Handwriting Biometric Hash Attack: A Genetic Algorithm with User Interaction for Raw Data Reconstruction

Karl Kümmel[1], Claus Vielhauer[1,2], Tobias Scheidat[1,2],
Dirk Franke[2], and Jana Dittmann[2]

[1] Brandenburg University of Applied Sciences, Department of Informatics and Media
Magdeburger Str. 50, 14770 Brandenburg, Germany
[2] University of Magdeburg, Department of Computer Science, Advanced Multimedia and Security Lab
Universitätsplatz 2, 39106 Magdeburg, Germany
{kuemmel,vielhauer,scheidat}@fh-brandenburg.de,
dirk.franke@st.ovgu.de, jana.dittmann@iti.cs.uni-magdeburg.de

Abstract. Biometric Hash algorithms, also called BioHash, are mainly designed to ensure template protection to its biometric raw data. To assure reproducibility, BioHash algorithms provide a certain level of robustness against input variability to ensure high reproduction rates by compensating for intra-class variation of the biometric raw data. This concept can be a potential vulnerability. In this paper, we want to reflect such vulnerability of a specific Biometric Hash algorithm for handwriting, which was introduced in [1], consider and discuss possible attempts to exploit these flaws. We introduce a new reconstruction approach, which exploits this vulnerability; to generate artificial raw data out of a reference BioHash. Motivated by work from Cappelli et al. for fingerprint modality in [6] further studied in [3], where such an artificially generated raw data has the property of producing false positive recognitions, although they may not necessarily be visually similar. Our new approach for handwriting is based on genetic algorithms combined with user interaction in using a design vulnerability of the BioHash with an attack corresponding to cipher-text-only attack with side information as system parameters from BioHash. To show the general validity of our concept, in first experiments we evaluate using 60 raw data sets (5 individuals overall) consisting of two different handwritten semantics (arbitrary Symbol and fixed PIN). Experimental results demonstrate that reconstructed raw data produces an EERreconstr. in the range from 30% to 75%, as compared to non-attacked inter-class EERinter-class of 5% to 10% and handwritten PIN semantic can be better reconstructed than the Symbol semantic using this new technique. The security flaws of the Biometric Hash algorithm are pointed out and possible countermeasures are proposed.

Keywords: Biometric Hashing, Online Handwriting, Vulnerabilities, Reproducibility, Security.

B. De Decker and I. Schaumüller-Bichl (Eds.): CMS 2010, LNCS 6109, pp. 178–190, 2010.

1 Introduction and Motivation

A variety of biometric identification and verification systems based on fingerprints, iris, voice etc. were introduced during the last years. In this paper we discuss the dynamic biometric modality handwriting. As for all biometric identification and verification systems, it is crucial to protect the original biometric raw data (templates) in order to prevent all kinds of misuse of individual and personal data. Identity theft is only one example, out of many others, that can be done with eavesdropped information. However, due to the variability of biometric data, templates cannot easily be protected by common cryptographic hash algorithms, like they are used in common password authentication systems comparing two passwords in hash domains. The variability (intra-class variability) has to be taken into account to ensure the reproducibility and protection of the template. One possible method to ensure reproducibility and simple template protection is for example the Biometric Hash algorithm for handwriting, originally introduced in [1] with further discussions in [2]. The goal of this method is to transform intra-subject biometric data, subject to variability, into stable and individual hash vector values; an overview is given in section 2. Although not originally suggested as template protection method in [1], we consider the application of the Biometric Hash scheme for template protection; due to its similar properties as cryptographic hash functions (see section 2).

Motivated by work from [3] and [6] for fingerprint modality, we investigate the generation of raw data based on a reference BioHash, which has the property to produce identical Biometric Hash values as the reference. Our approach is to exploit weaknesses from the intra-class-compensation process within the Biometric Hash generation, which allows different sets of raw data to produce an identical Biometric Hash value as the reference. First experiments on our new approach show the possibility to reconstruct such valid raw data.

Our proposed method is similar to a ciphertext-only-attack on cryptographic hashes, as described in the literature; see e.g. Bishop [5]. The ciphertext-only-attack depicts a scenario where an attacker has access to a collection of hashes and tries to determine the plaintext out of it. In our case we deal with an adaptive ciphertext-only-attack where side information such as system parameters for the BioHash algorithm is given. We further discuss our attack on the Biometric Hash algorithm for handwriting introduced in [1] where the Interval Matrix (IM) represents the side information in terms of system parameters of a BioHash algorithm.

The structure of the paper is composed as follows. Section 2 gives an overview to the BioHash algorithm for handwriting. We discuss potential vulnerabilities of the algorithm and consider possible attack methods based on these flaws. In section 3, the detailed design for the reconstruction of raw data is presented. The experimental evaluation is introduced in section 4, results and discussion are summarized. Finally, we present a conclusion, a comparison to the achieved results for fingerprints in [3] and our future work based on our findings.

2 Biometric Hash for Handwriting

The motivation to create a Biometric Hash algorithm is to develop a method which fulfils a similar task like a cryptographic hash does. However, due to the variability of

the input data a slightly different specification has to be made. The main differences and similarities of properties of both Biometric Hash and cryptographic hash are shown in Table 1. We denote *CH* as a cryptographic hash function, *BH* as a Biometric Hash function, *a* and *a'* as arbitrary digital input data (authentication information), *h* as a cryptographic hash, *b* as a BioHash and *P* and *P'* indicate two different persons.

Table 1. Brief overview of differences and similarities of cryptographic and biometric hashes

Property	Cryptographic Hash	Biometric Hash
a) Reproducibility	The hashes *h* and *h'* of a cryptographic hash function *CH* are identical, if *a* and *a'* are identical. *CH(a)=>h, CH(a')=>h', h=h',* if *a=a'*	The BioHashes *b* and *b'* of a Biometric Hash function *BH* are identical, if *a* and *a'* belong to the same person *P*. *BH(a)=>b, BH(a')=>b', b=b',* if *a* and *a'* belongs to *P*
b) Collision Resistance	It is difficult to find hashes *h* and *h'* of a cryptographic hash function *CH* which are identical, if *a* and *a'* are not identical. *CH(a)=>h, CH(a')=>h', h≠h',* if *a≠a'*	It is difficult to find BioHashes *b* and *b'* of a Biometric Hash function *BH* which are identical, if *a* and *a'* belong to two different persons *P* and *P'* respectively. *BH(a)=>b, BH(a')=>b', b≠b',* if *P≠P'*
c) Non-Reversibility	It should be computably hard to calculate or estimate the input data *a* out of a hash *h=>CH(a),* within a realistic time scale.	It should be computably hard to calculate or estimate the input data *a* out of a BioHash *b=>BH(a)*, within a realistic time scale.
d) Bit Sensitivity	Minor changes within *a* should have a massive effect on *h=>CH(a).*	Changes within *a* should have no effect on *b=>BH(a)*, if it derives from the same person *P*.

The Biometric Hash generation process differs in comparison to the cryptographic hash generation by the characteristics (a) Reproducibility, (b) Collision Resistance and (d) Bit Sensitivity. In consideration of these characteristics the BioHash from [1] and [2] belongs to a class of "Fuzzy commitment schemes", where a certain scope of variability is tolerated to get to the same result (introduced by Al-saggaf et al. in [7]).

The basic idea behind the BioHash algorithm is to extract a set of statistical feature values from actual handwriting samples and to find a parameterized transform function for mapping of these values to a stable hash value space. A workflow on the algorithm for handwriting is shown in Figure 1.

A human handwritten input (e.g. a handwritten signature) is acquired by a sensor which transforms pen positions and pressure into an analogous electrical signal (in the following we briefly summarize the main steps from [1]). This signal is converted by an analog-digital converter into a digital signal *a*, i.e. the digital raw data. Such digital raw data *a* is composed of time-dependent pen positions and corresponding pressure and angle values. Within the Biometric Hash generation process a feature extraction function determines statistical features based on the raw data *a*. Statistical features define characteristics such as *total-write-time*, *total-number-of-event-pixels* or *maximum-pressure* for example; a complete list of all actual used features can be found in table 2.

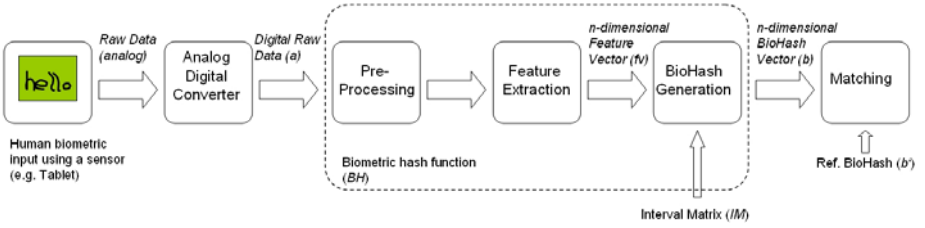

Fig. 1. General workflow of biometric bash generation

All determined features are collected in a feature vector of a dimension of n, whereas n denotes the number of statistical features being used during the extraction process. With help of an Interval Matrix (IM), these statistical features are mapped into a n-dimensional Biometric Hash vector (BioHash). The Interval Matrix is generated during an enrollment process for each subject and consists of an Interval Length Vector $\measuredangle I$ and an Interval Offset Vector Ω: $IM = (\measuredangle I, \Omega)$. Its function is to compensate intra-class variability of a subject by mapping a certain value range into one specific value. Each feature possesses a related pair of Interval Length and Interval Offset, therefore the length of Interval Matrix and feature vector are equal. The mapping for each feature element fv_i within a feature vector fv into a BioHash element b_i based on the Interval Matrix (Interval Length Vector and Offset Vector) is described in the following Equation 1, whereas i denote the index from 1 to n.

$$b_i = \left\lfloor \frac{fv_i - \Omega_i}{\Delta I_i} \right\rfloor \tag{1}$$

The result of a Biometric Hash generation process is an n-dimensional Biometric Hash vector b (BioHash). In verification mode this BioHash b is compared to a reference BioHash b_{ref}. The matching can be done, for example, by calculating the Hamming distance (amount of equal vector elements) between b and b_{ref}. Reference BioHash b_{ref} and Interval Matrix (IM) is stored for each user in a database.

A more detailed description on the BioHash algorithm is given in [1] and a further discussion in [2]. The BioHash cannot only be generated out of personal handwriting such as signatures, it is also possible or even advised to use pass phrases, pseudonyms, symbols or Personal Identification Numbers (PIN). These alternative handwriting samples are called semantics. It has been observed in [1] that these kinds of semantics produce similar recognition accuracy as compared to handwriting signatures, without disclosing the true identity of the writer. General comment: In the original design of the BioHash neither of the aspects of irreversibility nor attack scenarios has been considered. Therefore we address these aspect in this paper.

2.1 Potential Design Vulnerabilities of the BioHash for Handwriting

Our idea is to consider the Biometric Hash scheme as a method for template protection and to analyze the vulnerabilities for the reconstruction of raw data. We do so based on the assumption that an attacker has compromised a biometric based verification

Table 2. List of all features used during the BioHash generation process. Note: our attack classifies features into *cb – calculated basic feature, ib – interactive basic feature* and *gc – genetically calculated features* which are explained in the following sections.

fv_i: i=	Parameter Description	fv_i: i=	Parameter Description
1 (cb)	Total writing time in ms	53(gc)	Numeric Integration of Y values for 3rd one-fifth time period
2 (cb)	Total number of event pixels	54 (gc)	Numeric Integration of Y values for 4th one-fifth time period
3 (gc)	Image Width * 1000 DIV Height	55 (gc)	Numeric Integration of Y values for 4th one-fifth time period
4 (cb)	Average velocity in x direction in 1000 * pixels / ms	56 (gc)	Average Pen Down Pressure normalized to 1 * 1000
5 (cb)	Average velocity in y direction in 1000 * pixels / ms	57 (gc)	Average PenUp Pressure normalized to 1 * 1000
6 (ib)	Number of consecutive pen-down segments	58 (gc)	Baseline Angle of the Sample
7 (gc)	Minimum absolute x-velocity during sample	59 (gc)	Histogram of Y for Zone 1 in % * 100
8 (gc)	Maximum absolute x-velocity during sample	60 (gc)	Histogram of Y for Zone 2 in % * 100
9 (gc)	Minimum absolute y-velocity during sample	61 (gc)	Histogram of Y for Zone 3 in % * 100
10 (gc)	Maximum absolute y-velocity during sample	62 (gc)	Area(ConvexHull) vs. Area(BoundingBox) * 1000
11 (gc)	Centroid of horizontal pen position in bounding box	63 (gc)	Area(ConvexHull(Segments)) vs. Area(ConvexHull(Sample)) *
12 (gc)	Centroid of vertical pen position in bounding box	64 (gc)	Area(ConvexHull(Segments)) vs. Area(BoundingBox) * 1000
13 (gc)	Distance of Centroid from origin	65 (gc)	PathLength(ConvexHull) vs. PathLength(BoundingBox) * 1000
14 (ib)	Maximum absolute pressure occurred during writing	66 (gc)	PathLength(ConvexHull(Seg.)) vs.
15 (gc)	Centroid of horizontal pen position	67 (gc)	PathLength(ConvexHull(Seg.)) vs. PathLength(BoundingBox)
16 (gc)	Centroid of vertical pen position	68 (gc)	Histogram of X for left in % * 100
17 (gc)	Distance of Centroid from origin	69 (gc)	Histogram of X for right in % * 100
18 (gc)	Horizontal azimuth of centroid from origin	70 (gc)	Amount of maxima in X direction
19 (ib)	Maximum absolute altitude of pen occurred	71 (gc)	Amount of minima in X direction
20 (ib)	Minimum absolute altitude of pen occurred	72 (gc)	Amount of maxima in Y direction
21 (ib)	Maximum absolute azimuth of pen occurred	73 (gc)	Amount of minima in Y direction
22 (ib)	Minimum absolute azimuth of pen occurred	74 (gc)	Ratio of maxima in X direction vs. maxima in Y directions
23 (ib)	Average Writing Pressure relative to MaxPressure	75 (gc)	Ratio of minima X direction vs. minima directions
24 (ib)	Average Azimuth of pen projected on writing plane	76 (gc)	Amount of crossing points (intersections)
25 (ib)	Average Altitude of pen above the writing plane	77 (gc)	Amount of intersections with line at 1st quarter of X
26 (gc)	Normalized Average velocity in x direction in pixels	78 (gc)	Amount of intersections with line at 2nd quarter of X
27 (gc)	Normalized Average velocity in y direction in pixels	79 (gc)	Amount of intersections with line at 3rd quarter of X
28 (ib)	Absolute cumulated Pen-up time in ms	80 (gc)	Amount of intersections with line at 4th quarter of X
29 (gc)	Ratio of Pen-ups by total write time * 1000	81 (gc)	Amount of intersections with line at 1st quarter of Y
30 (gc)	Total Number of Sample Values	82 (gc)	Amount of intersections with line at 2nd quarter of Y
31 (gc)	Total absolute Path Length in Pixels	83 (gc)	Amount of intersections with line at 3rd quarter of Y
32 (gc)	Number of pixels in first row, first column	84 (gc)	Amount of intersections with diagonal line (up. left to bottom right)
33 (gc)	Number of pixels in first row, second column	85 (gc)	Amount of intersections with diagonal line (up. right to bottom left)
34 (gc)	Number of pixels in first row, third column	86 (gc)	Ratio of distance start/end to path length
35 (gc)	Number of pixels in first row, fourth column	87 (gc)	Ratio of distance min(X)/max(X) to path length * 1000
36 (gc)	Number of pixels in second row, first column	88 (gc)	Ratio of distance min(Y)/max(Y) to path length * 1000
37 (gc)	Number of pixels in second row, second column	89 (gc)	Ratio of distance start-centroid to end-centroid * 1000
38 (gc)	Number of pixels in second row, third column	90 (gc)	Mapping of maxima/minima in X to a value
39 (gc)	Number of pixels in second row, fourth column	91 (gc)	Mapping of maxima/minima in Y to a value
40 (gc)	Number of pixels in third row, first column	92 (gc)	Mapping of maxima/minima in P to a value
41 (gc)	Number of pixels in third row, second column	93 (gc)	Mapping of maxima/minima in A to a value
42 (gc)	Number of pixels in third row, third column	94 (gc)	Range of all stroke points
43 (gc)	Number of pixels in third row, fourth column	95 (gc)	Pixels inside radius 1/3*Boundingbox to point with least
44 (gc)	Numeric Integration of normalized X values	96 (gc)	Pixels inside radius 1/3*Boundingbox to point with most neighbour
45 (gc)	Numeric Integration of normalized Y values	97 (gc)	Pixels inside radius 1/3*Boundingbox to point with average
46 (gc)	Numeric Integration of X values for 1st one-fifth tp.	98 (gc)	Average angle of all cross-point-angles between 0-30°
47 (gc)	Numeric Integration of X values for 2nd one-fifth tp.	99 (gc)	Average angle of all cross-point-angles between 31-60°
48 (gc)	Numeric Integration of X values for 3rd one-fifth tp.	100 (gc)	Average angle of all cross-point-angles between 61-90°
49 (gc)	Numeric Integration of X values for 4th one-fifth tp.	101 (gc)	Angle count of all cross-point-angles between 0-30°
50 (gc)	Numeric Integration of X values for 4th one-fifth tp.	102 (gc)	Angle count of all cross-point-angles between 31-60°
51 (gc)	Numeric Integration of Y values for 1st one-fifth tp.	103 (gc)	Angle count of all cross-point-angles between 61-90°
52 (gc)	Numeric Integration of Y values for 2nd one-fifth tp.		

system and has access and knowledge to username, reference BioHash b_{Ref}, and Interval Matrix (*IM*) for each registered individual. The operating principle of the BioHash algorithm is published and is accessible for everyone who is interested in (Kerkhoff principles).

The first thing that attracts our attention is the Interval Matrix and the mapping function, which maps the feature vector into a BioHash. When BioHash b_{Ref} and corresponding Interval Matrix *IM* are given, we can perform a reverse mapping to create a feature vector fv_{calc}. If we convert Equation 1 according to fv_i, we calculate the lower limit of a value range which can be mapped to b_i. By adding the half of $\measuredangle I_i$ to it, we

compute the middle of the value range (see Figure 2) in feature space. Equation 2 formulates this approach, whereby fv_i is replaced with $fv_{calc,i}$ because they are not necessarily equal, related to the rounding in equation 1.

$$fv_{calc_i} = b_{ref_i} \cdot \Delta I_i + \Omega_i + \frac{\Delta I_i}{2} \tag{2}$$

Figure 2 gives a visual point of view on the reverse mapping done in Equation 2. Using this method, it is possible to calculate a complete feature vector fv_{calc}. Due to the fact that fv_{calc} is determined from b_{Ref} and corresponding *IM*, it can be mapped with help of *IM* to b_{Ref} again and therefore be used to reconstruct raw data, based on it.

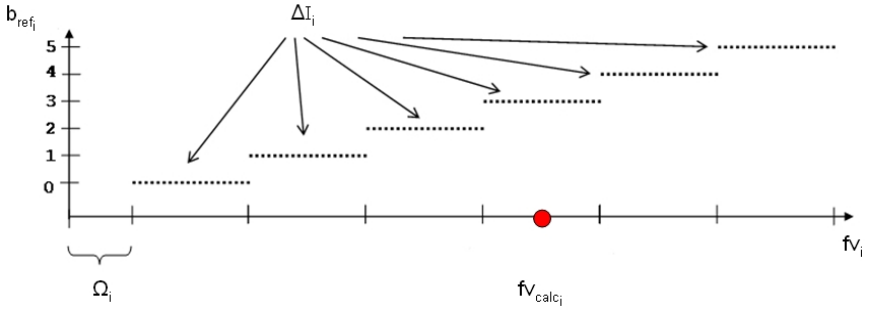

Fig. 2. Example of backward calculation (reverse mapping) of a feature vector element $fv_{calc,i}$ based on the corresponding reference BioHash $b_{ref,i}$, Interval Length ΔI and Interval Offset Ω_i

If an attacker takes advantage of this vulnerability (reverse mapping) he can reduce his work on reconstructing raw data based on that calculated feature vector fv_{calc}, i.e. in feature space rather than on the BioHash. The next section shows a possible method to achieve this.

2.2 Attack Considerations

By analyzing the feature vector and it's consisting statistical feature it turns out that some features can be used to build a basic structure to reconstruct raw data. For that reason it is necessary to calculate or determine some fundamental data. The idea is now to find fundamental data that helps to form the basic structure by looking into *total-write-time*, *total-number-of-event-pixels*, *maximum-value-in-x-direction* and *maximum-value-in-y-direction*. The first two values can easily be determined by reading the corresponding values out of the feature vector fv_{calc}. Maximum value in x and y direction can be calculated out of the features *total-write-time* and *average-velocity-in-x-direction* respectively *average-velocity-in-y-direction*. These four mentioned features, which we refer to as *calculated basic features*, form the basic raw data structure (see Table 2 marked with *cb*).

Based on this basic structure it is now possible to manually implement other features as well. This can be done as follows: an experienced user chooses a specific feature fv_i he wants to implement from fv_{calc}, he changes the basic raw data structure corresponding to that feature fv_i then he determines a temporary feature vector fv_{temp}

for this new set of raw data and compares feature fv_i from fv_{temp} with feature fv_i of feature vector fv_{calc}. If they are not the same, he changes the basic raw data structure again, until they match. This procedure is time-consuming and needs a lot of experience in interpreting and manipulating raw data in a way to fulfill a certain feature. However, by determining these features manually, we expect that it leads to more natural results within the artificial raw data. To perform a first test to determine the success tendency of this approach under controlled conditions, we introduce a time limit for this interactive process in our experiments as described in section 4.1.

Initial considerations about which features can be set manually revealed that pressure and angle based features are our first choices. That is because they are independent from horizontal and vertical (x/y coordinates) based features, which make up more than *80%* of all features used and described in [1]. Once the pressure and angle based features are implemented inside the raw data they can be locked, so that they cannot be changed anymore in the further reconstruction process. We call these pressure based, angle based and all other features that can be implemented manually within a raw data structure *interactive basic features* (see Table 2 marked with *ib*). All *calculated* and *interactive basic features* are additional summarized in Table 3.

Table 3. Classification of features based on the feature list in Table 2

Feature class	Description	Dedicated features
Calculated Basic Features (fv_{cb})	Features that form the basis for a raw data structure by calculating and determine fundamental data	$fv_{cb} = \{fv_1, fv_2, fv_4, fv_5\}$
Interactive Basic Features (fv_{ib})	Features that are implemented into the basic raw data structure manually	$fv_{ib} = \{fv_6, fv_{14}, fv_{19}, fv_{20}, fv_{21}, fv_{22}, fv_{23}, fv_{24}, fv_{25}, fv_{28}\}$
Genetically Calculated Features (fv_{gc})	All features that are implemented into raw data by a genetic algorithm.	$fv_{gc} = fv_{all} \setminus fv_{cb} \setminus fv_{ib}$
Overall features (fv_{all})	All features used during the BioHash determination process	$fv_{all} = \{fv_1, \ldots, fv_{103}\}$

Obviously, if *calculated* and *interactive basic features* are implemented into a set of raw data then all the remaining features have to be implemented as well to generate a feature vector that matches fv_{calc}. This can be done in many different ways (e.g. brute-force attack). Our first idea is to implement the remaining features into the raw data by using a genetic algorithm, first presented by Holland in 1975 [4]. We define features which are calculated this way as *genetically calculated features* (see also marked in Table 2 and listed in Table 3).

In the next subsection we focus on the description of our reconstruction approach based on the mentioned attack consideration by involving genetic algorithms.

3 Design Approach for Reconstruction

In this section we give a detailed description on our first design approach for reconstructing raw data out of a given BioHash and corresponding Interval Matrix, based on the vulnerability and feature classification, discussed in section 2.

Our approach can be divided into four major steps. The first step is to calculate the feature vector fv_{calc} as described in section 2.1. During the next step, we build a basic raw data structure based on the *calculated basic features* as introduced in section 2.2. The third step implies the implementation of *interactive basic features* into the basic raw data structure as presented in section 2.2. Finally, our last and fourth step is to determine all remaining features based on the raw data structure, by using a genetic algorithm, which is detailed later on in this section. These four steps build our new design approach for reconstructing raw data and are depicted in Figure 3.

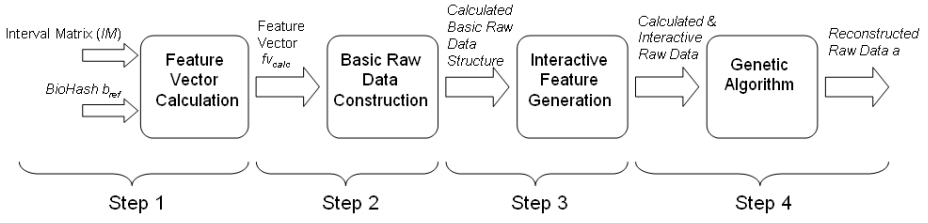

Fig. 3. Illustration of our new approach to reconstruct raw data out of BioHash b_{ref} and corresponding Interval Matrix (*IM*)

Step 1: With help of given BioHash and corresponding Interval Matrix (*IM*) we calculate the feature vector fv_{calc} based on equation 2. As already mentioned in section 2.1, fv_{calc} is not necessarily, or even likely to be equal to the original feature vector, but leads to the same BioHash b_{ref}, when mapped with the corresponding Interval Matrix. The feature vector fv_{calc} forms the basis for our reconstruction process described in steps 2 to 4.

Step 2: Based on fv_{calc} we build a basic raw data structure using the *calculated basic features*. This helps to implement additional features since a basic raw data structure is now given and has only to be modified.

Step 3: A user tries to implement all *interactive basic features* into the basic raw data structure, in a defined limited period of time to have a first evaluation with controlled time constrains. Depending on the experience of a user and constellation of fv_{calc} it is possible that not all *interactive basic features* can be implemented due to this time constant.

Step 4: All remaining features fv_{gc} are transcribed into the raw data structure by a genetic algorithm (GA), which is described as follows. A start population is composed of individuals, which are randomly generated. In order to generate individuals that represent a realistic signature, they have to be build of continuous horizontal and vertical signal components. Motivated by Galbally et al. in [8], where synthetic signatures are generated based on spectral analysis; we created an algorithm which generates different, almost realistic signatures. Due to the fact that we already generated pressure and angle values (implemented with the *calculated* and *interactive basic features*), only horizontal and vertical raw data values needs to be added. All individuals now possess an implementation of the same basic features and randomly generated continuous horizontal and vertical signals. The fitness function for each individual is based on the method “survival of the fittest”. To determine the fittest

individuals, a BioHash is calculated for each individual (based on given Interval Matrix) and compared, using the normalized Hamming distance, to the reference BioHash b_{ref}. Obviously, in our case, the normalized Hamming distance between any 2 BioHash vectors is defined by the number of non-equal vector components divided by the vector's dimension. Individuals which achieve the highest scores are defined as the fittest. Only the fittest are taken into the next round combined with a defined survival rate. We utilize the BioHash Hamming distance as fitness function, since this is calculated based on the original feature vector values, whereas fv_{calc} is just estimation. We use the genetic operators' mutation and crossover to create new generations and modify raw data in such a way that the before mentioned *calculated* and *interactive basic features* are preserved. Mutations are not always applied to individuals during the generation creation; it is randomly controlled using a mutation rate. The mutation operator changes only a part of an individual when it occurs. We only use a *one-point-crossover* genetic operator during a creation of a new generation. It swaps sub-sections of two individuals, whereby these sub-sections are at the same position within the two individuals. The genetic algorithm terminates if one of the following conditions is reached: (1) returned BioHash and reference BioHash are equal (or a specific threshold is reached), (2) individuals do not change any more in a positive way (higher matching scores) after multiple generation cycles or (3) a specific defined amount of generation cycles has been passed (e.g. x=100 iterations). An overview on the GA workflow is shown in Figure 4.

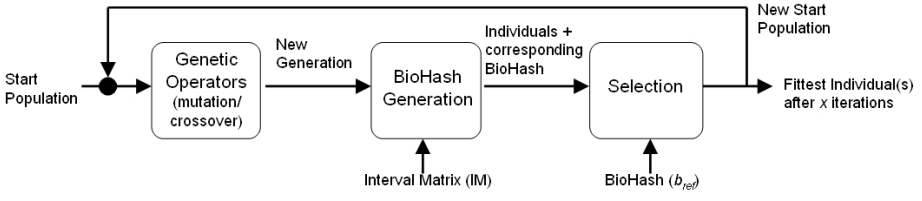

Fig. 4. Basic workflow of the used genetic algorithm during raw data reconstruction

After all four steps are performed; one or more raw data sets were reconstructed, depending on the GA result (two or more individuals achieve the highest fitness score). In order to evaluate our new design approach, we perform experiments, as described in the next section.

4 Experimental Evaluation

In this chapter we summarize our first experiments to evaluate the proposed reconstruction attack. Our goal is to see how successfully we can generate artificial biometric handwriting raw data on given Interval Matrix and corresponding BioHash by measuring the archived $FAR_{reconstr.}$ with the reconstructed raw data. First we define the experimental settings. Secondly we introduce our methodology to provide a comparative study of the achieved results to the general verification performance. Thirdly, results are presented and discussed.

4.1 Overall Settings

The biometric database of our initial tests consists of *5* subjects, which each have donated *6* handwriting samples for two different semantics (PIN and Symbol). The given PIN is a sequence of the five digits *77993*. Using this semantic, the individual kind of writing plays a more important role than the content to recognize a person as its self or distinguish him/her from other users. The freely chosen Symbol is based on individual creative characteristics and provides a knowledge based component in form of the sketched object (e.g. order of single strokes to create the symbol). In order to create the reference data, an Interval Matrix and a reference BioHash are calculated for each person using the first five handwriting samples. The remaining sixth sample is applied for verification.

In the next paragraphs we provide details about the settings for our experiments during the reconstruction process for step 3 and 4. In step 1 and 2 no parameter needs to be set. The dimension of Interval Matrix is 2x103 and thus the size of the BioHash vector 103; therefore 103 features are used during the BioHash determination and raw data reconstruction in our tests.

During the third step (setting interactive feature) in the reconstruction process, in our first test one human attacker and forger tries to implement the *interactive basic features* into a basic raw data structure, for this procedure we define a maximum processing time of *10* minutes.

Within step four of the reconstruction process, the settings for the genetic algorithm is as follows: start population of *100* individuals, *90%* survival rate, the recombination rate and mutation rate is *10%* (*10%* of all individuals are recombined or mutated). The genetic algorithm terminates if a *100%* matching rate is accomplished or the matching score does not change any more in a positive way after two full cycles (all features). The matching to determine the fittest individual is accomplished by calculating the Hamming distance as described in section 3. The fittest individual represents the reconstructed raw data and is used during verification together with genuine raw data. In Table 4 are all settings summarized.

Table 4. Overview of all overall settings

General Settings		Step 3: Interactive feature generation	
Number of b_{ref} per user:	10 (5 per semantic class PIN/Symbol)	Forger:	One computer science student (age: 26) as interactive attacker
SizeOf b_{ref}:	103	Forgery time:	≤ 10 min
SizeOf fv_{cb}:	4 features	Step 4: Genetic Algorithm	
SizeOf fv_{ib}:	10	Start population	100 individuals
SizeOf fv_{ge}:	89	Survival rate	90%
SizeOf fv_{all}:	103	Recombination rate	10%
		Mutation rate	10%
		Termination criteria	100% match or no improvement of fitness function after 2 generations

4.2 Evaluation Methodology and Measurements

In order to compare the performance of a verification using the reconstructed raw data with the verification using genuine raw data, biometric error rates FRR/FAR and EER are calculated.

The FRR (false rejection rate) describes the ratio between the number of false rejections of authentic persons and the total number of tests. Generally, the FAR (false acceptance rate) is the ratio between number of false acceptances of non-authentic persons and the entire number of authentication attempts. In our evaluation, we perform two kinds of false acceptance tests: firstly, interclass FAR ($FAR_{inter\text{-}class}$) errors are calculated by comparing BioHash values of all subjects against each other and analyzing false acceptances against normalized Hamming distance thresholds. Secondly, we determined false acceptances generated by the reconstructed raw data against the same threshold, in the following denoted as $FAR_{reconstr.}$.

For a comparative analysis of verification performance, the EER (equal error rate) is a common measurement in biometrics. EER denotes the point in error characteristics, where FRR and FAR yield identical value. In the further discussion of our experimental results, we analyze error rate diagrams, which consists of error rates graphs for $FAR_{inter\text{-}class}$, $FAR_{reconstr.}$, and FRR for each writing semantics to illustrate $EER_{inter\text{-}class}$ respectively $EER_{reconstr.}$.

4.3 Results and Discussion

The results of our test with different semantic classes are displayed in Figure 4. In average the reconstructed raw data produces an $EER_{reconstr.}$ of *75%* for the semantic PIN, whereas the original user based raw data leads to an $EER_{inter\text{-}class}$ of only *10%* (Figure 5) for inter-class verification. This means, that random forgeries (interclass tests) cause lower false acceptance than our achieved reconstructed raw data.

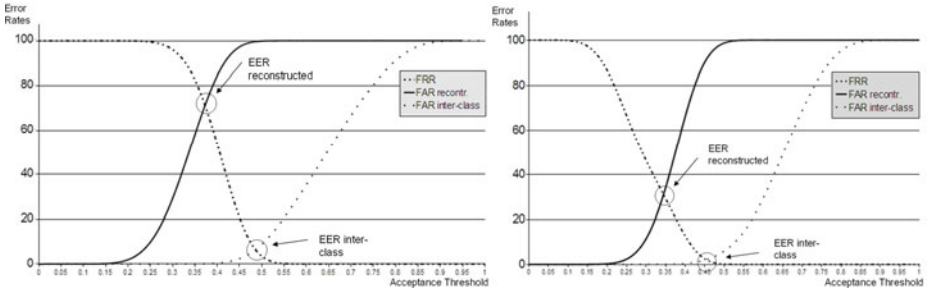

Fig. 5. FRR, $FAR_{inter\text{-}class}$ and $FAR_{reconstr.}$ for PIN (left) and Symbol (right)

For semantic class *symbol* the reconstructed raw data leads to an $EER_{reconstr.}$ of *30%* and an interclass $EER_{inter\text{-}class}$ of approximately *5%* (see Figure 4 right). It shows that the semantic class *Symbol* is more resistant to our attack based on reconstructed raw data then the semantic class *PIN*. The results reflect the first attempt to generate biometric raw data corresponding to a given Interval Matrix and corresponding reference BioHash. Please note that this is only a first attempt of producing reconstructed raw data and can just point into a direction because of the relatively limited amount of semantics and users during the test. However, these results show that in our case the reconstructed raw data creates a significantly higher FAR then the original user based raw data, which allows attackers to reproduce BioHash values in *75%* of all trails.

5 Conclusion and Future work

In this paper we have suggested a method for reconstructing biometric raw data from given BioHash values, by using features derived from user interaction and genetic algorithm for approximation. Our work reveals some vulnerabilities of the Biometric Hash algorithm for handwriting, introduced in [1]. Based on these vulnerabilities and motivated by earlier work e.g. for generating artificial forgeries [6], it is possible to design and implement an attack to generate raw data based on *calculated* and *interactive basic feature* determination, with an additional genetic algorithm. Our first experiment shows that such generated forgeries produce an $EER_{reconstr.}$ in the range of 30% to *75%* as compared to non-attack inter-class $EER_{inter\text{-}class}$ of *5%* to *10%*. If we compare our results with the one made in [3] by Galbally et al., where a fake fingertip was created from an image reconstructed from a minutiae template, we recognize similarities. The test set is in comparison to 5 Symbols and 5 PINs limited to 10 different fingerprints. They are using an attack rate to present the evaluation results, instead of an EER, because the FAR for all 5 considered thresholds are *0%*. By using the same measurement methodology we achieve similar results. The success-attack-rate in [3] is higher (*30%-100%*) compared to our attack rates (*0%-70%*) as we expected, due to the fact that fingerprint based recognition systems are more accurate in distinguishing different fingerprint samples.

In order to eliminate the vulnerability to the BioHash algorithm we suggest to not use *calculated basic features* during the BioHash generation process. It is more difficult to reconstruct raw data using our new approach if *calculated basic features* are nonexistent or derive to complex additional features.

In our future work we will run tests with more semantic classes and users to extend our first results. We also plan to examine all features being used during the BioHash generation to find more that can be used as *calculated basic features* or *interactive basic features*. The development of a more advanced genetic algorithm is also planned in the future. Another consideration is the development of an automatic approach, which makes an interactive interference needless.

Acknowledgements

This work is supported by the German Federal Ministry of Education and Research (BMBF), project “OptiBioHashEmbedded”) under grant number 17N3109). The content of this document is under the sole responsibility of the authors.

References

1. Vielhauer, C.: Biometric User Authentication for IT Security: From Fundamentals to Handwriting, Springer, New York (2006)
2. Vielhauer, C. and Steinmetz, R. and Mayerhoefer, A.: Biometric Hash based on Statistical Features of Online Signatures: In: Proceedings of the IEEE International Conference on Pattern Recognition (ICPR), vol.1, pp.123-126, (2002)

3. J. Galbally, R. Cappelli, A. Lumini, D. Maltoni and J. Fierrez: Fake Fingertip Generation from a Minutiae Template, in *Proc. Intl. Conf. on Pattern Recognition, ICPR*, Tampa, USA, (2008)
4. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology. Control and Artificial Intelligence: MIT Press, 1995, First Published by University of Michigan Press (1975)
5. Bishop, M.: Computer Security, Addison-Wesley, Boston, U.S.A, ISBN 0-201-44099-7, (2003)
6. R. Cappelli, A. Erol, D. Maio and D. Maltoni: Synthetic Fingerprint-image Generation, in proceedings 15th International Conference on Pattern Recognition (ICPR2000), Barcelona, vol.3, pp.475-478, (2000)
7. Alawi A. Al-saggaf, Acharya H. S.: A Fuzzy Commitment Scheme, In: Proc. IEEE International Conference on Advances in Computer Vision and Information Technology, India (2007)
8. Galbally, J., Fierrez, J., Martinez-Diaz M. and Ortega-Garcia, J.: Synthetic Generation of Handwritten Signatures Based on Spectral Analysis, in Defense and Security Symposium, Biometric Technologies for Human Identification, BTHI, Proc. SPIE, Orlando, USA (2009)