aligned minutia can occur (see Figure 3). Given a minutia (α, β, θ) at coordinate (α, β) and angle θ, its quantization is computed by determining the index j of the grid coordinate Λ_j being closest to (α, β), *i.e.*

$$j = \arg\min_i \Lambda_i; \tag{3}$$

the minutia angle is quantized by s different quanta encoded by $j' = \lfloor \theta/(2\pi) \cdot s \rfloor$ such that $j' + s \cdot j$ is the integer encoding the quantization of the minutia (α, β, θ).

We fix a finite field $\mathbf{F}$ where $|\mathbf{F}| \geq r \cdot s$ such that each minutia quantization can be uniquely encoded by a finite field element; throughout, we do not necessarily distinguish between finite field elements and integers encoding them. Finally, we quantize a minutiae template by successively quantizing its minutiae, preferring those attached with a higher quality estimation, and putting the quantizations in a *feature set* $\mathbf{A}$ until each minutia has been processed or $\mathbf{A}$ reaches a bound $t_{\max}$. The quantization process is visualized in Figure 3.

2.2 Enrolment

Assume that we are provided a feature set $\mathbf{A} \subset \mathbf{F}$. Then a secret polynomial $f \in \mathbf{F}[X]$ of degree smaller than k is generated uniformly at random and bound to the feature set $\mathbf{A}$ by computing the polynomial

$$V(X) = f(X) + \prod_{a \in \mathbf{A}} (X - a) \tag{4}$$

as an instance of the improved fuzzy vault scheme [21] (see Figure 4 for a visualization); as motivated in Section 1.1, we store a cryptographic hash $\mathrm{SHA}(f)$ of f along with $V(X)$ such that the pair $(V(X), \mathrm{SHA}(f))$ is considered as the vault record.

If $x \in \mathbf{A}$, then $V(x) = f(x)$ and thus $(x, V(x))$ is a genuine pair; otherwise, if $x \notin \mathbf{A}$, then $V(x) \neq f(x)$ and $(x, V(x))$ is a chaff pair. Hence, we can encode an instance of the original fuzzy vault scheme by a monic polynomial of degree $t = |\mathbf{A}|$.

2.3 Verification

On verification, using a query feature set $\mathbf{B} \subset \mathbf{F}$, briefly called *query set*, the unlocking pairs are computed as $\mathbf{U} = \{ (b, V(b)) \mid b \in \mathbf{B} \}$ which contains exactly $|\mathbf{A} \cap \mathbf{B}|$ genuine pairs. If sufficiently many genuine pairs are contained in $\mathbf{U}$, then the secret polynomial f can be recovered using an algorithm for decoding Reed-Solomon codes.

2.4 Parameter Configuration and Randomized Decoder

In [19], the parameters have been selected on base of a training using the fingerprint images contained in the FVC 2002 DB2-B [26] which have been scanned at a resolution of 569 dots per inch. We adopt the parameters by choosing

- a hexagonal grid of which coordinates are equidistantly spaced by $\ell = 25$ pixels,

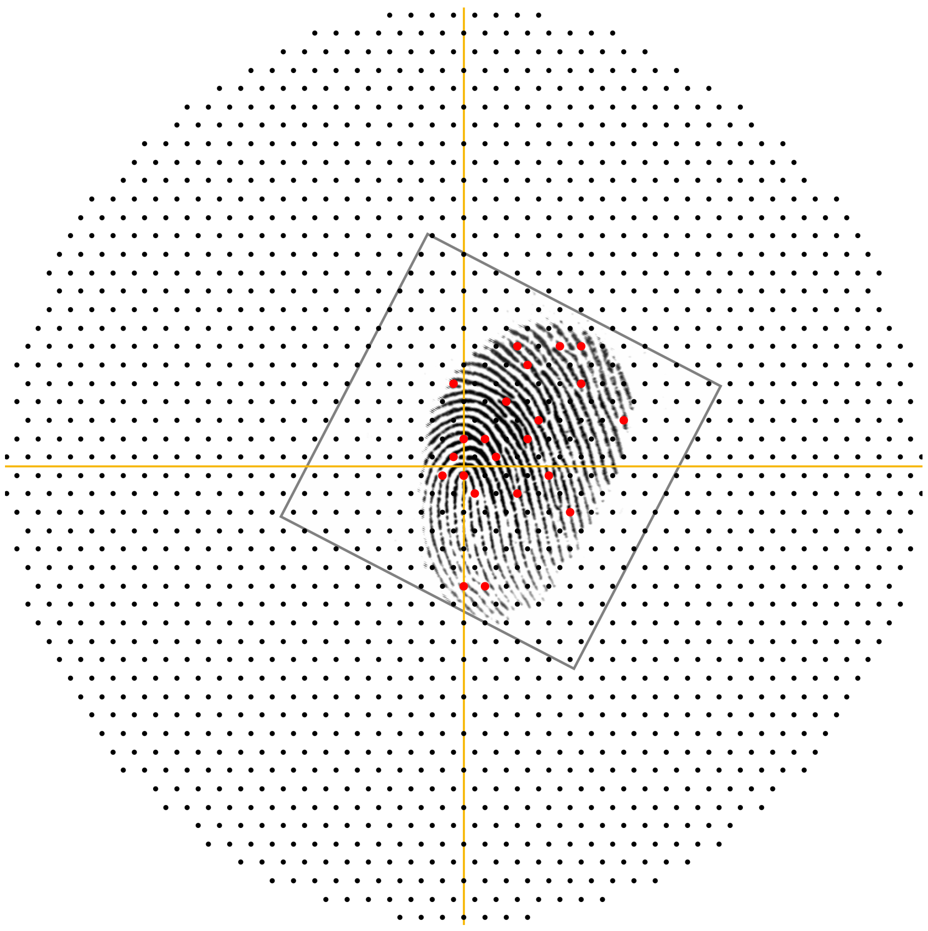


Figure 3: Visualization of the minutiae quantization process: The minutiae of a minutiae template, being absolutely pre-aligned w.r.t. a coordinate system (yellow) derived from a directed reference point estimation, are rounded to the coordinates of a hexagonal grid centred in the region in which the minutiae can occur; those coordinates to which minutiae round are used to encode genuine features (red) while the others are used to encode chaff features (black). Note that under the assumption that a fingerprint's directed reference point estimation can occur at any place within the fingerprint image and can have any angle, the region in which absolutely pre-aligned minutiae can occur forms a circle of radius equals the length of the fingerprint image's diagonal; however, the number of genuine features can only occur within a rectangle of dimension equals the fingerprint image's dimension which must be accounted in a security analysis. Furthermore, note that the minutiae angle are also used for minutiae quantization but their visualizations have been omitted for simplicity.

- $s = 6$ quanta for minutiae angles,
- an upper bound of the genuine feature set size $t_{\max} = 44$, and

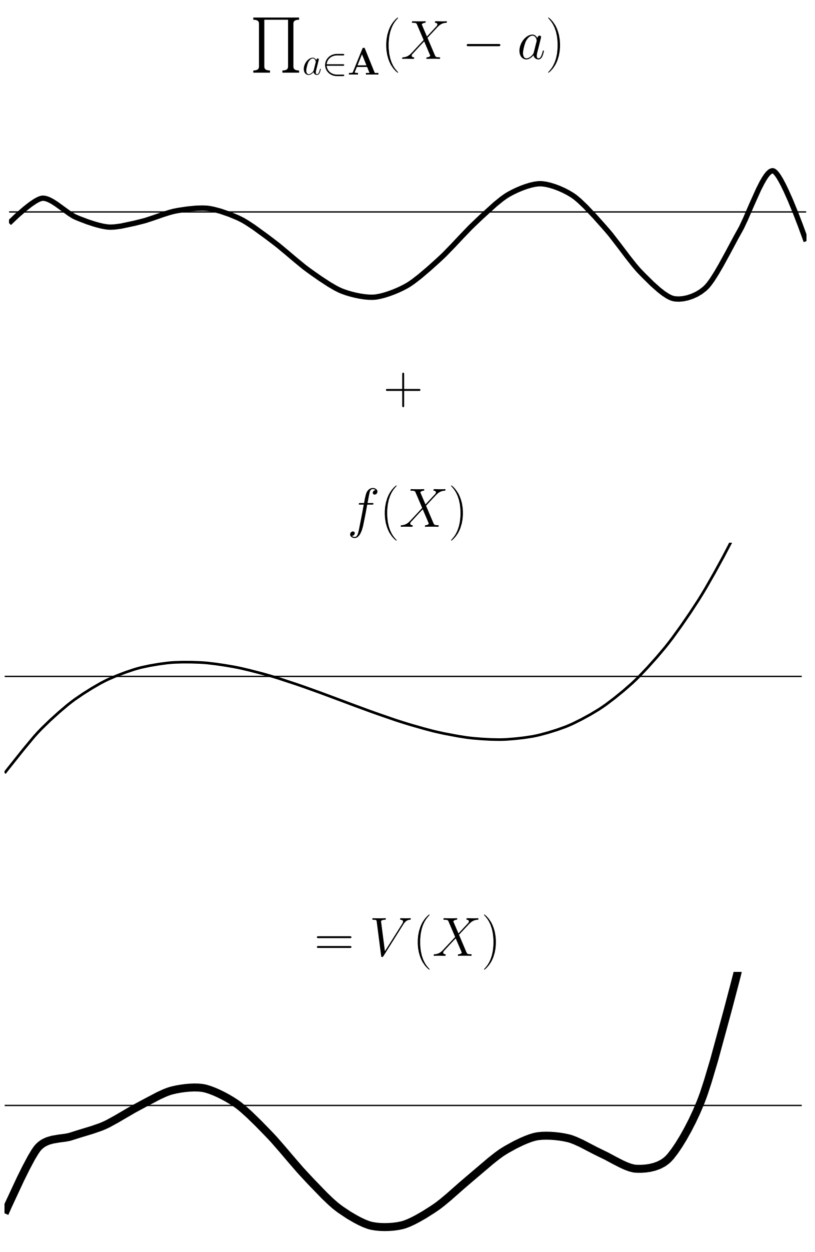


Figure 4: Visualization of the construction of an improved fuzzy vault instance: The feature set $\mathbf{A}$ is encoded by the roots of its characteristic polynomial $\prod_{a\in\mathbf{A}}(X-a)$ which are obscured by addition with a secret polynomial $f(X)$; the result is a polynomial $V(X)$ from which it should be hard to reconstruct the feature set unless a query set of reasonable similarity to $\mathbf{A}$ or $f(X)$ is known.

- varying sizes k of the secret polynomial;

note that the parameter $\ell = 25$ has been re-scaled for fingerprints scanned at 500 dots per inch.

Due to the choice of $t_{\max} = 44$ the unlocking sets occurring on verification can be comparably large which may render complexity of the systematic decoding approach used in nearly all implementations of fuzzy fingerprint vaults [9–12,15] infeasible. Consequently, in [19] a *randomized decoder* has been used in which at most $\mathcal{D}$ randomly chosen unlocking pairs are tested. Consequently, if an unlocking set $\mathbf{U}$ of size $u = |\mathbf{U}|$ contains $\omega = |\mathbf{A} \cap \mathbf{B}| \geq k$ genuine pairs, the probability of a successful verification is equals to

$$1 - \left(1 - \binom{\omega}{k} \cdot \binom{u}{k}^{-1}\right)^{\mathcal{D}}. \tag{5}$$

3 Implementation for Multiple Fingerprints

We now come to the description of our implementation for multiple fingerprints.

3.1 Fusion of Multi-Finger Minutiae Records

Our aim is to achieve high security against offline attacks through the fusion of multiple fingerprint minutiae templates while linkage attacks are avoided. There exist different fusion strategies [27] that one might consider to employ and we shall briefly discuss their properties in our context of biometric template protection.

For each minutiae template in the record, we could generate fuzzy vault-protected data using a method for single fingerprint (*e.g.* as outlined in Section 2) and store them separately. On verification, each query fingerprint template is used to unlock its corresponding fuzzy vault record. Depending on the number of positive individual verifications, we may accept the verification attempt thereby following the concept of a *decision-level fusion*. However, it has been argued in [28] that this fusion strategy may neither significantly improve on privacy nor security of biometric data since an attacker still has the possibility of running recovery attacks to each protected record separately. Similarly, in a *score-level fusion*, which can be considered as a generalization of a decision-level fusion, as each template is protected separately, they remain vulnerable to intensive offline attack.

To improve resistance to offline attacks, we may employ a *feature-level fusion*. Therefore, assume that the feature sets $\mathbf{A}_1, ..., \mathbf{A}_N$ have been generated from a user's N different fingers as described in Section 2.1; furthermore, assume that $\mathbf{B}_1, ..., \mathbf{B}_N$ are second acquisitions such that $\mathbf{B}_j$ matches with $\mathbf{A}_j$. To follow the concept of a feature-level fusion we may fuse the records $(\mathbf{A}_1, ..., \mathbf{A}_N)$ and $(\mathbf{B}_1, ..., \mathbf{B}_N)$ into new feature sets $\mathbf{A}$ and $\mathbf{B}$, respectively, such that $|\mathbf{A}_1 \cap \mathbf{B}_1| + ... + |\mathbf{A}_N \cap \mathbf{B}_N| = |\mathbf{A} \cap \mathbf{B}|$. In this way, the enrolment and verification procedure of our existing implementation for single finger (Section 2) can be adopted. There exist multiple equivalent ways to realize such a fusion.

In our implementation we attach the *finger position code* (*i.e.* an integer encoding a right/left thumb, right/left index finger *etc.*) to the elements of the feature sets that together form the fused feature set. More precisely, let $L_1, ..., L_N$ denote the pair-wise distinct position codes (encoded by values from

$0, ..., 9$) of those fingers from which the feature sets $\mathbf{A}_1, ..., \mathbf{A}_N$ have been estimated. For each $j = 0, ..., N$ we attach the code L_j to the elements of $\mathbf{A}_j$. Let $a \in \mathbf{A}_j$ denote a feature encoded as an integer; then $L_j + 10 \cdot a$ may be used as the feature element to which the finger code L_j has been attached. Finally, we may use the union of all these attached feature elements, *i.e.*

$$\mathbf{A} = \{L_j + 10 \cdot a \mid a \in \mathbf{A}_j,\ j = 1, ..., N\}, \tag{6}$$

as the fused feature set. It is important to note, that the size of the field $\mathbf{F}$ must now be at least 10 times larger than the minimal size required for single fingerprints.

3.2 Enrolment

Analogous to the enrolment for single finger (Section 2.2), given a secret polynomial $f \in \mathbf{F}[X]$ of degree smaller than k and a feature set $\mathbf{A}$ encoding a quantized multi-finger minutiae record, the polynomial

$$V(X) = f(X) + \prod_{a \in \mathbf{A}} (X - a) \tag{7}$$

is built. As usual (see sections 1.1 and 2.2) we publish a cryptographic hash $\mathrm{SHA}(f)$ along with $V(X)$ such that $(V(X), \mathrm{SHA}(f))$ serves as the protected template.

3.3 Verification

To decode the private template $(V(X), \mathrm{SHA}(f))$ protecting a multi-finger record using the quantization set $\mathbf{B}$ of a query record we build the unlocking set

$$\mathbf{U} = \{\ (b, V(b)) \mid b \in \mathbf{B}\ \} \tag{8}$$

in the same way as in Section 2.3. Then, a decoding algorithm is applied to the unlocking set $\mathbf{U}$ returning candidates for the secret polynomial f from which the correct one can be identified using $\mathrm{SHA}(f)$. If a polynomial with the correct hash can be recovered on decoding, then we consider the verification attempt as successful and otherwise as unsuccessful.

3.4 Decoder for Multi-Finger

If the randomized decoder (Section 2.4) were adopted for verification, then we may have expect a low verification performance: Assume we can expect that 20 out of 44 minutiae quantizations can be reproduced for two matching absolutely pre-aligned minutiae templates. For $k = 10$ and $\mathcal{D} = 2^{16}$ the randomized decoder (*e.g.* see Section 2.4) will successfully decode with probability $1 - (1 - \binom{20}{10}\binom{44}{10}^{-1}) \approx 99\%$; in a two-finger scenario we may extrapolate the unlocking set to contain $2 \cdot 44 = 88$ pairs of which $2 \cdot 20 = 40$ are laying on a polynomial's graph where the degree of the polynomial is smaller than $k = 20$; then, the randomized decoder will recover the secret polynomial with probability only $1 - (1 - \binom{40}{20}\binom{88}{20}^{-1}) \approx 0.03\%$. In view of this observation we should consider alternative decoding mechanisms.

Guruswami-Sudan Algorithm

An alternative way to tackle the decoding problem on verification is to use a Guruswami-Sudan algorithm [20]. This class of algorithms can potentially recover f from $\mathbf{U}$ in deterministic polynomial time if it contains $\omega > \sqrt{u \cdot (k-1)}$ genuine pairs where $u = |\mathbf{U}|$. Two statements have been given in [5] that led the community to believe Guruswami-Sudan algorithms were not well suited to support verification in a fingerprint-based fuzzy vault.

1. Implementations of a classical Reed-Solomon decoder are in general much more efficient than implementations of the Guruswami-Sudan algorithm.

2. For many of the parameter choices, we are likely to encounter in practice, $(u+k)/2$ is fairly close to $\sqrt{u \cdot (k-1)}$.

The second statement does not apply to our situation. In fact, $(u+k)/2$ can be much larger than $\sqrt{u \cdot (k-1)}$. For example, if $N = 3$, the unlocking sets can be of size up to $u = t_{\max} \cdot N = 132$. Then, if $k = 30$, a classical Reed-Solomon decoder can successfully decode if $\omega \geq 81$; an implementation of the Guruswami-Sudan algorithm, however, requires $\omega \geq 62$ which is significantly smaller.

Regarding the first statement, the original implementation of a Guruswami-Sudan algorithm [20] has a running time of $\mathcal{O}(u^{15})$. Even though improved to $\mathcal{O}(u^{7.752})$ in the meantime [29], this is significantly inferior to an $\mathcal{O}(u \cdot \log^2 u)$ achievable with classical Reed-Solomon decoders [30]. On the other hand, the high complexities are worst case complexities and hold only if we want to tolerate precisely up to $u - \sqrt{u \cdot (k-1)}$ errors in $\mathbf{U}$. If fewer errors suffice to be tolerated, the running times can become feasible. Therefore, it may be useful to briefly consider the two steps of a Guruswami-Sudan algorithm. For further details we refer to [20].

In the first *interpolation step* a non-zero bivariate polynomial $H \in \mathbf{F}[X,Y]$ of $(1, k-1)$-*weighted degree* not too large is computed having the unlocking pairs as roots with a certain algebraic multiplicity $\mu \geq 1$. In the second *factorization step* all (univariate) polynomials $f^* \in \mathbf{F}[X]$ of degree smaller than k are computed with $H(X, f^*(X)) = 0$. One can prove that if $\omega > \sqrt{u \cdot (k-1)}$ and if μ is sufficiently large, then the candidate list $\{f^*\}$ contains the correct polynomial f.

The bottleneck in the algorithm is the interpolation step and improving its efficiency is subject of current research [29, 31, 32] while the factorization step performs comparably well [33]. Overall, the multiplicity parameter μ is a very critical parameter: The higher μ, the more errors the algorithm can tolerate; on the other hand, the higher μ, the higher is its complexity. Specifically, the interpolation step is closely an $\mathcal{O}(u^2 \cdot \mu^4)$ (see [34]). Consequently, if μ is small, *e.g.* $\mu = 1$, the Guruswami-Sudan algorithm is closely an $\mathcal{O}(u^2)$ while requiring $\omega \gtrsim \sqrt{2 \cdot u \cdot (k-1)}$ [35] for successful recovery of f. If μ is sufficiently large, then the algorithm requires the well-known bound $\omega > \sqrt{u \cdot (k-1)}$ to be met in order to successfully recover f; but then the decoding complexity may be unacceptable.

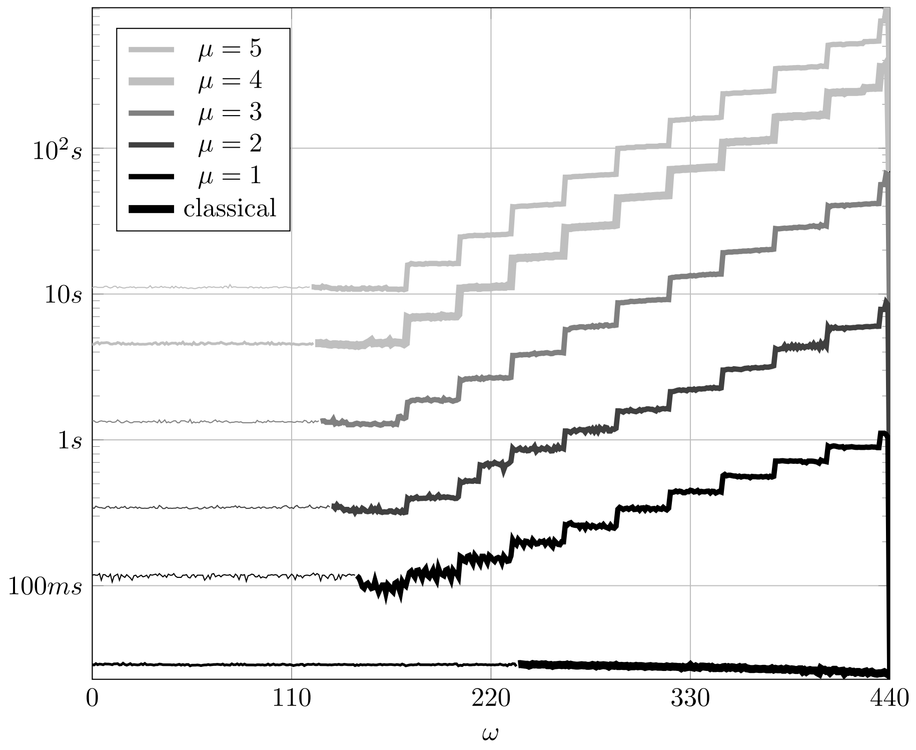


Figure 5: Computational performances of our implementation of a classical Reed-Solomon decoder and the Guruswami-Sudan algorithm for unlocking sets of size $u = 440$ and secret polynomials of length $k = 30$ over a finite field containing 2^{18} elements. The figure plots the measured computer times on a single core of a $3.2 GHz$ desktop for different multiplicities μ versus the number of genuine pairs ω in the unlocking sets. The domains for which the decoding attempts were successful are indicated by a thick plotting strength.

Preliminary Tests

It is not the purpose of the present article to provide comprehensive analyses and evaluations of Guruswami-Sudan algorithms; therefore, we refer to [36]. However, in order to design a decoding mechanism, we describe some experiments that we conducted. Therefore, we have implemented a Guruswami-Sudan algorithm (contained in our public C++ library THIMBLE) in which the factorization step and the interpolation step is performed as in [31] and [33], respectively.

For simplicity, we consider the extremal case of using ten fingers in which the unlocking sets can be of size up to $u = 440$. We consider secret polynomials of length $k = 30$ since this requires the unlocking sets to contain at least 25% genuine pairs (being typical for a fingerprint fuzzy vault) to be decodable with a Guruswami-Sudan algorithm. For each $\omega = 0, \ldots, 440$ we built an unlocking set $\mathbf{U} \subset \mathbf{F} \times \mathbf{F}$ (where $|\mathbf{F}| = 2^{18}$) of size $u = 440$ uniformly at random such that exactly ω unlocking pairs laid on the graph of a common polynomial f of degree smaller than $k = 30$. We applied our Guruswami-Sudan algorithm implementation with multiplicities $\mu = 1, \ldots, 5$ to $\mathbf{U}$ as input and determined whether the algorithm successfully discovered the correct polynomial; furthermore, we measured the times consumed by the decoding attempts that we have performed. Additionally, we applied an own implementation of a classical Reed-Solomon decoder [30] (also contained THIMBLE). The results of our tests are visualized in Figure 5. With a multiplicity of $\mu = 1, 2, 3, 4, 5$ it is possible to decode $\mathbf{U}$ if it contains $\omega \geq 146,\ 132,\ 126,\ 123,\ 121$ genuine pairs, respectively, while for a sufficiently large multiplicity the unlocking set must contain at least $\omega = 113$ genuine pairs.

Not surprisingly for increasing multiplicity μ, the required time for a decoding attempt increases significantly. It is however interesting that the time required also heavily depends on the number of genuine pairs ω contained in $\mathbf{U}$. In view of these observations it seems reasonable to use the following hierarchic decoding mechanism.

Proposed Mechanism

Given the unlocking set $\mathbf{U}$, a classical Reed-Solomon decoder is applied to recover the correct polynomial; if unsuccessful, a Guruswami-Sudan algorithm with increasing multiplicity $\mu = 1, ..., \mu_{\max}$ is applied until the correct polynomial is found or $\mu = \mu_{\max}$ has been reached.

During our experiments (Section 4), we applied the decoding mechanism for $\mu_{\max} = 3$ which results in the affordable worst case decoding time measured as less than $2s$. Note that the average decoding time can be much smaller on genuine verification.

4 Experiments

All experiments were performed on a single core of a $3.2GHz$ desktop computer with sufficient RAM. We evaluated our minutiae-based fuzzy vault implementation for multiple fingerprint using the optical fingerprint scans contained in the MCYT-Fingerprint-100 database [23]. The dataset contains fingerprint samples of 100 different persons where each person provided 12 acquisitions for each of their 10 fingers; the images were scanned at a resolution of 500 dots per inch and have a dimension of 256×400 pixels.

For each fingerprint we used Digital Persona's FingerJetFX OSE extractor to pre-compute the fingerprint minutiae database. Furthermore, from all fingerprints their directed reference points have been estimated using an implementation of the method described in [19]; for all but 0.55% of the fingerprints a valid directed reference point could be successfully estimated where the average time for an estimation attempt was measured as $\approx 629ms$. During our experiments, we represented the minutiae templates w.r.t. the coordinate systems derived from the directed reference point estimations to obtain absolutely pre-aligned minutiae templates.

We evaluated the verification performance for $N = 2$, $N = 3$, and $N = 4$ and various $k = 7, ..., 30$. Requiring the presence of not more than four fingers

Table 1: Evaluation of operational performance with our minutiae-based fuzzy vault implementation for $N = 1$ and $N = 2$; for $N = 1$ the right thumbs were considered and the decoding step was implemented using the randomized decoder (see Section 2.4); for $N = 2$ right index fingers and right middle fingers were fused and the Guruswami-Sudan algorithm-based decoder (Section 3.4) has been employed.

	$N = 1$ (randomized decoder)				$N = 2$			
k	GAR	FAR	GDT	IDT	GAR	FAR	GDT	IDT
7	89.54%	2.76%	$29ms$	$219ms$	97.21%	0.16%	$13ms$	$201ms$
8	85.26%	0.67%	$51ms$	$280ms$	96.53%	0%	$13ms$	$189ms$
9	80.76%	0.11%	$80ms$	$334ms$	95.83%	0%	$13ms$	$175ms$
10	74.90%	0%	$121ms$	$396ms$	95.30%	0%	$15ms$	$171ms$
11	68.34%	0%	$168ms$	$455ms$	94.53%	0%	$14ms$	$144ms$
12	62.08%	0%	$235ms$	$534ms$	93.73%	0%	$16ms$	$148ms$
13	55.23%	0%	$308ms$	$606ms$	93.06%	0%	$15ms$	$126ms$
14	48.73%	0%	$411ms$	$712ms$	92.36%	0%	$17ms$	$124ms$
15	42.72%	0%	$505ms$	$796ms$	90.86%	0%	$18ms$	$118ms$
16	36.15%	0%	$623ms$	$895ms$	89.68%	0%	$19ms$	$114ms$
17	30.63%	0%	$757ms$	$1.01s$	88.33%	0%	$20ms$	$108ms$
18	25.46%	0%	$862ms$	$1.09s$	86.85%	0%	$21ms$	$97ms$
19	20.81%	0%	$997ms$	$1.20s$	84.14%	0%	$23ms$	$95ms$
20	16.67%	0%	$1.14s$	$1.31s$	82.23%	0%	$25ms$	$90ms$
21	13.29%	0%	$1.30s$	$1.45s$	79.79%	0%	$27ms$	$94ms$
22	10.63%	0%	$1.43s$	$1.56s$	77.21%	0%	$30ms$	$88ms$
23	8.18%	0%	$1.60s$	$1.71s$	74.27%	0%	$33ms$	$92ms$
24	5.98%	0%	$1.73s$	$1.82s$	70.76%	0%	$36ms$	$88ms$
25	4.60%	0%	$1.89s$	$1.96s$	66.94%	0%	$39ms$	$93ms$
26	3.31%	0%	$2.05s$	$2.11s$	63.36%	0%	$40ms$	$83ms$
27	2.39%	0%	$2.24s$	$2.23s$	59.04%	0%	$42ms$	$80ms$
28	1.73%	0%	$2.41s$	$2.39s$	56.43%	0%	$44ms$	$77ms$
29	1.12%	0%	$2.59s$	$2.51s$	52.53%	0%	$48ms$	$82ms$
30	0.85%	0%	$2.80s$	$2.71s$	48.82%	0%	$49ms$	$82ms$

from the system users has the advantage that the verification process can be implemented with only one acquisition.

The protected minutiae records were generated using the method described in Section 2.1 and Section 3.1 for a fixed finite field $\mathbf{F}$ with $|\mathbf{F}| = 2^{18}$. Whenever a minutiae record's quantization contained less than k elements we counted this observation as a *failure to enrolment*. However, for none $N = 2, 3, 4$ and $k = 7, ..., 30$ we observed a failure to enrolment which corresponds to a *failure to enrolment rate* measured as 0%.

For each (N, k) we measured the genuine acceptance rate GAR by the following adoption of the FVC protocol [37]. We used each individuals jth scans $(j = 0, ..., 10)$ to generate a protected minutiae record using our implementation. The remaining scans $(j' = j + 1, ..., 11)$ were used to simulate a total of $11 \cdot 12/2 = 66$ genuine verification attempts per person. Consequently, we simulated up to 6600 genuine verification attempts for each (N, k).

To simulate impostor verification attempts, in order to estimate the operational false acceptance rate FAR, for each person (labelled with index $j = 0, ..., 98$) we generated a protected minutiae record using his first scans. The remaining persons' $(j' = j+1, ..., 99)$ first scans were used to simulate an impostor verification attempt. This protocol (again adopted from the FVC protocol) allowed us to simulate a total of up to 4950 impostor verification attempts.

Table 2: Evaluation of operational performance with our minutiae-based fuzzy vault implementation for $N = 3$ and $N = 4$ for which {*right index finger, right middle finger, right ring finger*} and {*right index finger, right middle finger,right ring finger, right little finger*} were fused, respectively, and the Guruswami-Sudan algorithm-based decoder (Section 3.4) has been employed.

	$N = 3$				$N = 4$			
k	GAR	FAR	GDT	IDT	GAR	FAR	GDT	IDT
7	99.21%	1.72%	15ms	357ms	99.53%	6.89%	25ms	477ms
8	99.09%	0.73%	13ms	312ms	99.39%	3.21%	23ms	430ms
9	98.89%	0.20%	14ms	293ms	99.23%	1.31%	23ms	422ms
10	98.65%	0.02%	14ms	273ms	99.06%	0.71%	21ms	374ms
11	98.33%	0.02%	15ms	259ms	98.94%	0.32%	21ms	351ms
12	98.15%	0%	15ms	239ms	98.74%	0.18%	21ms	336ms
13	97.85%	0%	15ms	239ms	98.56%	0.08%	20ms	303ms
14	97.38%	0%	16ms	220ms	98.50%	0%	20ms	293ms
15	96.89%	0%	17ms	194ms	98.29%	0%	20ms	291ms
16	96.47%	0%	17ms	187ms	98.12%	0%	21ms	290ms
17	95.91%	0%	18ms	182ms	97.83%	0%	21ms	261ms
18	95.33%	0%	20ms	189ms	97.47%	0%	22ms	262ms
19	94.41%	0%	21ms	177ms	97.05%	0%	20ms	220ms
20	93.73%	0%	22ms	173ms	96.73%	0%	22ms	231ms
21	92.77%	0%	23ms	153ms	96.47%	0%	23ms	211ms
22	92.08%	0%	25ms	164ms	96.17%	0%	25ms	231ms
23	91.41%	0%	26ms	150ms	95.58%	0%	26ms	212ms
24	90.44%	0%	25ms	136ms	95.05%	0%	26ms	203ms
25	88.73%	0%	27ms	128ms	94.52%	0%	28ms	215ms
26	87.38%	0%	28ms	129ms	94.12%	0%	31ms	200ms
27	86.18%	0%	29ms	128ms	93.41%	0%	28ms	187ms
28	84.85%	0%	31ms	125ms	92.42%	0%	34ms	202ms
29	83.70%	0%	33ms	125ms	91.62%	0%	35ms	185ms
30	82.00%	0%	36ms	121ms	90.86%	0%	35ms	172ms

In addition to an evaluation of our implementation for multiple fingerprints, we also evaluated our implementation for single fingerprints $N = 1$ following the same protocol. It is important to note that for $N = 1$ we used the randomized decoder [19] with $\mathcal{D} = 2^{16}$ decoding iterations [19] (also see Section 2.4). Note that for $N = 1$ we measured a 2% failure to enrolment rate for all $k = 7, ..., 26$; for $k = 30$, the rate was measured as 5%.

We also kept track of the *average genuine decoding time* GDT and *average impostor decoding time* IDT; it is important to note that these times do not include the times needed to estimate absolutely pre-aligned minutiae templates from the fingerprints; therefore, the time $N \cdot 629ms$ should be added to the GDT and IDT to obtain the times required in operational mode.

The result of our experimental evaluations are listed in Table 1 and Table 2.

5 Security

In this section, we discuss the difficulty of an intruder to conduct an offline attack against protected minutiae records generated by our implementation. Therefore, one could estimate the brute-force security from Equation (1): For example, for $N = 4$ within $N \cdot r' \cdot s = 4 \cdot 200 \cdot 6 = 4800$ chaff features up to $N \cdot t_{\max} = 4 \cdot 44 = 176$ genuine features are hidden; note that $r' = 200$ is the number of hexagonal grid coordinates equidistantly spaced by $\ell = 25$ pixels that fit in an image of dimension 256×400 (*c.f.* the grey frame visualized in Figure 3) while $r = 1381$ is the number of grid coordinates for the region in which absolutely pre-aligned minutiae can occur. For $k = 12$, the expression $\binom{r' \cdot s}{k} \cdot \binom{N \cdot t_{\max}}{k}^{-1}$ for computing the brute-force security evaluates to approximately 2^{63}. It is important, however, that for $(N, k) = (4, 13)$ our performance evaluation indicates a false acceptance rate of 0.08% which clearly contradicts a security of 2^{63} suggested by the brute-force attack. In view of the fact, that brute-force security tends to drastically overestimate any realistic security notion, in the following we focus on the more realistic security notion suggested by false-accept attacks.

5.1 False-Accept Security Estimation

In this section, we discuss the possibility of an intruder to break a record generated by our implementation through a false-accept attack. The probably most intuitive approach is to iterate through a large database containing real absolutely pre-aligned minutiae records with which he successively simulates impostor verification attempts. This may, however, not result in sharp estimates: According to Table 1 and Table 2 no false accepts have been observed for $N = 1, 2, 3, 4$ where $k \geq 13$. The rule of three [17] suggests that with confidence 95% the false acceptance rates will be smaller than $3/4950 \approx 0.06\%$ which is way much too coarse for a satisfactory estimation of false-accept security. Yet, the false acceptance rate may be much smaller, especially for $k \gg 12$. In [19] the following heuristic method for estimating an upper false-accept security bound has been introduced and applied for a single fingerprint implementation.

Assume that an attacker has intercepted a vault record $(V(X), \mathrm{SHA}(f))$ generated by our implementation protecting the feature set $\mathbf{A}$. He may then use the jth query set $\mathbf{B}^{(j)}$ from his established attack database containing real fingerprint records to build the unlocking set $\mathbf{U}^{(j)}$ of size $u_j = |\mathbf{U}^{(j)}|$ which contains exactly $\omega_j = |\mathbf{A} \cap \mathbf{B}^{(j)}|$ genuine vault pairs. The attacker has the possibility to run the randomized decoder (Section 2.4) with $\mathcal{D}$ iterations. The probability that he will recover the correct polynomial from $\mathbf{U}$ is equals to

$$p_j = \begin{cases} 0 & \text{, if } \omega < k \\ 1 - \left(1 - \binom{\omega_j}{k}\binom{u_j}{k}^{-1}\right)^{\mathcal{D}} & \text{, if } \omega \geq k \end{cases}. \tag{9}$$

The expected value for p_j may be considered as the false acceptance rate achievable with the randomized decoder. It has been proven in [19] that the cost for running the above attack becomes minimal for $\mathcal{D} = 1$: Although the value for p_j decreases, less iteration steps have to be performed. Consequently, against this specific attack we can estimate the difficulty for the attacker much more sharper than possible by the rule of three [17]. It is important to note that the expected value for p_j is not equals the false acceptance rate of the implementation; rather, its inverse is the difficulty for running the above false-accept attack which implicitly accounts for decoding complexity.

During evaluation of the false acceptance rates (Section 4), we kept track of $M = 4950$ observations for p_j. This allowed us to estimate the difficulty of

Table 3: Estimations of the difficulty against a false-accept attack scenario; the genuine acceptances rates, of which more precise values can be found in Table 1 and Table 2, are listed for the reader's convenience as well.

	$N=1$		$N=2$		$N=3$		$N=4$	
k	GAR	security	GAR	security	GAR	security	GAR	security
7	90%	2^{20}	97%	2^{19}	99%	2^{18}	100%	2^{18}
8	85%	2^{23}	97%	2^{21}	99%	2^{21}	99%	2^{20}
9	81%	2^{26}	96%	2^{23}	99%	2^{23}	99%	2^{23}
10	75%	2^{29}	95%	2^{26}	99%	2^{25}	99%	2^{25}
11	68%	2^{32}	95%	2^{28}	98%	2^{28}	99%	2^{27}
12	62%	2^{36}	94%	2^{31}	98%	2^{30}	99%	2^{30}
13	56%	2^{40}	93%	2^{33}	98%	2^{33}	99%	2^{32}
14	49%	2^{44}	92%	2^{36}	97%	2^{35}	99%	2^{34}
15	43%	2^{49}	91%	2^{39}	97%	2^{37}	98%	2^{36}
16	36%	–	90%	2^{41}	96%	2^{40}	98%	2^{39}
17	31%	–	88%	2^{44}	96%	2^{42}	98%	2^{41}
18	25%	–	87%	2^{47}	95%	2^{45}	97%	2^{43}
19	21%	–	84%	2^{50}	94%	2^{47}	97%	2^{46}
20	17%	–	82%	2^{54}	94%	2^{49}	97%	2^{48}
21	13%	–	80%	2^{57}	93%	2^{52}	96%	2^{50}
22	11%	–	77%	2^{61}	92%	2^{54}	96%	2^{53}
23	8%	–	74%	2^{66}	91%	2^{57}	96%	2^{55}
24	6%	–	71%	2^{70}	90%	2^{59}	95%	2^{57}
25	5%	–	67%	2^{75}	89%	2^{62}	95%	2^{60}
26	3%	–	63%	2^{80}	87%	2^{65}	94%	2^{62}
27	2%	–	59%	2^{86}	86%	2^{68}	93%	2^{65}
28	2%	–	56%	–	85%	2^{70}	92%	2^{67}
29	1%	–	53%	–	84%	2^{73}	92%	2^{70}
30	1%	–	49%	–	82%	2^{76}	91%	2^{73}

performing the above attack as

$$\frac{1}{M}\sum_{j=1}^{M} p_j. \tag{10}$$

The result of the suggested security estimations are listed in Table 3. It is important to note that eventually for sufficiently large k we encountered no $p_j \neq 0$ such that our estimate becomes zero; for this case it is not possible to report a realistic estimation and the corresponding entries in Table 3 are left empty.

Important Remark

Our estimates of false-accept security look promising indeed (see Figure 6). For example, at $(N,k)=(4,27)$ we achieve an operational genuine acceptance rate of 93% at a false-accept security estimated as 2^{65}. However, a few words of caution must be given. Even though the size in Table 5.1 are more realistic than suggested by a brute-force attack notion, they remain heuristic estimates for upper bounds for the difficulty in breaking our implementation. It is important to note that any analyses against a specific attack cannot provide more than mere upper bounds. It may well be that smaller upper bounds can be found against the above attack or that an attacker can run an improved attack. In

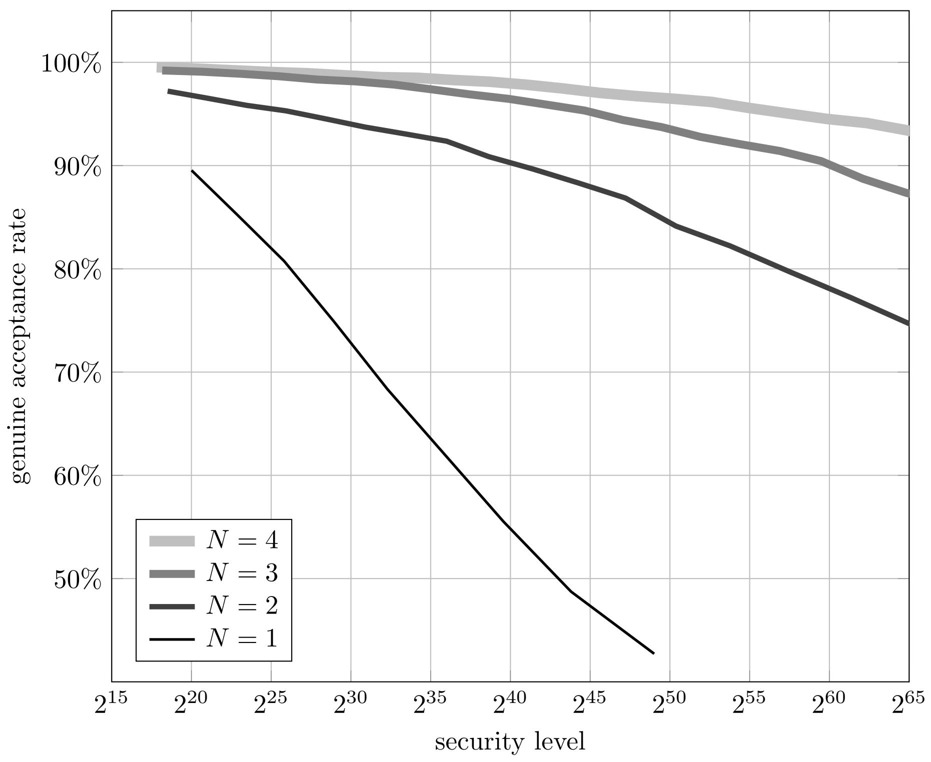


Figure 6: Genuine acceptance rates plotted versus the estimated security levels for a different number of fingerprints N; note that for $N = 1$ the randomized decoder has been used (Section 2.4).

order to ease analyses against any further attack, we published C++ source code of our implementation as part of THIMBLE.

5.2 Record Multiplicity Attacks

One of the most serious problem with a traditional minutiae-based fuzzy vault is its vulnerability against the correlation attack [18]. For our implementation there exists no correlation between genuine vault features which could be exploited by an attacker to gain advantage in linking or breaking two (or more) related vault records. We stress that this is an obvious property of our implementation: Conducting experiments with this specific attack will definitely result in a 0% advantage. In the following we set our focus to a record multiplicity attack that is of more relevance for our implementation.

Table 4: Rates at which two related vault records can be cross-matched and broken with the attack from [38,39] for the configurations from our experiments (Section 4). The attack was never successful for non-related vault records. Furthermore, if the feature sets have been passed through a random record-specific but public permutation process, the attack was never successful, too—neither for related nor non-related records.

	$N=1$		$N=2$		$N=3$		$N=4$	
k	related linkage rate	related recovery rate	related linkage rate	related recovery rate	related linkage rate	related recovery rate	related linkage rate	related recovery rate
7	35.78%	35.54%	76.28%	76.28%	78.63%	78.63%	78.41%	78.41%
8	35.49%	35.06%	75.28%	75.27%	78.10%	78.10%	77.76%	77.76%
9	30.43%	29.68%	72.62%	72.59%	76.91%	76.91%	76.34%	76.34%
10	30.03%	28.36%	71.55%	71.50%	76.22%	76.22%	75.67%	75.67%
11	24.98%	21.85%	68.90%	68.79%	74.46%	74.46%	74.53%	74.53%
12	24.67%	19.47%	67.77%	67.36%	73.71%	73.71%	73.78%	73.78%
13	19.90%	11.94%	65.08%	64.49%	71.84%	71.84%	72.28%	72.28%
14	19.72%	8.63%	63.84%	62.69%	70.80%	70.78%	71.66%	71.66%
15	14.89%	0%	60.68%	58.83%	68.78%	68.76%	70.29%	70.29%
16	14.59%	0%	59.17%	56.52%	67.80%	67.77%	69.39%	69.39%
17	10.61%	0%	56.52%	52.72%	65.92%	65.89%	67.92%	67.92%
18	10.39%	0%	55.10%	49.76%	65.18%	65.12%	67.31%	67.31%
19	7.33%	0%	51.70%	44.95%	63.07%	62.95%	65.87%	65.87%
20	7.22%	0%	50.29%	41.74%	61.94%	61.77%	64.97%	64.97%
21	4.73%	0%	47.25%	36.76%	59.86%	59.60%	63.51%	63.51%
22	4.59%	0%	45.71%	33.50%	58.65%	58.23%	62.67%	62.66%
23	2.74%	0%	42.81%	28.70%	56.93%	56.26%	61.05%	61.00%
24	2.68%	0%	41.14%	25.11%	55.88%	54.91%	60.01%	59.96%
25	1.45%	0%	37.89%	19.90%	53.52%	52.04%	58.33%	58.30%
26	1.42%	0%	36.21%	15.87%	52.49%	50.31%	57.22%	57.18%
27	0.66%	0%	32.78%	10.24%	50.23%	47.41%	55.75%	55.66%

In 2013, Blanton and Aliasgari [38] observed that, given two vault records

$$\begin{aligned} W(X) &= f(X) + \prod_{a\in\mathbf{A}}(X-a) \text{ and} \\ W(X) &= g(X) + \prod_{b\in\mathbf{B}}(X-b) \end{aligned} \tag{11}$$

protecting the feature sets $\mathbf{A}$ and $\mathbf{B}$ of size t with $\deg(f), \deg(g) < k$, such that $|\mathbf{A}\cap\mathbf{B}| \geq (t+k)/2$, then the set difference $\mathbf{A}\setminus\mathbf{B}$ and $\mathbf{B}\setminus\mathbf{A}$ can be recovered explicitly by solving a system of non-linear equations. In particular, if $\mathbf{A}=\mathbf{B}$, an attacker may observe that the upper $t-k$ coefficients of $V(X)$ and $W(X)$ are equal which is hardly the case for $\mathbf{A}\neq\mathbf{B}$. We can therefore not guarantee that the improved fuzzy vault scheme fulfills the unlinkability requirement. For the general case $|\mathbf{A}\cap\mathbf{B}| \geq (t+k)/2$, Blanton and Aliasgari argued that recovery of the feature sets' differences is computationally hard since solving a system of non-linear equations is NP hard in general. However, Merkle and Tams [39] observed that the extended Euclidean algorithm can be used to efficiently solve the equations established by Blanton and Aliasgari. In particular, if $z_j = w_j\cdot V + v_j\cdot W$ denotes the sequence in the extended Euclidean algorithm applied to V and W where without loss of generality $|\mathbf{A}|\geq|\mathbf{B}|$, then there exists j_0 such that v_{j_0} and w_{j_0} split into linear factors of which roots coincide with $\mathbf{A} \setminus \mathbf{B}$ and $\mathbf{B} \setminus \mathbf{A}$, respectively (more specifically, j_0 is such that $\deg(v_{j_0})$ is minimal where $\deg(v_{j_0}) + k > \deg(z_{j_0})$).

As we observed experimentally during our performance evaluation (see Section 4) the property $|\mathbf{A} \cap \mathbf{B}| \geq (\max\{|\mathbf{A}|, |\mathbf{B}|\} + k)/2)$ is way much too often fulfilled for related records (*related linkage rate*; see Table 4); furthermore and even worse, the discovered differences $\mathbf{A} \setminus \mathbf{B}$ and $\mathbf{B} \setminus \mathbf{A}$ are very often sufficient (*related recovery rate*) to break two related records which conflicts with the irreversibility requirement. On the other side in our experiments we never observed $|\mathbf{A} \cap \mathbf{B}| \geq (\max\{|\mathbf{A}|, |\mathbf{B}|\} + k)/2)$ fulfilled for non-related records. This allows an adversary to attack unlinkability of our implementation and calls for a countermeasure.

Fortunately, the property $|\mathbf{A} \cap \mathbf{B}| \geq (\max\{|\mathbf{A}|, |\mathbf{B}|\} + k)/2$ can be destroyed by passing the feature sets through a random record-specific but public permutation process [19,39]. Let $P, Q : \mathbf{E} \to \mathbf{E}$ be two random permutations between the vault feature space $\mathbf{E} \subset \mathbf{F}$ of size $n = |\mathbf{E}|$. Instead of constructing the vault records to protect the feature sets $\mathbf{A}$ and $\mathbf{B}$ the vaults are generated to protect the shuffled feature sets

$$\begin{aligned} \mathbf{A}' &= P(\mathbf{A}) = \{ \ P(a) \mid a \in \mathbf{A} \ \} \\ \mathbf{B}' &= Q(\mathbf{B}) = \{ \ Q(b) \mid b \in \mathbf{B} \ \}; \end{aligned} \tag{12}$$

note that since the permutations are public the verification process can be easily modified by passing the query features through the same permutation. Since $P, Q : \mathbf{E} \to \mathbf{E}$ are random, the shuffled feature sets $\mathbf{A}'$ and $\mathbf{B}'$ are random, even for related feature sets. Then, assuming without the loss of generality that $|\mathbf{B}'| \leq |\mathbf{A}'|$, the probability that $\mathbf{A}'$ and $\mathbf{B}'$ share at least ω_0 elements is equals to

$$\mathbb{P}(\ |\mathbf{A}' \cap \mathbf{B}'| \geq \omega_0 \ ) = 1 - \frac{\sum_{j=0}^{\omega_0 - 1} \binom{|\mathbf{B}'|}{j} \cdot \binom{n - |\mathbf{B}'|}{|\mathbf{A}'| - j}}{\binom{n}{|\mathbf{A}'|}} \tag{13}$$

which directly follows from the definition of the *hyper-geometric distribution*.

For the configurations considered in this paper the feature sets have a size of $|\mathbf{A}'|, |\mathbf{B}'| \leq t_{\max} \cdot N$ where $t_{\max} = 44$ and $N = 1, ..., 4$. Furthermore, the feature space is of size $n = N \cdot r \cdot s = N \cdot 1381 \cdot 6 = N \cdot 8286$. For $N = 1, ..., 4$, $k = 7, ..., 30$, and $k \leq |\mathbf{B}'| \leq |\mathbf{A}'| \leq N \cdot t_{\max}$ it is easy to verify that the probability computed with Equation (13) never becomes larger than 2^{-72} where $\omega_0 = \lfloor (|\mathbf{A}'| + k)/2 \rfloor$. This is sufficient for an effective and valid countermeasure against record multiplicity attacks. In fact, during our experiments (Section 4), we never observed related or non-related records against which the attack from [39] was successful when the countermeasure was implemented.

Finally, note that it is not necessary to store the full table for the permutation process along with a vault record $(V(X), \mathrm{SHA}(f))$. Instead we may store a compact seed generating the permutation. In our implementation, which is contained in THIMBLE, we exploit the public hash $\mathrm{SHA}(f)$ as the seed generating the record-specific public permutation process of the records. In such a way no additional storage bits are needed to prevent the attack from [39].

5.3 Variant with Password

The estimates for our implementation's resistance against offline attacks look promising. Furthermore, using record-specific (but public) permutation processes, there are currently no efficient and effective record multiplicity attacks known with which an attacker can link (or even break) two related records he might have intercepted. It might however well be that in the process of future research more efficient attacks will be found. We made our implementation public such that this process can be eased and accelerated. If as a result significant vulnerabilities are found, it may be important to consider the possibility of securing the implementation with an additional user-specific secret password/PIN.

For a user password κ let $\mathrm{enc}_\kappa, \mathrm{dec}_\kappa : \{0,1\}^* \to \{0,1\}^*$ be a symmetric encryption and decryption function, respectively. On enrolment, the function enc_κ can be used to encrypt the bits needed to represent the first $t = \deg(V)$ coefficients of the monic record's vault polynomial $V(X)$. On verification, the correct vault polynomial can be recovered using dec_κ which reveals the first t coefficients of $V(X)$ while the leading coefficient is known to be equals to 1; otherwise, if another user password $\kappa' \neq \kappa$ is used to decrypt the vault, then another vault polynomial (indistinguishable from the correct one) is obtained with which the verification will, with overwhelming probability, fail.

It is important to note that some encryption/decryption functions (such as AES [40]) work with a multiple of a fixed block length. In order to fulfil the requirement that falsely decrypted data is indistinguishable from correctly decrypted data, it may be necessary to pad the leading block by random bits which are ignored on decryption. In such a way, additional security provided by the strength of a user password is multiplied with the security provided by the fuzzy vault-based protection of the minutiae templates. For example, at $(N, k) = (4, 20)$ our security estimates as 2^{48} at a GAR = 97% (see Table 3). If a four digit PIN is used to encrypt the vault records, then the security can be increased by approximately 13 bits to $2^{48+13} = 2^{61}$. Assuming that the same PIN can be reproduced on genuine verification, the GAR of 97% will be maintained.

There does already exist an implementation of a minutiae-based fuzzy vault that additionally protects the records with a user password [15] in which however the possibility of using a password goes at the cost of the genuine acceptance rate. Furthermore, no guarantees that falsely decrypted vault records are indistinguishable from correctly decrypted vault records are addressed in [15]. As a consequence in [15] the securities provided by the user password and by the fuzzy vault may not multiply, thereby requiring a certain strength from the user passwords (*e.g.* 64 bits). Our implementation has the advantage that it can be combined with even weak user passwords (*e.g.* easily memorable PINs).

5.4 Slow-Down Functions

Another approach to improve security against offline attacks is to involve measures for artificially forcing the verification to be slowed down. In such a way, on a simulated impostor verification attempt, an attacker has to wait a certain time before he can continue with a next trial. In the following we outline a simple way of realizing such a slow-down mechanism of which an interface is implemented in THIMBLE.

A slow-down mechanism can be obtained by exploiting the possibility of encrypting/decrypting the vault records with a secret key. Assume that we want to artificially slow down the verification process by the factor K. On enrolment, a secret quiz $q = 0, ..., K - 1$ is chosen uniformly at random. The quiz is used to encrypt the vault in a similar manner as outlined above in Section 5.3. Finally, the quiz q is dismissed. On (impostor) verification, the verifier has to iterate through all possible quizzes $q' = 0, ..., K-1$ and with each q' the vault is temporarily decrypted of which only one reveals the correct vault polynomial that possibly leads to a (false) accept.

In this way, the verification process can effectively be slowed down. This enables us to improve the absolute security of our implementation against offline attacks by $\log_2(K)$ bits. On genuine verification, on the other hand, the process is expected to be slowed down by the factor $K/2$. Consequently, the relation between system security and genuine verification time cannot be changed by slow-down measures. Nonetheless, slow-down functions may be a measure to increase absolute security while maintaining relative security in view of increasing computer power enabling faster and faster offline attacks while also enabling faster and faster verification.

6 Discussion

We presented and described a new implementation of a minutiae-based fuzzy vault for multiple fingerprints of which C++ source code is publicly available. Our implementation has been designed to resist known record multiplicity attacks and features several security-enhancing functionalities, *e.g.* combination with user password/PIN and slow-down functions. Our evaluation and security estimates indicate promising verification performances and comparably high resilience against offline attacks — even without a user password/PIN that needs to be kept secret. For example, at a security level estimated as 53 bits we achieve a genuine acceptance rate of 96% whilst in [24] a genuine acceptance rate of 75% has been achieved for the same security level estimation. Yet, pessimism is appropriate as we may have learned from earlier security overestimates: It is definitely possible that attacks will be discovered that significantly perform better than the false-accept attack scenario that we considered in Section 5.1. In order to allow the community to analyse our implementation with their own attacks we published the source code of our implementation. This may accelerate and ease future research.

We have demonstrated that, by fusing multiple biometric instances, it may be possible to achieve high usability at high resistance to known recovery and linkage attacks. We used the fingerprint modality to demonstrate the effectiveness of fusing multiple biometric instances. In view of the presence of other well-studied biometric modalities such as *face*, *iris*, and *signature*, it is legitimate to ask whether our implementation can be modified for other multi-biometric fusions as well to achieve high security and usability. It seems safe to claim that if the multi-biometric templates can be encoded as feature sets of which similarity is reflected by the set difference metric, then it may be possible to reach this goal; it is due to analyses and evaluations of specific well-conceived implementations to verify this. However, some templates of certain biometric modalities may not be suitable for being encoded as feature sets. This can negatively affect verification performance of the scheme and may require other fusions strategies. For an overview of the complexity of this theme, we refer the reader to [24].

Another interesting task for future research addresses the effectiveness of applying a random record-specific but public permutation process in an improved fuzzy vault scheme to prevent record multiplicity attacks (Section 5.2). As of now, no attack from record multiplicity is known with which an intruder can gain significantly more advantage than breaking one of the vaults individually. To support this notion, a proof would be highly desirable or alternatively we might disprove the effectiveness of the countermeasure by formulating an effective and efficient linkage attack exploiting the publicity of the permutation processes. The need for an answer to this question is emphasized by the fact that a similar, yet different, countermeasure in a binary fuzzy commitment scheme [41] makes the scheme vulnerable to another attack from record multiplicity [42,43].

Acknowledgements

The author gratefully acknowledges support of the Felix Bernstein Institute for Mathematical Statistics in the Biosciences and the Volkswagen Foundation. Furthermore, the author would like to thank Christian Rathgeb from the Hochschule Darmstadt for his valuable assistance in performing the experimental part of this paper.

References

[1] ISO/IEC 24745:2011: 'Information technology—security techniques—biometric information protection', 2011

[2] Jain, A.K., Flynn, P., Ross, A.: 'Handbook of biometrics' (Springer, 2007)

[3] Maltoni, D., Maio, D., Jain, A., Prabhakar, S.: 'Handbook of fingerprint recognition' (Springer, 2007, 2nd edn. 2009)

[4] Juels, A., Sudan, M.: 'A fuzzy vault scheme'. Proc. Int. Symp. Inf. Theory, Lausanne, Switzerland, June–July 2002, p. 408

[5] Juels, A., Sudan, M.: 'A fuzzy vault scheme', Des. Codes Cryptography, 2006, 38, (2), pp. 237–257

[6] Clancy, T. C., Kiyavash, N., Lin, D. J.: 'Secure smartcard-based fingerprint authentication'. Proc. ACM SIGMM workshop on Biometrics methods and applications, Berkeley, USA, November 2003, pp. 45–52

[7] Yang, S., Verbaudwhede, I.: 'Automatic secure fingerprint verification system based on fuzzy vault scheme'. Proc. Int. Conf. on Acoustics, Speech and Signal Processing, Philadelphia, USA, March 2005, pp. 609–612

[8] Uludag, U., Pankanti, A., Jain, A. K.: 'Fuzzy vault for fingerprints'. Proc. Int. Conf. on Audio- and Video-Based Biometric Person Authentication, Rye Brook, New York, USA, July 2005, pp. 310–319

[9] Uludag, U., Jain, A. K.: 'Securing fingerprint template: fuzzy vault with helper data'. Proc. Workshop on Privacy Research In Vision, New York, USA, June 2006, pp. 163–169

[10] Nandakumar, K., Jain, A. K., Pankanti, S.: 'Fingerprint-based fuzzy vault: Implementation and performance', IEEE Trans. Inf. Forensics Security, 2007, 2, (4), pp. 744–757

[11] Nagar, A., Nandakumar, K., Jain, A. K.: 'A hybrid biometric cryptosystem for securing fingerprint minutiae templates', Pattern Recogn. Lett., 2010, 31, (8), pp. 733–741

[12] Li, P., Yang, X., Cao, K., Tao, X., Wang, R., Tian, J.: 'An alignment-free fingerprint cryptosystem based on fuzzy vault scheme', J. Netw. Comput. Appl., 2010, 33, (3), pp. 207–220

[13] Merkle, J., Ihmor, H., Korte, U., Niesing, M., Schwaiger, M.: 'Performance of the fuzzy vault for multiple fingerprints'. Proc. BIOSIG, Darmstadt, Germany, September 2011, pp. 57–72

[14] Scheirer, W. J., Boult, T. E.: 'Cracking fuzzy vaults and biometric encryption'. Proc. Biometrics Symp., Baltimore, USA, September 2007, pp. 1–6

[15] Nandakumar, K., Nagar, A., Jain, A.: 'Hardening fingerprint fuzzy vault using password'. Proc. Int. Conf. on Biometrics, Seoul, Korea, August 2007, pp. 927–937

[16] Mihăilescu, P., Munk, A., Tams, B.: 'The fuzzy vault for fingerprints is vulnerable to brute force attack'. Proc. BIOSIG, Darmstadt, Germany, September 2009, pp. 43–54

[17] Hanley, J. A., Lippman-Hand, A.: 'If nothing goes wrong, is everything allright? interpreting zero numerators', Journal of the American Medical Association, 1983, 249, (13), pp. 1743–1745

[18] Kholmatov, A., Yanikoglu, B.: 'Realization of correlation attack against the fuzzy vault scheme'. Proc. SPIE vol. 6819, San Jose, USA , February 2008, pp. 1–7

[19] Tams, B., Mihăilescu, P., Munk, A.: 'Security considerations in minutiae-based fuzzy vaults', IEEE Trans. Inf. Forensics Security, 2015, 10, (5), pp. 985–998

[20] Guruswami, V., Sudan, M.: 'Improved decoding of reed-solomon and algebraic-geometric codes', IEEE Trans. Inf. Theory, 1998, 45, (6), pp. 1757–1767

[21] Dodis, Y., Reyzin, L., Smith, A.: 'Fuzzy extractors: how to generate strong keys from biometrics and other noisy data'. Proc. Int. Conf. Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2004, pp. 523–540

[22] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: 'Fuzzy extractors: how to generate strong keys from biometrics and other noisy data', SIAM J. Comput., 2008, 38, (1), pp. 97–139

[23] Ortega-Garcia, J., Fierrez-Aguilar, J., Simon, D., *et al.*: 'MCYT baseline corpus: a bimodal biometric database', IEE Proc. on Vision, Image and Signal Processing, 2003, 150, (6), pp. 395–401

[24] Nagar, A., Nandakumar, K., Jain, A. K.: 'Multibiometric cryptosystems based on feature-level fusion', IEEE Trans. Inf. Forensics Security, 2012, 7, (1), pp. 255–268

[25] Juels, A., Wattenberg, M.: 'A fuzzy commitment scheme'. Proc. of ACM Conf. on Computer and Communications Security, 1999, Singapore, pp. 28–36

[26] Maio, D., Maltoni, D., Cappelli, R., Wayman, J., Jain, A.: 'FVC2002: Second fingerprint verification competition'. Proc. Int. Conf. on Pattern Recognition, Quebec City, Canada, August 2002, pp. 811–814

[27] Ross, A., Nandakumar, K., Jain, A. K.: 'Handbook of multibiometrics' (Springer, 2006)

[28] Merkle, J., Kevenaar, T., Korte, U.: 'Multi-modal and multi-instance fusion for biometric cryptosystems'. Proc. BIOSIG, Darmstadt, Germany, September 2012, pp. 51–62

[29] Cohn H., Heninger, N.: 'Ideal forms of coppersmith's theorem and guruswami-sudan list decoding'. Proc. Innovations in Computer Science, Bejing, China, January 2011, pp. 298–308

[30] Gao, S.: 'A new algorithm for decoding reed-solomon codes', in Bhargava, V.K., Poor, H.V., Tarokh, V., Yoon, S. (Eds.): 'Communications, Information and Network Security' (Springer, 2002), pp. 55–68

[31] Trifonov, P.: 'Efficient interpolation in the guruswami-sudan algorithm', IEEE Trans. Inf. Theory, 2010, 56, (9), pp. 4341–4349

[32] Alekhnovich, M.: 'Linear diophantine equations over polynomials and soft decoding of reed-solomon codes'. Proc. Symp. on Foundations of Computer Science, Vancouver, Canada, November 2002, pp. 439–448

[33] Roth, R. M., Ruckenstein, G.: 'Efficient decoding of reed-solomon codes beyond half the minimum distance', IEEE Trans. Inf. Theory, 2000, 46, (1), pp. 246–257

[34] 'The guruswami-sudan decoding algorithm for reed-solomon codes', `http://www.ee.caltech.edu/EE/Faculty/rjm/papers/RSD-JPL.pdf`, accessed January 2015

[35] Sudan, M.: 'Decoding of reed solomon codes beyond the error-correction bound', Journal of Complexity, 1997, 13, (1), pp. 180–193

[36] Guruswami, V., Rudra, A.: 'Error correction up to the information-theoretic limit', Commun. ACM, 2009, 52, (3), pp. 87–95

[37] Maio, D., Maltoni, D., Cappelli, R., Wayman, J., Jain, A.: 'FVC2000: fingerprint verfication competition', IEEE Trans. Pattern Anal. Mach. Intell., 2000, 24, (3), pp. 402–412

[38] Blanton, M., Aliasgari, M.: 'Analysis of reusability of secure sketches and fuzzy extractors', IEEE Trans. Inf. Forensics Security, 2013, 8, (9), 1433–1445

[39] Merkle, J., Tams, B.: 'Security of the improved fuzzy vault scheme in the presence of record multiplicity', arXiv:1312.5225, 2013

[40] FIPS PUB 197: 'Announcing the advanced encryption standard (AES)', 2001

[41] Kelkboom, E. J. C., Breebaart, J., Kevenaar, T. A. M., Buhan, I., Veldhuis, R. N.: 'Preventing the decodability attack based cross-matching in a fuzzy commitment scheme', IEEE Trans. Inf. Forensics Security, 2011, 6, (1), pp. 107–121

[42] Tams, B.: 'Decodability attack against the fuzzy commitment scheme with public feature transforms', arXiv:1406.1154, 2014

[43] Simoens, K., Tuyls, P., Preneel, B.: 'Privacy weaknesses in biometric sketches'. Proc. IEEE Symp. on Security and Privacy, Oakland, USA, May 2009, pp. 188–203