Biometric user authentication on smart cards by means of handwritten signatures

Olaf Henniger[a], Katrin Franke[b]
[a] Fraunhofer Institute for Secure Telecooperation
Rheinstr. 75, 64295 Darmstadt, Germany
henniger@sit.fraunhofer.de
[b] Fraunhofer Institute for Production Systems and Design Technology
Pascalstr. 8–9, 10587 Berlin, Germany
franke@ipk.fraunhofer.de

Abstract. This paper describes a biometric method for user authentication on smart cards. Smart cards are chip cards with the ability for data processing directly on the card. They are not only usable for storing biometric reference data, but biometric user authentication methods can also be performed on card in order to protect security-relevant functions or data on the cards. The biometric data under consideration are handwritten signatures captured by means of a graphic tablet and a special pen. The feature-matching algorithm is a variant of dynamic time warping, taking the limited resources of smart cards into account. It is implemented as an operating prototype on two types of smart cards.

1 Introduction

Smart cards are chip cards with an integrated microprocessor chip. For smart cards providing security-relevant functions (like the creation of electronic signatures) or carrying values (like an electronic purse) or sensitive data (like medical data), the verification that the person who wants to use the card is the legitimate card holder should take place inside the card itself. On-card matching prevents the feigning of a positive result to the smart card. Moreover, on-card matching effects that the sensitive reference data remain safely stored inside the smart card. The function protected by on-card matching can only be used after successful user authentication. Figure 1 illustrates the division of labor between the smart card service system and the smart card in case of on-card matching. On-card matching implementations are available for fingerprint biometrics, but so far not for handwritten signatures.

The major advantage of handwritten signatures over other biometric features is their high level of user acceptance. Handwritten signatures have long been accepted in many places as a means for authenticating persons. Moreover, a handwritten signature is regarded as evidence of a deliberate decision on the part of the signer because, in general, a signature is not delivered coincidentally or unintentionally.

There is a multitude of methods for analyzing on-line signatures, i.e. handwritten signatures captured by means of a graphic tablet and/or a special pen [1, 2]. Since smart cards possess only a limited memory capacity and their computing power falls far short of that of PC’s, algorithms for on-card matching must be chosen carefully and must be adapted to these conditions. In this paper, we discuss the selection of a signature recognition method that is suitable for implementation on smart cards. In a feasibility study, the selected method has been implemented as an operating prototype on two types of smart cards.

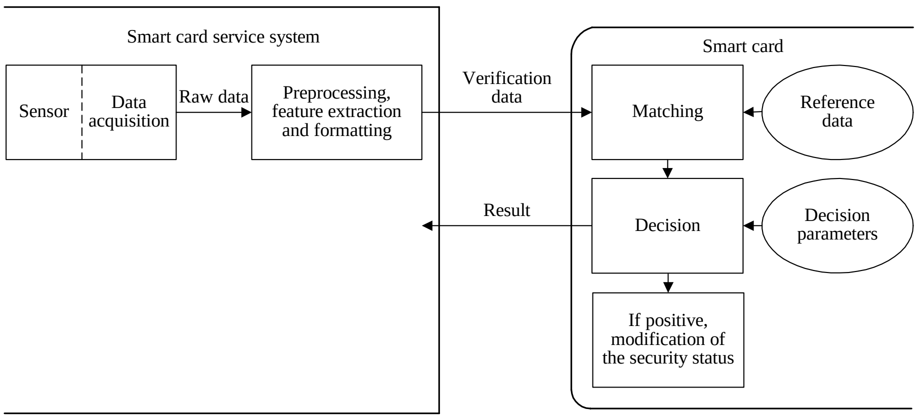
Smart card service system
Sensor
Data acquisition
Raw data
Preprocessing, feature extraction and formatting
Verification data
Result
Smart card
Matching
Reference data
Decision
Decision parameters
If positive, modification of the security status

Figure 1 Biometric user authentication with matching on card

The remainder of the paper is arranged as follows: In Section 2 the side conditions for the on-card matching implementation are described. Section 3 deals with design decisions. Section 4 contains first experimental results obtained with our prototype implementations. Section 5 summarizes the results and gives an outlook.

2 Side conditions

2.1 Capture devices

The on-card matching method should be widely independent of the capture device used for data acquisition, so that the smart card can be used in different environments.

Graphic tablets provide time series for the pen position (*x* and *y* coordinates). Some tablets also provide time series for the pen-tip pressure and the pen orientation, which may support signature matching. However, because not every kind of tablet provides this information, pen-tip pressure and pen orientation are left out from our on-card matching method. Even if different tablets provide time series for the pen-tip pressure, their uniform calibration is difficult.

Since the signature dynamics of a person vary slightly from signature to signature and signatures must be accepted anyway within a certain range of tolerance, for signature capture an approach "as accurate as necessary" and not "as accurate as possible" is needed. We assume a resolution of at least 1,000 dpi and a sample rate of 100 per second to be sufficient not to lose substantial information about on-line signatures. The time series produced should be equidistant, so time stamp information need not be recorded.

The graphic tablets deployed should record the pen position not only if the pen touches the tablet (pen-down strokes), but also if the pen is close-by above the tablet (pen-up strokes) since these movements may bear signer-specific information.

2.2 Implementation platforms

Java cards. Java cards are smart cards with an interpreter (Java Card Virtual Machine) [3] for the execution of processor-independent byte code. Code development for Java cards is based on a subset of Java and Java development tools.

Java cards provide only limited memory space and their computing speed is limited because the Java byte code is interpreted at run-time. On Java cards, only a subset of the Java language is available. Java cards support the data types *Boolean*, *byte*, *short*, and optionally also *int*; the data types *char*, *double*, *float*, and *long* are not available. Only one-dimensional arrays are supported. By default, only basic arithmetic operations and no mathematical libraries are available. There is no garbage collection and no *finalize*(). Objects and arrays once created cannot be deleted; their storage location remains occupied. Therefore, all necessary objects and arrays have to be created at the installation of a Java card applet and be reused later. Dynamic loading of classes is not supported; all necessary classes must be brought onto the Java card at production time or during the installation of a Java card applet.

We selected rather powerful Java cards as implementation platform (CPU word length: 16 bit, 2 Kbytes RAM, 64 Kbytes ROM, 32 Kbytes EEPROM, default clock rate: 3.5712 MHz) [4]. However, their computing power falls far short of that of today's PC's.

The shortage of RAM causes considerable problems. There would be sufficient EEPROM available to use it also as main memory; however, writing into EEPROM cells is substantially slower than writing into RAM, and the number of write cycles for EEPROM is limited.

Java cards are well suited for the rapid development of prototype smart card applications. That's why the signature recognition approach described in this paper has first been implemented on Java cards. However, their drawback is that the interpretation of Java card applets by a Java Card Virtual Machine is rather slow.

Native-code cards. Program execution on smart cards running native code is considerably faster than on Java cards. Therefore, native-code cards would be a better choice for developing an end-product rather than a prototype. However, the memory space on native-code cards is as limited as on Java cards. Code development for native-code cards is based on cross-compiling programs in higher programming languages.

The signature recognition approach described in this paper has also been implemented as an operating prototype on a smart card running native code, viz. on a Funcard 2. The Funcard 2 is available e.g. from hobby electronics shops and is well suited for applications requiring rapid development and small volume roll-outs. It is a programmable multi-chip smart card with an AVR-based RISC microcontroller (Atmel AT90S8515A: CPU word length: 8 bit, 512 bytes RAM, 8 Kbytes programmable flash, 512 bytes EEPROM, default clock rate: 3.5712 MHz) [5] and a separate 8 Kbytes EEPROM chip.

3 Design of the on-card matching method

3.1 Signature recognition approach

Methods for analyzing on-line signatures can be roughly classified into methods for the statistical, spatial, temporal, and spectral analysis and combinations thereof. The statistical analysis is too weak alone. The spatial analysis considers features perceptible in the shape of on-line signatures. For a temporal analysis, the features to be compared are time series, i.e. sequences of values at successive points in time. The spatial or temporal analysis could be implemented on smart cards. For a spectral analysis today's smart cards do not have sufficient resources.

With standardization of the feature data format in mind, it stands to reason to use directly the time series for x and y, possibly supplemented with time series for velocity, acceleration, pen orientation, etc., as biometric features. Therefore, we have chosen the temporal analysis of on-line signatures as on-card matching algorithm.

3.2 Preprocessing, feature extraction, and formatting

Preprocessing the captured on-line-signatures. The goal of preprocessing is to suppress insignificant information that is expression of random variation, but not of individual signature dynamics, and to even out differences between the data captured with different graphic tablets. Preprocessing steps include translation and rotation of the signature, removal of the linear component from the *x*-time regression line, normalization and scaling.

If the time series for the *x* and *y* coordinates had a normal (GAUSSian) distribution, then about 70% of the normalized *x* and *y* coordinates lay in the interval between –1 and 1, almost 100% of them lay in the interval between –4 and 4. Due to the lack of floating-point arithmetic on Java cards, floating-point numbers are not suitable for on-card processing. Therefore, the *x* and *y* coordinates are scaled such that they each can be coded in a signed byte, i.e. as an integer in the interval from –128 to 127. In order that almost all values lie in the desired interval, the normalized *x* and *y* coordinates are scaled by multiplying with $128/4 = 32$ and rounding off towards the nearest integer. In case a value lies outside the desired interval after scaling (which is possible, but very improbable if the values were normally distributed), it is replaced by –128 or 127, respectively. Rounding off to integers entails a quantization error. We assume it to be negligibly small compared to the random variation among signatures of the same person.

Figure 2 (a) shows pen-down and pen-up strokes of a sample signature. Figure 2 (b) shows the preprocessed time series of the *x* and *y* coordinates of the sample signature.

Feature extraction. The features to be compared are the preprocessed time series of the *x* and *y* coordinates and time series for the velocity in *x* and *y* direction derived from the time series of the *x* and *y* coordinates by forming difference sequences. If the derivation of the difference sequences takes less time on the card than their derivation outside the card plus the transmission to the card, then the difference sequences should be derived on card.

Format of the signature feature data at the smart-card interface. Reference data are transmitted to the smart card in the command data fields of CHANGE REFERENCE DATA APDU's; verification data are transmitted in the command data fields of VERIFY APDU's. Both, reference data and verification data are required to be packed in a biometric data object [6, 7]. The structure of biometric data objects used in the prototype implementation is described in Table 1.

In general, only 255 bytes can be transmitted in the data field of a smart-card command. Because the feature data may be longer than this, we use command chaining for the APDU's [9] and send the data belonging to a signature in subsequent APDU's.

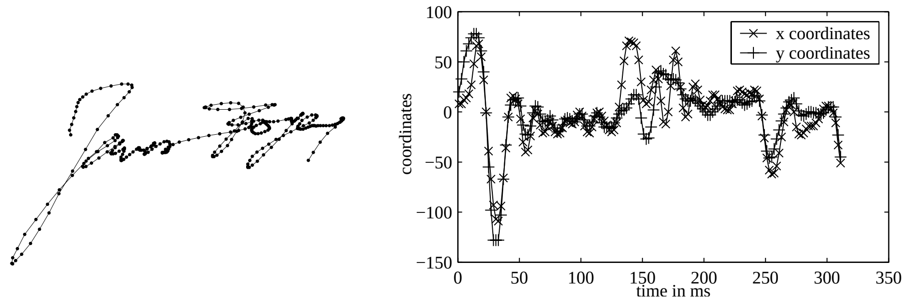

Figure 2 (a) Sampled signature (b) Preprocessed pen-position time series

Table 1 Format of biometric data objects for signature feature data

Tag	Length	Value
5f 2e (Card Holder Biometric Data [6])	1–3 bytes, coded according to the ASN.1 Distinguished Encoding Rules [8]	OCTET STRING $x_1y_1...x_ny_n$

3.3 Feature matching on card

The feature-matching algorithm compares the verification data presented by the user with the reference data stored on the card and determines their “distance” as a measure for their dis-similarity. If the distance is larger than a threshold value, the presented signature is regarded as a forgery; otherwise, it is regarded as a genuine signature. The threshold depends linearly on the length of the reference data. The reference signature is one signature selected from several signatures captured during the enrolment phase.

The “distance” of two signatures can be determined by nonlinear time warping of the time series of the signature to be verified to the time series of the reference signature [2, 10, 11]. Nonlinear time warping is achieved by dynamic programming. The goal is to determine a mapping between the time axes of the signatures to be compared (distortion path) minimizing the distance of the two signatures and observing certain side conditions. The side conditions are that the start and end points of verification and reference data must be mapped one on the other, that the temporal ordering of the sample points must be maintained, and that no sample point be omitted. The distance between verification data and reference data is the sum of the local distances along the optimal distortion path. The local distance between the feature vectors of verification data and reference data (consisting of *x* and *y* coordinates as well as velocities in *x* and *y* direction at a sample point) can be defined in different ways. On the smart card we use the absolute value norm of the difference vector as a distance measure because its computation requires less resources than more sophisticated distance measures.

Dynamic programming algorithms are described in the literature (e.g. [11]) and are not repeated here. Dynamic programming is a time-consuming procedure. The computing time is reduced by looking for the optimal distortion path only within a limited band (Sakoe/Chiba band [12]).

4 Experimental results

For testing purposes, the prototypes reveal the distance values of each matching attempt. In normal operation, the distance values should not be revealed to avoid hill-climbing attacks (where an attacker systematically modifies the verification data to obtain smaller and smaller distances until the decision threshold is met).

The false acceptance rate (FAR, probability that an unauthorized user is falsely accepted) and the false rejection rate (FRR, probability that a legitimate user is falsely rejected) can be estimated experimentally with a certain statistical significance using large test databases [13]. We carried out a preliminary technology evaluation of our prototype implementations by testing against a signature database collected within our research group. The signatures were collected during several sessions from 5 colleagues, all male right-handers under 40 years of age. The signature data were captured on a WACOM tablet. The database contains a total of 50 genuine signatures and 69 forgery attempts. For a statistically more significant technology evaluation, a larger database will be needed.

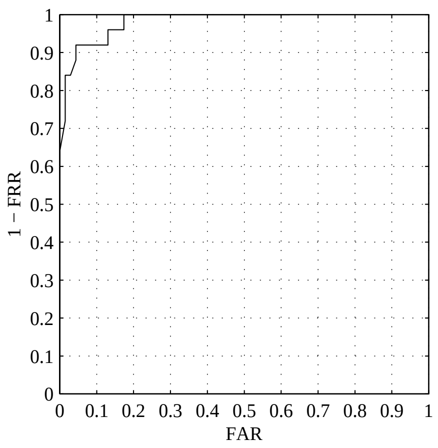

Figure 3 Receiver operating characteristic curve

For the forgery attempts, the impostors had the original signatures to be forged available on paper (a quite realistic scenario) and were allowed to produce forgeries to the best of their ability. They were allowed to practice the signatures to be forged, to look at the original while forging, and even to retrace the original.

Figure 3 summarizes the preliminary performance results in a receiver operating characteristic curve [13] plotting the relative frequency of correct acceptances over the relative frequency of false acceptances at varying threshold values. The equal error rate (where FRR equals FAR) lies at about 8%.

The time needed for transmitting verification data to the card and for on-card matching grows linearly with the size of the verification data. For the shortest on-line signature in the database consisting of 70 sample points (i.e. writing it took 0.7 seconds), the computing time is about 7 seconds on the Java card and about 2.5 seconds on the native-code card. For an on-line signature consisting of 250 sample points, the computing time is about 25 seconds on the Java card and about 8.5 seconds on the native-code card. On the native-code card computing takes about one third of the time it takes on the Java card. For practical applications, less waiting time would be desirable.

5 Summary and outlook

Biometric user authentication on smart cards by means of handwritten signatures is feasible. We have implemented first prototypes of an on-card matching algorithm for handwritten signatures. The challenge consisted in implementing the feature-matching algorithm using only the limited resources available on smart cards.

The approach we selected for implementation on smart cards is the temporal analysis of signatures. The features to be compared are the time series of the *x* and *y* coordinates, which are captured by a graphic tablet and a special pen and pass through some preprocessing steps, and the time series for the velocity in *x* and *y* direction, which are derived from the time series of the *x* and *y* coordinates. By means of dynamic programming, a nonlinear time warping of the signatures to be compared is performed and their distance is determined as a measure for their dissimilarity.

A more thorough statistic evaluation of the attainable error rates based on larger signature databases is pending. An improvement of the performance is expected from dissecting the signatures into pen-down and pen-up strokes, as discussed in [11]. The recognition performance may be further improved by taking into account more features, like pen orientation. The computing time needed for on-card matching will decrease as smart cards become more and more powerful.

Acknowledgements

This research was supported by the German Federal Ministry of Education and Research within the project "Ascribability of actions in virtual worlds" (http://zavir.sit.fraunhofer.de). The authors are grateful to the other members of the project team for fruitful discussions.

References

[1] R. Plamondon and G. Lorette. Automatic signature verification and writer identification – The state of the art. *Pattern Recognition* 22 (1989), pp. 107–131

[2] C. Schmidt. *On-line Unterschriftenanalyse zur Benutzerverifikation*. RWTH Aachen, Germany, PhD Thesis, 1998, in German

[3] *Java Card 2.1.1 Virtual Machine Specification*. Sun Microsystems, Revision 1.0, May 2000

[4] *Reference manual of the Java card operating system Sm@rtCafé 2.0*. Giesecke & Devrient, edition 12/01, 2001

[5] Atmel 8-bit AVR microcontroller with 8 Kbytes in-system programmable flash – AT90S8515. Datasheet, 2001

[6] Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 6: Interindustry data elements. International Standard ISO/IEC 7816-6

[7] Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 11: Personal verification through biometric methods. International Standard ISO/IEC 7816-11

[8] Information technology – ASN.1 encoding rules – Part 1: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). International Standard ISO/IEC 8825-1

[9] Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange. International Standard ISO/IEC 7816-4

[10] J.B. Kruskal. An overview of sequence comparison, In D. Sankoff, J.B. Kruskal (eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, 1983, pp. 1–44

[11] B. Wirtz. *Segmentorientierte Analyse und nichtlineare Auswertung für die dynamische Unterschriftsverifikation*. TU Munich, Germany, PhD Thesis, 1998, in German

[12] H. Sakoe and S. Chiba. Dynamic programming optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech and Signal Processing* 26 (1980), pp. 623–625

[13] A.T. Mansfield and J.L. Wayman. *Best practices in testing and reporting performance of biometric devices*. Version 2.0, Aug. 2002