A New Forgery Scenario Based on Regaining Dynamics of Signature

Jean Hennebert, Renato Loeffel, Andreas Humm, and Rolf Ingold

Université de Fribourg, Boulevard de Pérolles 90, 1700 Fribourg, Switzerland

Abstract. We present in this paper a new forgery scenario for dynamic signature verification systems. In this scenario, we assume that the forger has got access to a static version of the genuine signature, is using a dedicated software to automatically recover dynamics of the signature and is using these regained signatures to break the verification system. We also show that automated procedures can be built to regain signature dynamics, making some simple assumptions on how signatures are performed. We finally report on the evaluation of these procedures on the MCYT-100 signature database on which regained versions of the signatures are generated. This set of regained signatures is used to evaluate the rejection performance of a baseline dynamic signature verification system. Results show that the regained forgeries generate much more false acceptation in comparison to the random and low-force forgeries available in the MCYT-100 database. These results clearly show that such kind of forgery attacks can potentially represent a critical security breach for signature verification systems.

1 Introduction

One of the main advantage of biometric systems lies in the fact that the user does not have anymore to remember passwords or keep all his different access keys. Another important advantage lies in the difficulty to steal, imitate or generate genuine biometrics data, leading to enhanced security. The work described in this paper is challenging this last statement in the framework of dynamic signature verification systems.

Many signature verification systems have been developed in the past [1] [2]. Signature verification has the advantage of a very high user acceptance because people are used to sign in their daily life. Signature verification systems are said to be static (off-line) or dynamic (on-line). Static verification systems use a static digitalized image of the signature. Dynamic signature verification (DSV) systems use the dynamics of the signature including coordinates, pressure and sometimes angle of the pen as a function of time. Thanks to the extra information included in the time evolution of these features, dynamic systems are usually ranked as more accurate and more difficult to attack than static verification systems.

Signature verification systems are evaluated by analyzing their accuracy to accept genuine signatures and to reject forgeries. When considering forgeries, four categories can be defined from the lowest level of attack to the highest (as presented in [3] [4], and extended in [5]).

S.-W. Lee and S.Z. Li (Eds.): ICB 2007, LNCS 4642, pp. 366–375, 2007.

- **Random forgeries**. These forgeries are simulated by using signature samples from other users as input to a specific user model. This category actually does not denote intentional forgeries, but rather accidental accesses by non-malicious users.
- **Blind forgeries**. These forgeries are signature samples generated by intentional impostors having access to a descriptive or textual knowledge of the original signature.
- **Low-force forgeries**. The impostor has here access to a visual static image of the original signature. There are then two ways to generate the forgeries. In the first way, the forger can use a blueprint to help himself copy the signature, leading to low-force **blueprint** forgeries. In the second way, the forger can train to imitate the signature, with or without a blueprint, for a limited or unlimited amount of time. The forger then generate the imitated signature, without the help of the blueprint and potentially after some time after training, leading to low-force **trained** forgeries. The so-called skilled forgeries provided with the MCYT-100 database [6] correspond here to low-force trained forgeries.
- **Brute-force forgeries**. The forger has access to a visual static image and to the whole writing process, therefore including the handwriting dynamics. The forger can analyze the writing process in the presence of the original writer or through a video-recording or also through a captured on-line version of the genuine signature. This last case is realized when genuine signature data can be intercepted, for example when the user is accessing the DSV system. In a similar way as in the previous category, the forger can then generate two types of forgeries. Brute-force **blueprint** forgeries are generated by projecting on the acquisition area a real-time pointer that the forger then needs to follow. Brute-force **trained** forgeries are produced by the forger after a training period where he or she can use dedicated tools to analyze and train to reproduce the genuine signature. In [4] [3] and [5], tools for training to perform brute-force forgeries are presented.

We report in this article on a new type of signature forgeries that we call **regained forgeries**. The forgery scenario makes three assumptions. First, the forger has got access to one or more versions of a static genuine signature. This assumption is reasonable as it is nowadays usual for us to provide static signatures to third-parties. For example, we are signing credit card receipt almost on a daily basis. Second, the forger is using a dedicated software that automatically allows to analyse the static version of the signature and to *regain* one or more versions of its dynamics. Third, the forger has a way to stream the regained dynamic version into the verification system. This can be done if, for example, the forger has access to the acquisition tablet[1].

Our primary objective in this work is therefore to assess if a procedure can be put in place to automatically compute reliable estimates of signature

[1] Alternatively to the third assumption, the forger could also train himself to imitate the recovered dynamics in a similar way as for the brute-force forgery scenario described earlier.

dynamics starting from a static version of it. A secondary objective is to measure the impact of such forgeries on a state-of-the-art DSV systems. Finally, if such forgeries reveals efficient in breaking DSV systems, the open question is how can we improve DSV systems to diminish the potential risk of such forgeries.

Section 2 introduces the procedure that was crafted to automatically compute sequence of strokes. In Section 3, we present the procedure used to estimate local pen speed. In section 4, the procedure used to estimate the pen pressure is described. In Section 5, experiments performed using the regain procedure are reported using the MCYT-100 database. Finally, conclusions are drawn in the last section.

2 Problem Description

Our objective is to design a procedure to regain the dynamics of a signature given its static representation. Figure 1 illustrates the regain procedure with, on the right-hand part of the figure, a typical example of the signature information that needs to be recovered. A dynamic signature sample is actually a time sequence of points that includes the (x/y) coordinates and the pressure of the pen measured at a given frequency, typically 100 Hz. The time sequence may also include the pen azimuth and elevation angles that are provided with some high-end graphical tablets. In this paper, we focus on the recovery of pen coordinates and pressure. The recovery of pen angles is not addressed.

More formally, the regain procedure is defining a mapping

$$I \stackrel{regain}{\longrightarrow} (\ x,\ y,\ p\ )(\ t\ ) \tag{1}$$

where I is a scanned version of the signature, $x(t)$ and $y(t)$ correspond to the pen trajectory and $p(t)$ is the evolution of the pen pressure. As these information are sampled at a regular time interval, the distance between each point also defines the instantaneous pen speed that also needs to be recovered. The gray block areas in Figure 1 corresponds to the so-called *pens-up* that appear in the signal when the hand jumps from one stroke to another. These pens-up are usually recorded by the acquisition device provided that the pen remains in a

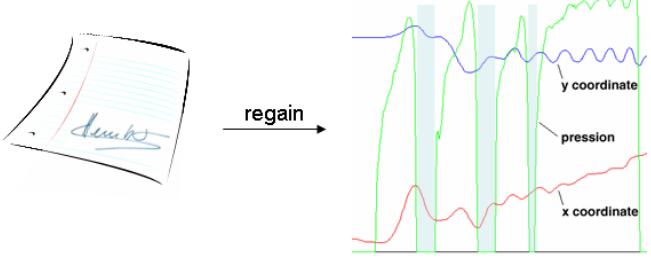


Fig. 1. The regain procedure aims at estimating pen dynamics by analyzing a static version of a signature

given range above the tablet. The corresponding samples have then zero pressure values and valid coordinates[2]. Obviously, such pens-up sequences are not visible in the static version and will therefore need to be re-created.

Of course, the regain mapping of equation 1 has no unique solution as we have no information about the ordering of coordinates and evolution of pressure. The only thing we know for sure is that the the coordinates of the signature pixel in the image must appear at some point of the dynamic signature. On the other hand, not all the solutions are equally probable if we can make some assumptions on how signatures are produced by humans.

3 Regaining the Drawing Order

Our first objective is here to build a procedure that will reorder the set of visible signature points in the input image. In other words, this operation needs to discover, for each black pixel $I(x, y)$ of the image, its corresponding time index i to re-create an ordered sequence $O = [(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_N, y_N)]$. We propose a procedure in three steps: segments detection, segments merging and segments ordering.

3.1 Segments Detection

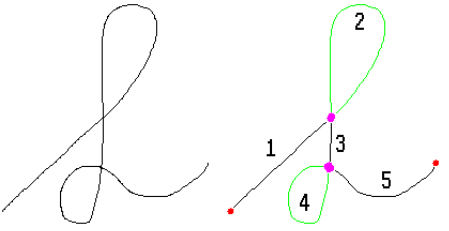
In this step, we exploit the fact that adjacent signature points of the input image are drawn in consecutive order. The order of the pixels can be determined using a simple neighborhood function applied to each pixel. In the example of Fig. 2, 5 segments are detected. The segments extremities are defined either when the neighborhood function returns no more pixels (for example left hand side of segment 1 in 2(b), or when it returns more than one pixel (crossing points of segments 1, 2 and 3 in 2(b). At this stage, we have a set of K segments S_k that are composed of ordered sets of pixels. We do not know yet if the pixels are ordered in the right direction in a given segment, but at least, an order is existing.

3.2 Segments Merging

In order to reduce further the number of segments, we apply a continuity criterion to detect segments that can be merged together. This criterion simply states that whenever a crossing point is reached, it is very likely that the drawing will continue in a similar direction. We propose to use the following continuity measure:

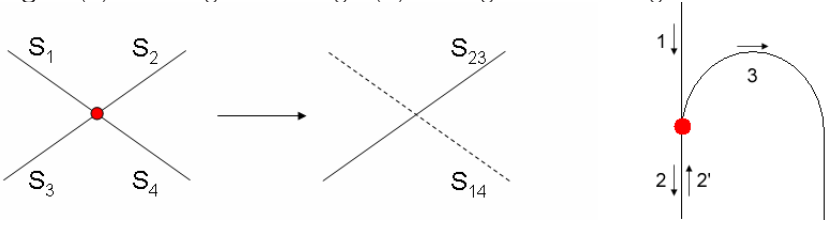
$$c(S_k, S_l) = \begin{cases} 0.1 * \cos(\alpha) \text{ if } \alpha < \frac{\Pi}{2} \\ \left(\frac{2*\alpha}{\Pi} - 1\right)^2 \text{ if } \frac{\Pi}{2} \leq \alpha \leq \Pi \end{cases} \quad (2)$$

[2] Some devices may not record such pens-up but we assume for our approach that they need to be also recovered by the regain procedure.



(a) (b)

Fig. 2. (a) Static signature image. (b) The signature after segment detection.



(a) (b)

Fig. 3. (a) Procedure to merge segments. According to the computation of the continuity measure, segments S_1 and S_4 are merged, as well as segments S_2 and S_3. (b) Special case of a crossing point having 3 incoming segments.

where α denotes the angle between segment S_k and S_l. The angle α is estimated from the two straight lines tangent of both segments at the crossing point. The closer the angle to 180 degrees, the more likely both segments are the continuation of each other. As illustrated on Fig. 3(a), the continuity measure c is computed for each crossing point of the signature, i.e. where segment extremities are coinciding. The decision to merge two segments S_k and S_l into a unique one is then taken if the value $c(S_k, S_l)$ higher than a given threshold.

A special treatment is performed in the case of crossing points with 3 incoming segments, such as illustrated on Fig. 3(b). In this example, the values of $c(S_1, S_2)$ and $c(S_2, S_3)$ are above the merging threshold but no decision can be taken as the number of segment is odd. In such case, it is likely that the segment that is common in the best two values of the continuity measure is actually double traced by the user (S_2 in our example). In this case, our procedure artificially duplicates the concerned segment S_2 into a new segment S_2' where the pixel order is reversed. The merging procedure can then continue.

3.3 Segments Ordering

The reduced set of segments needs now to be ordered. We have to determine the most likely sequence of segments but also the ordering direction of pixels within each segment. Our proposal is here to compute a likelihood measure for a given sequence by taking into account some reasonable assumptions on the signature production process. We can then compute this likelihood measure for all possible

sequence of segments and elect the best one. The procedure also allows to elect the n best sequences of segments if required. Assuming a sequence of segments $(S_1, \ldots, S_K)$, the likelihood of the sequence is computed with

$$L(S_1, \ldots, S_K) = is(S_1)sd(S_K) \prod_{i=1}^{K-1} em(S_i, S_{i+1})wd(S_i, S_{i+1})sd(S_i) \quad (3)$$

where we have the following contributing terms:

Energy minimization. This term is introduced according to the assumption that humans are minimizing the energy spent while producing signatures. If the ending point of one segment is close to the starting point of another, then it is likely that these segments follow each other:

$$em(S_i, S_j) = \max\{1 - \sigma d(S_i, S_j), 0\} \quad (4)$$

where $d(S_i, S_j)$ is the euclidian distance between the ending point of S_i and the starting point of S_j and σ is a parameter used to control the severity of the energy criterion. In our configuration, σ has been experimentally set by visually analyzing the output of the system.

Writing direction. In western civilizations it is common to write from left to right and from the top to the bottom. More likelihood is then given to consecutive segments that are ordered according to this rule:

$$wd(S_i, S_j) = 0.5 + 0.5\cos(\alpha - \frac{\Pi}{4.0}) \quad (5)$$

where α denotes the angle between the horizontal axis and the direction vector formed by the ending point of S_i and starting point of S_j.

Intra segment direction. We assume here that segments drawn from left to right and from top to bottom are more likely:

$$sd(S_i) = 0.5 + 0.5\cos(\beta + \frac{\Pi}{4.0})d(S_i) \quad (6)$$

where $d(S_i)$ is the distance between the starting point and the ending point of segment S_i and β the angle of the vector defined by these points.

Initial segment. Segments that are close to the origin of the coordinate system are more likely:

$$is(S_1) = 0.5 + 0.5\max\left(1 - \frac{d(S_1, o)}{d(S, o)}, -1\right) \quad (7)$$

where $d(S_1, o)$ represents the distance between the starting point of segment S_1 and the origin and $d(S, o)$ is the average distance between all segment starting points and the origin.

3.4 Invisible Segment Recovery

In this step we estimate the invisible sequences of point that are recorded between a pen-up and a pen-down. We have chosen to implement a simple linear interpolation between the pen-up and the pen-down. This is rather an inaccurate estimation, but as the amount of invisible points is much lower than for visible points, this estimation is probably sufficient as a first approach. A possible improvement would be to implement an non linear interpolation that preserves realistic values of the inertia of the pen.

4 Velocity Recovery

Thanks to the previous steps, we now have a unique segment composed of a likely ordered sequence of visible and invisible signature points. The next step is to simulate back the 100 Hz sampling procedure of the tablet, taking into account the effect of the velocity of the pen that may vary between each point. We analyzed the velocity information of several dynamic signatures and we could make the following observations. First, we observed that the speed of the pen remains relatively stable but presents local variations. When observing these local variations, we noticed that the longer and the straighter the segments, the faster they were drawn. The velocity recovery procedure that we propose here is directly inspired from these observation.

We first estimate the average speed of the regained signature looking up the speech value in a table that was built by inspecting 50 signatures taken from the MCYT database. The table stores the average speed values according to the spatial length of the signature. The number of signature points recovered as explained in the previous sections allows us to compute the spatial length of the signature from which we can look up the average speed from the table.

In a second step, the sequence of points of the static signature is divided into smaller sub-sequences S_k that are delimited by regions of abrupt change of direction. An instantaneous speed $s_l(S_k)$ is associated to each of these sub-sequences according to

$$s_l(S_k) = s_a f_l(S_k) f_s(S_k) \tag{8}$$

where s_a is the average speed, $f_l(S_k)$ is a multiplying factor which is dependent to the length of segment S_k and where $f_s(S_k)$ is a factor which is dependent to the straightness of the segment S_k. Without entering into too many details, we implemented these factors with sigmoid-like functions where the parameters were tuned to provide realistic values of the instantaneous speed. The sequence of signature points can then be non-linearly downsampled according to the instantaneous speed values and a natural evolution of the pen velocity can be recovered.

5 Regaining the Pressure

The last value that needs to be recovered for our purpose is the pressure value. We used here a very simple step function where every visible signature point

obtains a pressure value that corresponds to the maximum pressure value and where invisible points obtain a pressure value of 0. This is a rather crude approximation of the actual pressure value, but sufficient in a first approach. We could easily refine this procedure by ramping up and down the passage from invisible to visible points or by introducing a correlation between the speed and the pressure. However, we leave this for future work.

6 Database, DSV System Description and Experiments

Experiments have been done with online signatures of the public MCYT-100 database [6]. This mono-session database contains signatures of 100 users. For each user, 25 genuine signatures and 25 low-force trained forgeries are available[3]. These forgeries are produced by 5 other users by observing the static images and training to copy them. We used the 50 first signatures of the database as development data on which the different parameters of the regain procedure were tuned. The 50 last signatures of the database were used for the evaluation. As static versions of signatures are not available in MCYT, we generated static signature images from the dynamic data. This procedure may be considered as optimistic as the static versions of signatures are not including additional noise or distortion coming from a scanning procedure. However, we can reasonably assume that an intentional impostor could easily arrive to similar image quality by using widely available image processing tools to thin the segments and to remove noise from a scanned signature.

We have chosen to measure the impact of our regained signatures on a state-of-the-art DSV system based on local feature extraction and Gaussian Mixture Models (GMMs). The system and the evaluation procedure is here very much similar to what is described in [7]. For the feature extraction, x, y, pressure, trajectory tangent angles $\left(\arctan\frac{\dot{y}(t)}{\dot{x}(t)}\right)$ and instantaneous displacements $\left(\sqrt{\dot{x}(t)^2+\dot{y}(t)^2}\right)$ are used. The features are mean and standard deviation normalized on a per signature basis. For the modelling with GMMs, we make the hypothesis that the features are uncorrelated so that diagonal covariance matrices can be used. We normalize the client score using a world model trained by pooling the data of many other users. The training of the client and world models is here performed with the Expectation-Maximization (EM) algorithm [8]. The client and world model are trained independently by applying iteratively the EM procedure until convergence is reached, typically after few iterations. In our setting, we apply a simple binary splitting procedure to increase the number of gaussian mixtures to the predefined value. For each user, we trained a GMM composed of 64 Gaussian mixtures using the data of 5 signatures taken out from the 25 available genuine signatures. Additionally to the user models, we trained a world model comprising 1250 genuine signatures taken from the first 50 users.

For each users, the remaining 20 signatures were used for genuine tests. We performed three different types of forgeries: random, low-force and regained.

[3] These forgeries are named as *skilled* forgeries in the distribution of the database.

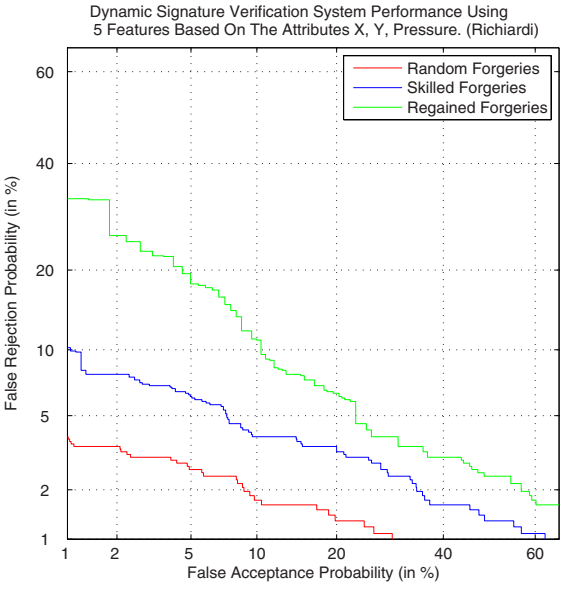


Fig. 4. Evaluation of regained forgeries on the MCYT-100 database

Random forgeries were performed using one genuine signature randomly selected from each remaining user. The low-force forgeries are performed using the 25 so-called *skilled* forgery signatures supplied with the MCYT database. For the regained forgeries, we took the 5 most probable signature versions found during the regain process. If less versions were available only these were considered. From the 50 signatures composing our test suite, 4 signatures could not be regained with our procedure. The number of segments after the segment merging step described in section 3.2 was simply too high to allow for the computation of the best segment ordering. Fig. 4 shows the detection performance curve for the tree types of forgeries. We can see that regained forgeries are significantly degrading the performance in comparison to random and skilled forgeries. The equal error rate jumps from about 6% for skilled forgeries to more than 10% for regained forgeries.

7 Conclusions and Future Work

We have introduced a new forgery scenario where it is assumed that the forger has got access to a static version of a genuine signature to attack a dynamic verification system. We have shown that automated procedures can be proposed to regain signature dynamics, making some simple assumptions on how signatures are produced by humans. We have finally evaluated these procedures on the MCYT-100 signature database on which regained versions of signatures are successfully generated. This set of regained signatures is used to evaluate the rejection performance of a state-of-the-art dynamic signature verification system.

Results show that the regained forgeries generate much more false acceptation in comparison to the random and low-force forgeries. These results clearly show that such kind of regained forgery attacks can potentially represent a critical security breach for signature verification systems. In potential future work, we would like to investigate more refined recovery procedures, attempting to model more closely the signature production process of humans. Also, an important area of research would be to investigate how DSV systems can be improved in order to diminish the potential risks of such regained forgeries.

Acknowledgments. This work was supported by the EU BioSecure NoE (6th Framework - IST-2002-507634). and by the University of Fribourg.

References

1. Plamondon, R., Lorette, G.: Automatic signature verification and writer identification - the state of the art. Pattern Recognition 22(2), 107–131 (1989)
2. Leclerc, F., Plamondon, R.: Automatic signature verification: the state of the art–1989-1993. Int'l J. Pattern Recog. and Artificial Intelligence 8(3), 643–660 (1994)
3. Vielhauer, C.: Biometric User Authentication for IT Security. Springer, Heidelberg (2006)
4. Zoebisch, F., Vielhauer, C.: A test tool to support brut-force online and offline signature forgery tests on mobile devices. In: Proc. of the IEEE Int'l Conf. on Multimedia and Expo 2003 (ICME), Baltimore, USA, vol. 3, pp. 225–228 (2006)
5. Wahl, A., Hennebert, J., H, A., Ingold, R.: Generation and evaluation of brute-force signature forgeries. In: Gunsel, B., Jain, A.K., Tekalp, A.M., Sankur, B. (eds.) MRCS 2006. LNCS, vol. 4105, pp. 2–9. Springer, Heidelberg (2006)
6. Ortega-Garcia, J., et al.: Mcyt baseline corpus: a bimodal biometric database. IEE Proc.-Vis. Image Signal Process. 150(6), 395–401 (2003)
7. Richiardi, J., Drygajlo, A.: GMMs for on-line signature verification. In: Proc. 2003 ACM SIGMM workshop on Biometrics methods and applic., pp. 115–122. ACM Press, New York (2003)
8. Dempster, A., Laird, N., D.B., R.: Maximum likelihood from incomplete data via the em algorithm. Journal of Royal Statistical Society 39(1), 1–38 (1977)