Security of Features Describing the Visual Appearance of Handwriting Samples Using the Bio-hash Algorithm of Vielhauer against an Evolutionary Algorithm Attack

Andreas Hasselberg[1], Rene Zimmermann[1], Christian Kraetzer[1], Tobias Scheidat[2], Claus Vielhauer[2], and Karl Kümmel[2]

[1] Otto-von-Guericke-University Magdeburg, Germany
[2] Brandenburg University of Applied Sciences

Abstract. To improve the security and stability of biometric handwriting samples a Bio-Hash algorithm for handwriting was introduced in [1]. It utilizes features to describe how the sample was written, but the current set of features does not characterize the visual appearance of the sample itself. In this paper we present a set of new features derived from handwriting forensics and OCR algorithms to address this issue. Furthermore, here the security of the old and new sets of features is evaluated for their resilience against a new, fully automated attack trying to compute raw data matching a given hash vector.

The main contributions of this paper are: The introduction of new features with a potential to increase the attack resilience of the Bio-Hash algorithm, and, an improvement of the attack approach from [6] to produce more realistic looking synthetic handwriting signals.

Keywords: bio-hash, evolutionary algorithm, handwriting, biometrics.

1 Introduction

The most common approach to efficiently protect the biometric templates in a system (i.e. the reference data) is to transform the data using a one-way-function. An example for this kind of function is the biometric hash algorithm for dynamic handwriting (hereafter called Bio-Hash algorithm) from Vielhauer [1].

In [6] Kümmel et al. introduce an evolutionary algorithm based attack to the template space of the Bio-Hash algorithm. The main drawbacks to this rather successful attack are the fact that it relies on user interaction and the strongly artificial look of its output.

In this paper we make two contributions to the research in this field:

A) We extend the feature space used by the Bio-Hash algorithm for dynamic handwriting based user authentication. Our new features aim at the description of the visual appearance of the sample itself, something that is amiss amongst the features used until now.

B. De Decker et al. (Eds.): CMS 2013, LNCS 8099, pp. 85–94, 2013.

B) We extend the attack from [6] to a completely automated (i.e. without the user interaction required in [6]) means to create synthetic handwritings which, on one hand, produces a Bio-Hash similar to the reference Bio-Hash and, on the other hand, look natural enough to be able to fool visual inspection of the input in an authentication scenario.

Our results imply for the first contribution that we introduce two (of the 13 new features) which show a strong potential to increase the attack resilience of the Bio-Hash algorithm, because they have been exceptionally hard to successfully attack in our evaluations. For the second main contribution, our improved attack generates much more realistic looking synthetic handwritings.

The paper is structured as follows: Section 2 summarizes some required basics. In section 3 the prerequisites, ideas and the algorithms to compute the 13 new features describing the visual appearance of online captured handwriting samples are presented. In section 4 the attack approach used for evaluation is described with its prerequisites and its implementation. Section 5 contains the evaluation results, while the final section 6 draws a conclusion and presents possible direction for future work.

2 Related Work

The template used in biometric authentication systems has to be protected for various security, legal and ethical reasons. For handwriting data, one very strong motivation for such protection is the fact that the template might contain enough information for re-engineering parts of the original biometric data, e.g. a handwritten signature. Amongst the various protection approaches one of the most prominent is the Bio-Hash algorithm of Vielhauer [1], which has been developed for handwriting, but could also be used for different biometrics (e.g. voice).

The Biometric Hash (short: Bio-Hash) algorithm was introduced by Vielhauer in [1] to generate individual, stable hash values from dynamic handwriting data as well as to perform biometric verification based on these hashes. Generally, the raw data of each dynamic handwriting sample derived from a handwriting digitizer device (e.g. Tablet PC) consists of a time-dependent sequence of physical values. Each sample point of such a sequence contains the five values horizontal pen tip position, vertical pen tip position, pen tip pressure and pen orientation angles altitude and azimuth.

During the enrollment process, n handwriting raw data samples are acquired per person. From each of n-1 raw data samples, one k-dimensional statistical feature vector containing static and dynamic features is calculated. The individual Interval Matrix (IM) consists of a vector containing a mapping interval length for each feature and an offset value vector. Both are calculated based on the analysis of intra-class variability of the person using the n-1 statistical feature vectors acquired during enrollment session. There are two parameters to influence the Bio-Hash generation by scaling the mapping intervals stored in the IM: the tolerance factor TF and tolerance

vector TV. TF is a global hash generation parameter, which is a scalar value. Using the TF, the mapping intervals of all features are scaled by the same global factor. In contrast, TV provides an individual scaling of the mapping interval of every single statistical feature. Based on the statistical feature vector derived from the remaining enrollment sample of the person and its individual IM, the interval mapping function determines the reference Bio-Hash vector b_{ref} of this user. Therefore, the feature dependent interval lengths and offsets provided by IM are used to map each of the corresponding statistical features to a hash value. Each further biometric hash is calculated in the same manner, whether it is used for biometric verification or hash generation. In case of verification, a Bio-Hash vector b, which is derived from the currently presented handwriting sample, is compared against the reference hash vector b_{ref} by using the Hamming distance and a predefined threshold. For more details of the particular calculation steps, the interested reader is referred to [1].

In order to build the feature vector, 131 features are extracted from each raw data sample. As described in [7] by Makrushin et al. the features can be classified by the signals required for calculation: 3 time based statistics features, 88 static spatial statistics features (time and order independent), 8 dynamic spatial statistics features, 22 pressure-based statistics features and 10 angles-based statistics features.

In [6] Kümmel et al. introduce an attack that uses a genetic algorithm for raw handwriting data reconstruction. In a Kerckhoffs' compliant setup it assumes that the attacker gains access to the Bio-Hash b_{ref} of a user and the corresponding IM and can therefore evaluate the fitness of artificially created handwritings by computing the Hamming distance to b_{ref}. Despite the fact that this attack works quite well, as shown in [6], it suffers from two rather severe drawbacks that are addressed in this paper: first, the attack in [6] is requiring user interaction for the implementation of selected features, and second, its output looks artificial to a human observer (see an comparison between a real handwriting and the outputs of the attack tool from [6] and our attack in figure 2 at the end of the document). Both drawbacks are addressed here, by introducing a new, fully automated, genetic algorithm based attack that aims at the generation of natural looking artificial handwritings.

3 Description of the New features

The feature extraction follows a pipeline which is reflected in the following four points summarised below: The necessary preprocessing, the determination of the baseline as well as the slope correction, the estimation of the reference lines and the approximation of the dominant slant of the written sample.

For the preprocessing, the majority of the following algorithms has its origin in OCR and is rather based on continuous handwritten samples than the discrete point sets returned by online systems for capturing of handwriting samples. Therefore, to adapt these algorithms the output generated by such systems is preprocessed in our approach by connecting the points using the Bresenham-algorithm [3]. The algorithm

capitalizes on the available online information from the sensor device, specifically the fact that the order of points and their coordinates are known. After this preprocessing, the sample from an online system looks much more like a real continuous writing and is usable for OCR approaches.

The line on which most of the lower ends of written letters align is called the baseline. It allows for a determination of the orientation of the writing direction. Our idea for the computation of the baseline is derived from the approach of Senior and Robinson [2]. In contrast to [2], our algorithm abstains from cutting descenders in the first step, because for our test samples no horizontal slope for the sample can be guaranteed. Instead our first step is to identify the point with the smallest y-coordinate in each column (of the x-y matrix representing the handwritten sample), i.e. for each distinct x-coordinate with points. Through the connection of points performed in the preprocessing, the need for an identification of unrequired points is kept to a minimum. In the worst case, every point of the sample might have been returned, as long as they were in different columns of the x-y matrix. The next step is to compute the local minima. All points except for the local minima will be discarded for the further baseline computation steps. This reduces the set of points to only the ones relevant for the baseline computation.

As starting point for further optimization a first baseline approximation is computed via linear regression on the remaining points. For these, their distance to the regression line is calculated using the Hesse normal form. Depending on whether the points lay above or below the regression line, their distances have either a positive or a negative sign. Given that only local minima were used in the computation of the first approximation of the baseline, it is safe to assume that most of these points lay close to the actual baseline and only a minority possesses a larger deviation. Therefore, we assume that with the first linear regression we already found a suitable first baseline approximation, i.e. we assume that the calculated distances have a zero mean Gaussian distribution. For this sample and the expected value it is possible to calculate the standard deviation so that all necessary parameters for the description of a Gaussian distribution are in place. Usually, the processing steps described above leave only a small number of points for further processing. Thus we approximate the Gaussian distribution with a Student's *t*-distribution. To compensate for outliers in distances before performing another linear regression, we cut off the outer 20 % of the distribution. For the remaining points we compute again a linear regression line, which is the final baseline returned by our approach. Here, it has to be mentioned that the baseline is already conceptually considered in [1] and used there for a preprocessing of the evaluation samples.

Finding the baseline allows for the extraction of the first seven features describing the appearance of the sample: (1) slope of the baseline, (2) angle between baseline and x-axis, (3) y-coordinate of interception of baseline and y-axis, (4) x- and (5) y-coordinate of the left interception point of baseline and bounding box, (6) x- and (7) y-coordinate of the right interception point of baseline and bounding box. After the extraction of these features, all points of the sample are rotated using the angle between baseline and x-axis to create a sample aligned parallel with the x-axis.

When learning to write in a language using the Latin alphabet in school, often special paper with four helper lines is used to help the pupils to keep the letter sizes relatively constant during the writing process. These four lines are called reference lines and even without them preprinted on a sheet of paper handwriting in languages using the Latin alphabet usually aligns itself on these (trained) lines. Our approach to compute the reference lines follows in parts the histogram based approach of Yanikoglu and Sandon [4]. The first step is to determine the histogram h_0 of the horizontal pixel density; one distinct x-coordinate is one distinct class in the histogram. Step two is to smooth the resulting histogram. This is done by accumulating the sum from the two previous classes, the class itself and the two subsequent classes.

This smoothing results in a far more stable computation, since it becomes more infrequent to observe single empty classes, i.e. with the value zeros in the histogram. In contrast to Yanikoglu and Sandon [4] we abstain from doing another smoothing step because the determination of the two inner reference lines is implemented here in a different way and the results using a second smoothing were too unstable in our experiments.

In cursive handwriting it is possible for the dot over an "i" to be at a higher position than the upper end of the word boundary or to even protrude in upper text lines. Therefore, starting from the center of the smoothed histogram s_0, moving upwards and downwards, the first class not equaling null, is the upper respectively lower reference line. As we already determined the baseline in the previous pipeline step it is enough to simply rotate this line using the angle between baseline and x-axis to obtain the rotated baseline.

To approximate the core line, we use an approach similar to the approach for determining the baseline, but instead of identifying column minima we are looking for column maxima and their peak values. Unfortunately, the existence of ascenders in normal handwritten text complicates our approach. Under the assumption that letters without ascenders are more common than letters with, the majority of the maxima should be somewhere around the core line. That is why we use a sliding window of size of 15 % of the sample height (after correcting the slope) to identify the area with the majority of local maxima. For the points within this window we calculate the arithmetic mean, which is assumed to be the y-coordinate of the core line. With this all reference lines are in place and the following features can be extracted: (8) y-coordinate of the rotated baseline, (9) y-coordinate of the core line, (10) the height of ascenders, e.g. the distance between the upper reference line and the core line, (11) the height of the letter core, e.g. the distance between the core line and the baseline, (12) the height of descenders, e.g. the distance between baseline and the lower reference line.

Letters in words in cursive handwriting have the characteristic of displaying a certain slant. This slant usually lies somewhere between 20° and 130° [5], depending on whether the characters are more slanted to the left or the right. To determine the dominant slant, a histogram, describing the distribution of angles between a line through two consecutive points and the x-axis, is computed. The class width for this histogram

is defined by us to be 1 degree. All angles not contained in the interval [20°, 130°] are neglected. For the remaining angles the arithmetic mean is calculated and returned as feature (13) - our approximation of the dominant slant of the written sample, which is also the final new feature considered in this paper.

4 Our Attack Approach

Like in [6], which is the basis for our attack considerations, the goal of the performed attack is to create raw data of handwriting which will produce the same Bio-Hash as a specific real handwriting sample. In addition to this potential authentication impact, the attack should be fully automated (in contrast to [6] where user interaction is required) and the created raw data should look like real handwriting – which does not mean to create a handwriting which looks like the original sample, because there is not enough information left in the Bio-Hash to re-create the original.

The targets for our attack are on one hand an extended version of Vielhauers Bio-Hash algorithm (see [7]), and on the other hand our version of this algorithm which is even further extended by the 13 new features described in section 3. Regarding the version that extracts 131 features out of the online writing sample, here our attack concentrates only on the features which are solely positional dependent, meaning features which can be calculated directly from the horizontal (x) and vertical (y) positions, plus three features which can be used to compute the bounding box of the writing. This reduces the number of features considered in this version of the Bio-Hash algorithm to 83 (out of 131).

4.1 Requirements of the Attack

The attack has certain requirements: In order to create raw data, which will produce the same Bio-Hash as a specific real handwriting, some information about the chosen real handwriting is necessary (see [6]). These required inputs to the attack are the Interval matrix (see section 2) and the corresponding reference Bio-Hash. This data is used to calculate a Bio-Hash vector representation of the actual synthetic raw data and also to determine the Hamming distance between this Bio-Hash and the reference Bio-Hash. That means the attacker will be able to evaluate how similar the Bio-Hash from the synthetic writing will be in regard to the reference Bio-Hash.

4.2 Algorithm Layout of the Attack

The attack presented here uses an evolutionary algorithm to compute improved synthetic handwritings. It extends the attack presented in [6] by removing the dependence on user input and creating much more natural looking raw handwriting samples.

The initial step for the approach used in this paper is to generate the synthetic handwriting required as input for the evolutionary algorithm. We choose to let a subject write a text and extract the single characters as raw sample data to compose a

dictionary that we call in the following the basic set. With such a basic set it is possible to create own words by simply concatenating individual characters drawn from the basic set. Several rules can be specified for the creation. Here, the following set of basic rules is applied:

1. Scale the character in a way that they have a similar height as the reference
2. Alignment of the characters to their base line with respect to their position on the y-axis
3. Use a fixed distance between the characters with respect to their position on the x-axis
4. Start a word with an upper-case character followed by lower case characters
5. Try to create a word with a bounding box as similar as possible to the bounding box derived from the reference Bio-Hash

The advantage of our approach is that it is possible to use the handwriting style of the basic set. Therefore, the newly created synthetic handwriting as input for the evolutionary algorithm is by design looking very natural. It has to be admitted that there are some deviations from this rule. For example the connections between letters can look unnatural since they are created by simply connecting the last point of a letter with a straight line to the first point of his successor.

In an evolutionary algorithm there are two general types of operations to change data: Mutation (1:1 operations) and recombination (n:n operations). A recombination tries to combine the positive features of different individuals together in one. The problem in this case is the computation of the features: Most of them are dependent on every single data point in the raw data, which means if two individuals are combined; the result would not have features partially of one and partially of another individual, but completely new ones. For this reason recombinations are excluded from our work and only mutations are used for our attack approach. Nevertheless, it has to be ensured that the applied mutations are capable of producing results across the whole sample space. This means especially:

- A data point can move by mutations to any location
- A sample can have any number of data points
- There can be any number of pen-ups in a handwriting and they can be everywhere

Additionally, mutations should not impose strong degradations to the naturalness of the original synthetic handwriting. The set of mutation operations used by us is: Point change (controlled by a radius), add or remove pen up, exchange a character (from the basic set) by another, dilation of a handwriting in width and/or height, rotation (of writing or character), changes of the slant (writing or character) and slight movements of a whole character. It has to be admitted that the sample space would only fully covered if the char-change-mutation would be able to create any number of data points, which is not the case in our work. Even so, the sample space is covered enough which is why there is no need for the implementation of add-character or delete-character mutations.

In the evolutionary algorithm, the fitness function for the rating of a generated synthetic writing is the hamming distance to the reference Bio-Hash and as selection method elitism is applied with parameterization of 50%. Thereby, only the best 80 individuals out of a generation of a total of 160 synthetic writing samples are carried over into the next generation.

5 Results

The goal of our evaluations is to determine, whether the new features introduced in section 3 improve the security in regards to the attack introduced in section 4. Due to the fact that the current Matlab implementation of the attack is not optimized for fast processing, the computation time for an attack can take over three days of time, depending on the number of sample points in the writings and the performance of the hardware. Since we only had a limited amount of time to conduct our evaluation; only attacks on seven sample signatures are performed here. Every attack is parameterized with 350 generations to compute the final synthetic writing.

For evaluation purposes we log the fitness of every individual per generation (i.e. the hamming distance to the reference Bio-Hash) and for every feature per generation the number of synthetic handwriting samples which implement the feature incorrectly, i.e. which do not achieve a feature value close enough to the reference Bio-Hash to successfully authenticate this feature.

A first approach to evaluate the usefulness of the new features against the attack presented in this paper is to survey if and how often a feature could be successfully created / implemented in the generated synthetic samples. This is of great interest because if a feature cannot be implemented in a synthetic writing it is very secure. In contrast, if most mutations lead to a correct implementation it would be very insecure. Using the number of incorrect implementations for every feature, a hit-score is computed in every generation. For the list of hit scores computed for a feature, a tendency indicator is created for every generation by subtracting the value for the current generation by its successor. A positive value for this tendency indicator signals an improvement for the new generation. A negative number implies degradation. At the end, out of the computed tendency indicators a global tendency for the feature is computed. Since only the correct implementations are of interest, only positive value tendency indicators are added up. The computed number is not the number of correct implementations of this feature, but gives nevertheless a value to compare the old and the new features in regards to their reachability. The hit-score average for the old features is 287.62 while the average for the new features is 349.40. These values lead to a first impression that the new features are far worse than the old ones. However there is a special case which can distort the basis data, which is when a feature was correctly implemented in every individual of the first generation and the implementation rate never degraded. In such a case the feature was correct in every instance of a synthetic writing, but never added something to the hit-score. To take a closer look onto this problem; these cases are counted for every feature in every of the seven attack simulations, class-divided for old and new features and normalized, i.e. divided

by the number of attack simulations and number of features. The resulting values are 1.08 for the old and 0.62 for the new features. This leads to the conclusion that, although the hit score is far worse for the new features, we cannot conclude that the new features are performing worse than the old ones.

The hit-score is an attempt to compare the two categories: old features and the new features introduced in this paper. However, one could take a single feature and compare it to the rest of the features. In this case two of the new features are very noticeable: While most of the features seem to be reproducible with average difficulty, the third (y-coordinate of interception of baseline and y-axis) and the fifth feature (y-coordinate of the left interception point of baseline and bounding box) create in a considerable proportion of the attack simulations the special case that not a single synthetic writing implemented them correctly. While in case of the fifth feature three out of seven attacked hashes resulted in not a single correct implementation; the third feature showed no correct implantation in five out of seven attacked hashes. Based on our evaluation data no other feature, neither in the old nor in the new set, created such a special case this often. This indicates that these two new features show a strong potential to increase the attack resilience of the Bio-Hash algorithm.

6 Summary and Future Work

Our evaluations show, that like the old feature set described in [7], every new feature can be implemented correctly in a synthetic writing created with our attack. However, we cannot say that the older features are more secure than the new ones in regards to the presented attack, or otherwise. Nevertheless, it was shown, that two features of the new ones seem to be exceptionally secure. Further tests should be done to support or reject this observation and to evaluate what makes these features more secure than the rest.

In regards to the outcomes of the attacks: The synthetic handwriting as output of the attack looks for the most part very naturally. An example for such a synthetic writing is shown in Figure 1. Nevertheless, it has to be admitted that in some cases unnatural characteristics, e.g. a straight linking line between two letters, are created.

Fig. 1. Examples for real (left) and artificially created handwriting - in the middle an output of the attack tool from [6] and right an output of the attack introduced here

Regarding open issues for future work, the first step should be the extension of the evaluation set used. The number of seven sample signatures used as attack target here are suitable for first indications on the performance of the new features and attack, but statistically significant evaluations would require larger variance.

Also, a wider variety in the basic set used as input for the evolutionary algorithm (and therefore a more diverse starting point for the attack) should be considered. This could be achieved by using handwriting samples from different persons in contrast to only one person used in this paper. A drawback might be that the output artificial writing might show a much stronger divergence regarding different instances of the same latter and would therefore lose some of its natural look.

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. We also like to thank Prof. Jana Dittmann of the Otto-von-Guericke University Magdeburg and the StepOver GmbH for supporting the project "OptiBioHashEmbedded".

References

1. Vielhauer, C.: Biometric User Authentication for IT Security: From Fundamentals to Handwriting. Spinger, New York (2006)
2. Senior, A.W., Robinson, A.J.: An Off-Line Cursive Handwriting Recognition System. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 309–321 (1998)
3. Bresenham, J.: Algorithm for computer control of a digital plotter. IBM Systems Journal 4 (1965)
4. Yanikoglu, B., Sanson, P.: Off-Line Cursive Handwriting Recognition Using Style Parameters, Dartmouth College, Computer Science, Version: 1993 (PCS-TR93-192) (1993), `http://www.cs.dartmouth.edu/reports/TR93-192.ps.Z`
5. Koppenhaver, K.: Forensic Document Examination: Principles and Practices. Humana Press (2007)
6. Kümmel, K., Vielhauer, C., Scheidat, T., Franke, D., Dittmann, J.: Handwriting Biometric Hash Attack: A Genetic Algorithm with User Interaction for Raw Data Reconstruction. In: De Decker, B., Schaumüller-Bichl, I. (eds.) CMS 2010. LNCS, vol. 6109, pp. 178–190. Springer, Heidelberg (2010)
7. Makrushin, A., Scheidat, T., Vielhauer, C.: Improving reliability of biometric hash generation through the selection of dynamic handwriting features. In: Shi, Y.Q., Katzenbeisser, S. (eds.) Transactions on DHMS VIII. LNCS, vol. 7228, pp. 19–41. Springer, Heidelberg (2012)