Implementation of the USB Token System for Fingerprint Verification

Daesung Moon, Youn Hee Gil, Sung Bum Pan, and Yongwha Chung

Biometrics Technology Research Team, ETRI, Daejeon, Korea
{daesung, yhgil, sbpan, ywchung}@etri.re.kr

Abstract. In the modern electronic world, the authentication of a person is an important task in many areas of day-to-day. Using biometrics to authenticate a person's identity has several advantages over the present practices of Personal Identification Numbers (PINs) and passwords. To gain maximum security in the verification system using biometrics, the computation of the verification as well as the store of the biometric pattern has to take place in the security token(e.g., smart card, USB token). However, there is an open issue of integrating biometrics into the security token because of its limited resources(processing power and memory space). In this paper, we describe our implementation of the USB token system having 206MHz StrongARM CPU, 16MBytes Flash memory, and 1MBytes RAM. Also, we describe a fingerprint verification algorithm that can be executed in the restricted environments. To meet the memory space specification and processing power of the security token, in fingerprint verification algorithm, we develop a data structure, called a multi-resolution accumulator array. Based on experimental results, we confirmed that the RAM requirement of the proposed algorithm is about 16 KBytes, and the Equal Error Rate(EER) is 1.7%. Therefore, our fingerprint verification algorithm can be executed in real-time on the developed USB token without degrading accuracy.

1 Introduction

In the modern electronic world, the authentication of a person is an important task in many areas of day-to-day life such as E-commerce and E-business. Using biometrics to authenticate a person's identity has several advantages over the present practices of Personal Identification Numbers (PINs) and passwords [1-7].

In typical biometric verification systems, the biometric patterns are often stored in a central database. With the central storage of the biometric pattern, there are open issues of misuse of the biometric pattern such as the "Big Brother" problem. To solve these open issues, the database can be decentralized into millions of security token such as smart card, USB token[8-10]. USB token is technologically identical to smart cards, with the exception of the interface to the computer. The smart card requires an additional card reader, whereas the USB token having about the size of a house key,

J. Bigun and T. Gustavsson (Eds.): SCIA 2003, LNCS 2749, pp. 998–1005, 2003.

shown in Fig. 1., is designed to interface with the universal standard bus (USB) ports found on millions of computers and peripheral devices.

Fig. 1. The example of the USB token

Most of the current implementations of USB token using biometrics have a common characteristic that the biometric verification process is solely accomplished out of the USB token. This system is called Store-on-Token because USB token is used only as a storage device to store the biometric pattern. For example, in a fingerprint-based Store-on-Token, the fingerprint pattern stored in USB token needs to be insecurely released into a host PC to be compared with an input fingerprint pattern.

To heighten the security level, the verification operation needs to be performed by in- USB token processor, not the host PC. This system is called Match-on-Token because the verification operation is executed on USB token. Note that standard PCs on which typical biometric verification systems have been executed have 1GHz CPU and 128MBytes memory. On the contrary, the system specification of the USB token that we have developed is 206MHz CPU, 16MBytes Flash memory, and 1MBytes RAM. Therefore, the typical biometric verification algorithms may not be executed on the USB token successfully.

The fingerprint is chosen as the biometrics for verification in this paper. In this paper, we present a minutiae-based fingerprint verification algorithm for the Match-on-Token system that can be executed in real-time on the resource-constrained environments. To meet the processing power and memory space specification of the USB token, we develop a data structure, called a multi-resolution accumulator array, with which the equal amount of memory space is required at each resolution. Based on the experimental results, we confirmed that the memory requirement of the proposed algorithm is about 16KBytes, and the Equal Error Rate(EER) is 1.7%.

The rest of the paper is structured as follows. Section 2 explains structural elements and system specification of our USB token, and Section 3 describes the proposed a fingerprint matching algorithm. The experimental results are given in Section 4. Finally, conclusions are given in Section 5.

2 Match-on-Token

The fingerprint verification system can divide into three parts. Image Pre-Processing, Minutiae Extraction and Minutiae Matching. To assign the verification steps to the USB token or the host PC, we evaluate first the resource requirements of each step. Gil et al.[10] reported that the Pre-processing and Extraction steps cannot be executed on

the resource-constrained environments such as USB token. Thus, we determined the Minutiae Matching step is executed on the USB token. Especially, the Minutiae Matching step is most important for the security.

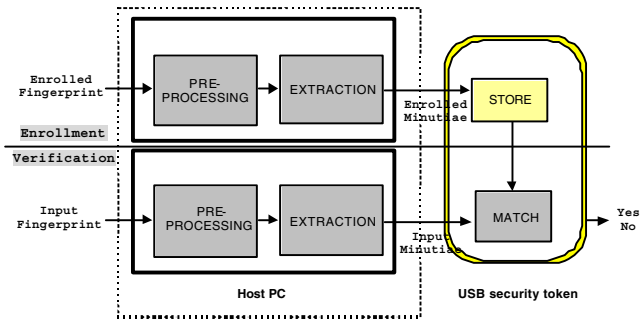

Fig. 2. Fingerprint-based Match-on-Token

Note that the Minutiae Matching step(alignment and matching stages) to compute the similarity between the enrolled minutiae and the input minutiae is executed on the Match-on-Token, whereas the Image Pre-Processing and Minutiae Extraction steps are executed on the host PC. Fig. 2. shows a fingerprint-based Match-on-Token system. In the off-line enrollment phase, an enrolled fingerprint image is preprocessed, and the minutiae are extracted and stored. In the on-line verification phase, the minutiae extracted from an input fingerprint are transferred to the USB token. Then, the similarity between the enrolled minutiae and the input minutiae is examined in the USB token.

Table 1. System Specification of the USB token

CPU	32-bit RISC Processor (StrongARM, 206MHz)
Flash Memory	16 Mbyte
RAM	1 Mbyte
Physical Size	7cm×2cm×1cm

Table 1 shows the system specification of the USB token we developed. The USB token employs 206MHz CPU, 16MBytes Flash memory, and 1MBytes RAM. The size of the USB token is 7cm×2cm×1cm. The reason that we adopt the powerful StrongARM processor[11] is to execute many other applications in real-time such as speaker verification, face verification, and PKI, in addition to fingerprint verification. Though there is sufficient memory space in our USB token, the RAM space available to the fingerprint verification is less than 50KBytes. This is because the 1MBytes RAM memory should be used by Linux Kernel version Embedded Linux 2.4. (717Kbytes) and other applications.

Fig. 3. shows the hardware architecture of the USB token. The processing core of The Intel SA-1110 processor includes the USB end-point interface to communicate between the host PC and the token. Also, the USB token employs the serial port and JATG interface to use in debugging.

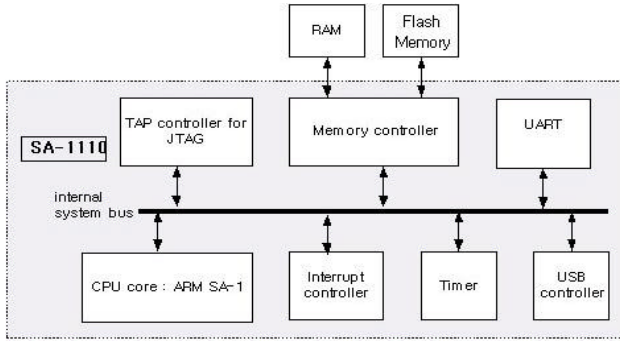

Fig. 3. Architecture of the USB token

3 Fingerprint Verification Algorithm for Match-on-Token

In general, the fingerprint verification system consists of three stages that are the fingerprint image pre-processing, minutiae extraction and minutiae matching. In this paper, we focus on the minutiae matching stage. Fingerprint minutiae extracted from the PC have three members as below,

(1) x and y coordinate of the minutia point
(2) orientation (θ)
(3) type of the minutia point (e.g., ridge ending or ridge bifurcation)

The fingerprint matching stage is composed of two phases: minutiae alignment and point matching. In general, the stored template and the input minutiae cannot be compared directly because of random noise or deformations. The minutiae alignment phase computes the shift and rotation parameters in order to align the two fingerprints. Then, the point matching phase counts the overlapping minutia pairs in the aligned fingerprints. Typically, the minutiae alignment phase requires a lot of memory space and execution time than the point matching phase.

Our alignment algorithm employs an accumulator array in order to compute the shift and rotation parameters[12]. When the two fingerprints are from the same fingerprint, The input to the alignment phase consists of two sets of minutiae points P and Q extracted from fingerprint images[5].

We assume that the second fingerprint image can be obtained by applying a similarity transformation (rotation and translation) to the first image. The second point set Q is then a rotated and translated version of the set P, where points may be shifted by a random noise, some points may be added and some points deleted. The task of fingerprint alignment is to recover this unknown transformation. Since we do not know whether the two fingerprints are the same or not, we try to find the best transformation in the sense.

We discretize the set of all possible transformations, and the matching score is computed for each transformation. The transformation having the maximal matching score is believed to be the correct one. Let's consider a transformation,

$$F_{\theta,\Delta x,\Delta y}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \quad (1)$$

where θ and $(\Delta x, \Delta y)$ are the rotation and translation parameters, respectively. The space of transformation consists of $(\theta, \Delta x, \Delta y)$, where each parameter is discretized into a finite set of values :

$$\theta \in \{\theta_1, ..\theta_L\},\ \Delta x \in \{\Delta x_1, ..\Delta x_M\}, \text{ and } \Delta y \in \{\Delta y_1, ..\Delta y_N\},$$

where *L*, *M* and *N* are the allowable parameters.

Matching scores for the transformations are collected in the accumulator array *A*, where the entry *A(l,m,n)* counts the evidence for the transformation $F_{\theta,\Delta x,\Delta y}$. For each pair (*p*,*q*), where *p* is a point in the set *P* and *q* is a point in the set *Q*, we find all possible transformations that map *p* to *q*. Then, is incremented the evidence for these transformations in the array *A*.

In this straightforward implementation of the accumulator array A, the requirement memory size is *O*(*LMN*). If the numbers of *L*, *M* and *N* are 64, 128 and 128, respectively, the memory size of accumulator array *A* is 1,048,576 bytes. It can not be executed on the Match-on-Card or Security Token. Therefore, we develop a fingerprint matching algorithm using a multi-resolution accumulator array as shown in Fig. 4.

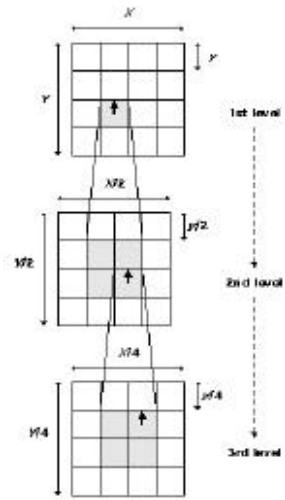

Fig. 4. Computation Flow of our Fingerprint Verification Algorithm

In the first level with the coarse resolution, considering search range ***Y*** and unit size ***y***, find the maximum bin(↑) of the accumulator array is found, that is approximate alignment parameter. To obtain more exact alignment parameter using the same mem-

ory space, our algorithm iterates the same process with the finer resolution than the first level with search range ***Y/2*** and unit size ***y/2*** around the positions found in the first level. Finally, in the third level with the finest resolution, search range ***Y/4*** and unit size ***y/4***, the exact alignment parameter is found.

We can align two fingerprints by selecting the reference minutia pair and getting the difference ($dx, dy, d\theta$) of the selected minutia pair. The notation of $dx, dy, d\theta$ means the difference of the minutia pair, i.e., (x_1, y_1, θ_1) and (x_2, y_2, θ_2). However, it is difficult to select the reference minutiae pair because fingerprints tend to be deformed irregularly. Therefore, all of the possible minutia pairs of two fingerprints have to be considered. The corresponding minutia pairs usually have the similar difference, and this difference can be used as the shift and rotation parameters in order to align the two fingerprints. The accumulator array $\boldsymbol{A}$ is used to find the difference. After computing the difference ($dx, dy, d\theta$) for the minutia pair, each of the entry of array $\boldsymbol{A}(dx, dy, d\theta)$ is increased. At the end of the accumulating processing, the array index with the maximum value is selected as the shift and rotation parameters.

Note that the requirement of memory space of the proposed algorithm is the same at each level. For example, In level 3, the required memory space is 16,384B, (($64/2^{3-1}$)*(128/ 2^{3-1})*(128/ 2^{3-1})). Also, in level 2 and 1, our algorithm uses 16,384B, (($32/2^{2-1}$)*(64/ 2^{2-1})*(64/ 2^{2-1})) and (($16/2^{1-1}$)*(32/ 2^{1-1})*(32/ 2^{1-1})). On the contrary, the memory space of the straightforward implementation requires 1,048,576B. Moreover, the accuracy of our fingerprint verification algorithm is similar to the typical algorithm, because our algorithm takes the unit size 1 in last level.

4 Evaluation of Prototype System

Fig. 5. shows the system environments of our USB token. The input fingerprint image is captured from the fingerprint sensor, and then input minutiae are extracted from input fingerprint image in the host PC. The minutiae extracted from an input fingerprint image are transferred to the USB token. Then, the similarity between the enrolled minutiae that stored in the USB token and the input minutiae is examined in the USB token. Finally, the verification result is transfer again red back to host PC.

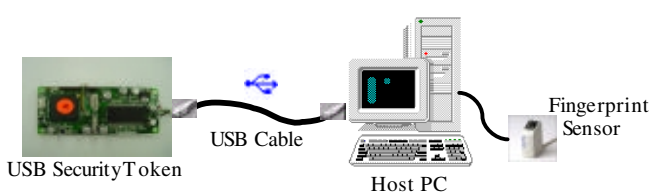

Fig. 5. Configuration of the security token system

We have tested our fingerprint verification algorithm on the fingerprint images captured with an optical scanner manufactured by SecuGen[13], which has resolution of

500dpi. The size of captured fingerprint images was 248×292. The image set is composed of four fingerprint images per one finger from 100 individuals for a total of 400 fingerprint images. When these images were captured, no restrictions on the spatial position and direction of fingers were imposed. Also, the captured fingerprint images vary in quality.

As shown in the Table 2, the required working memory space of the proposed algorithm is about 16KBytes, and the total number of instruction is about 80MBytes. Thus, it is executable in real-time on the USB token. The straightforward implementation requires about 300KBytes RAM and 20Mbytes instruction, respectively. Also, the EER(Equal Error Rate) of the proposed algorithm was almost equal to that of the straightforward algorithm(e.g., 1.7%).

Table 2. The Performance Analysis

	Total No of Instructions	Required Memory Space
Straightforward Algorithm	20M	About 300 KB
Proposed Algorithm	80M	About 16 KB

5 Conclusions

USB token is a model of very secure device, and the biometrics is the promising technology for verification. These two can be combined for many applications to enhance both the security and the convenience. However, typical biometric verification algorithms that have been executed on standard PCs may not be executed in real-time on the resource-constrained environments such as USB token.

In this paper, we have presented a memory-efficient fingerprint verification algorithm that can be executed in real-time on the USB token. To reduce the memory requirement, we employ a small-sized accumulator array. Then, to compute the alignment parameters more accurately, we perform more computations at from a coarse-grain to a fine-grain resolution on the accumulator array. Currently, we are porting memory-efficient speaker and face verification algorithms to the USB token for multi-modal verification.

References

[1] A. Jain, R. Bole, and S. Panakanti,: Biometrics: Personal Identification in Networked Society, Kluwer Academic Publishers, (1999)

[2] L. Jain, et al.,: Intelligent Biometric Techniques in Fingerprint and Face Recognition, CRC Press, (1999)

[3] F. Gamble, L. Frye, and D. Grieser,: Real-time Fingerprint Verification System, Applied Optics, Vol. 31, No. 5, pp. 652-655, (1992)

[4] A. Jain, L. Hong, and R. Bolle,: On-line Fingerprint Verification, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.19, No.4, pp.302-313, (1997)

[5] N. Ratha, K. Karu, and A. Jain,: A Real-Time Matching System for Large Fingerprint Databases, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, August (1996)
[6] S. Lim, and K. Lee, : Efficient Iris Recognition through Improvement of Feature Vector and Classifier. ETRI Journal, Vol. 23, No. 2, (2001)
[7] S. Im, et. al.,: A Direction Based Vascular Pattern Extraction Algorithm for Hand Vascular Pattern Verification , ETRI Journal, Vol. 25, No. 2, (2003)
[8] Kingpin,: Attacks on and Countermeasures for USB Hardware Token Devices, Proceedings of the Fifth Nordic Workshop on Secure IT Systems Encouraging Co-operation, Reykjavik, Iceland, pp 35-57, October 12-13. (2000)
[9] M. Janke, FingerCard Project Presentation, http://www.finger-card.org, (2001)
[10] Y. Gil, et. al.,: Performance Analysis of Smart Card-based Fingerprint Recognition for Secure User Authentication, in Proc. of IFIP on E-commerce, E-business, E-government, pp. 87-96, (2001)
[11] Intel, http://www.intel.com.
[12] S. Pan, et. al.,: A Memory-Efficient Fingerprint Verification Algorithm using A Multi-Resolution Accumulator Array, ETRI Journal, Vol. 25, No. 3, To be published, June (2003)
[13] SecuGen, http://www.secugen.com.