A Memory-Efficient Fingerprint Verification Algorithm Using a Multi-Resolution Accumulator Array

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

Using biometrics to verify a person's identity has several advantages over the present practices of personal identification numbers (PINs) and passwords. At the same time, improvements in VLSI technology have recently led to the introduction of smart cards with 32-bit RISC processors. To gain maximum security in verification systems using biometrics, verification as well as storage of the biometric pattern must be done in the smart card. However, because of the limited resources (processing power and memory space) of the smart card, integrating biometrics into it is still an open challenge. In this paper, we propose a fingerprint verification algorithm using a multi-resolution accumulator array that can be executed in restricted environments such as the smart card. We first evaluate both the number of instructions executed and the memory requirement for each step of a typical fingerprint verification algorithm. We then develop a memory-efficient algorithm for the most memory-consuming step (alignment) using a multi-resolution accumulator array. Our experimental results show that the proposed algorithm can reduce the required memory space by a factor of 40 and can be executed in real time in resource-constrained environments without significantly degrading accuracy.

Manuscript received June 4, 2002; revised Nov. 7, 2002.

This work was supported in part by the Ministry of Information and Communication of Korea.

Sung Bum Pan (phone: +82 42 860 6665, e-mail: sbpan@etri.re.kr), Youn Hee Gil (e-mail: yhgil@etri.re.kr), Daesung Moon (e-mail: daesung@etri.re.kr), Yongwha Chung (e-mail: ywchung@etri.re.kr), and Chee Hang Park (e-mail: chpark@etri.re.kr) are with Information Security Research Division, ETRI, Daejeon, Korea.

I. Introduction

Traditionally, verified users have gained access to secure information systems, buildings, or equipment through multiple means: PINs, passwords, smart cards, and so on. However, these security methods have significant vulnerabilities: they can be lost, stolen, or forgotten. In recent years, there has been an increasing use of biometrics, which refers to personal biological or behavioral characteristics used for verification or identification [1]-[11]. Biometrics relies on "something that you are" and can differentiate between an authorized person and an imposter. The problem of resolving the identity of a person can be categorized by two distinct types. Verification matches a person's claimed identity to his/her previously enrolled pattern: this is a one-to-one comparison. Identification identifies a person from an entire enrolled population by searching a database for a match: this is a one-to-many comparison.

In typical biometric verification systems, the biometric patterns are often stored in a central database. Central storage of biometric patterns leaves the system open to misuse of the biometric patterns: for example, the Big Brother problem. To protect against misuse, the database can be decentralized into millions of smart cards [12]-[22]. Most of the current implementations of this solution accomplish the biometric verification process solely outside of the smart card. This system is called *store-on-card* [18], because the smart card is used only as a storage device to store the biometric pattern. For example, in a store-on-card for fingerprint verification, the

fingerprint pattern stored in the smart card needs to be released to a potentially insecure external card reader to be compared with an input fingerprint pattern.

To heighten security, the verification operation needs to be performed by an in-card processor, not an external card reader. This system is called *match-on-card* [18] because the verification operation is executed on the smart card. Standard PCs on which typical biometric verification systems have been executed have 1 GHz CPUs and 128 Mbytes memory. However, a state-of-the-art smart card may have at most a 50 MHz CPU with 256 kB of ROM, 72 kB of EEPROM, and 8 kB of RAM. For example, a S3CS9PB smart card chip, the latest version released by Samsung [23], is based on the 50 MHz SecurCore SC100, 160 kB of ROM, 64 kB of EEPROM, and 6.5 kB of RAM. This capacity is not sufficient for executing typical biometric verification algorithms.

We chose the fingerprint as the type of biometric verification for our investigation. It is more mature in terms of algorithm availability and feasibility. Fingerprint verification and identification algorithms can be classified into two categories: *image-based* and *minutiae-based* [4]-[9], [14].

Image-based methods include methods involving optical correlation [4], [5] and transform-based features [2], [6]. They may achieve higher computational efficiency than minutiae-based methods. In addition, they may be the only choice when the image quality of the given fingerprint is low. A disadvantage of image-based methods is their limited ability to handle potential variations in position, scale, and orientation angle.

Minutiae-based methods, the most popular matching techniques, are used in most contemporary fingerprint identification and verification systems. The minutiae form a pattern of points, and several well-known point pattern-matching algorithms were proposed in the late ’80s. Jain et al. developed a string-matching technique [7], and Isenor and Zaky proposed a graph-based fingerprint-matching algorithm [8]. Fan et al. described a fingerprint verification algorithm based on a bipartite graph construction between model and query fingerprint feature clusters [9].

Note that most of the previous fingerprint identification and verification algorithms have focused on either accuracy or execution time, because these algorithms were for the automatic fingerprint identification systems or resource-free environments such as PCs. However, these algorithms cannot be applied to a resource-constrained system such as the smart card.

Moon et al. proposed a fingerprint verification algorithm with limited processing power suitable for a smart card [16]. Although some pre-match computation was conducted in a trusted host to reduce the workload of the smart card, there remained an increased potential for corresponding error tolerance. Some additional results of integrating fingerprint identification into a smart card have been reported [19]-[21]. However, no one has yet analyzed the detailed relationship among accuracy, execution time, and required temporary memory space.

In this paper, we present a minutiae-based match-on-card system that can be executed in real time in resource-constrained environments such as the smart card. To meet the processing power and memory space specification of the smart card, we developed a data structure, called a *multi-resolution accumulator array*, for which an 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 for both alignment and matching is about 6.8 kB RAM, and the equal error rate (EER) is 6.0%.

The rest of the paper is structured as follows. Section II explains the technical issues in developing the match-on-card, and section III describes the proposed fingerprint verification algorithm. The experimental results are given in section IV. Finally, conclusions are given in section V.

II. Fingerprint-Based Match-on-Card

1. Fingerprint Verification

A fingerprint verification system (Fig. 1) has two phases: enrollment and verification. 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 similarity between the enrolled minutiae and the input minutiae is examined.

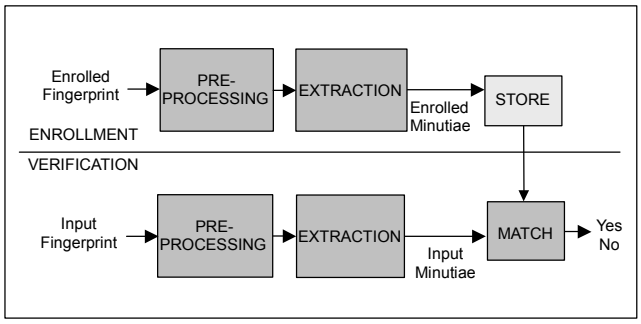

Fig. 1. Fingerprint verification algorithm.

In general, there are three steps involved in the verification process: 1) image preprocessing, 2) minutiae extraction, and 3) minutiae matching.

Image preprocessing refers to the refinement of the fingerprint image against the image distortion obtained from a

fingerprint sensor. It consists of three stages. The *binary conversion stage* applies a low-pass filter to smooth the high frequency regions of the image and threshold to each sub-segment of the image. The *thinning stage* generates a one-pixel wide skeleton image by considering each pixel with its neighbors. In the *positioning stage*, the skeleton is transformed and/or rotated so that valid minutiae information can be extracted.

Minutiae extraction refers to the extraction of features in the fingerprint image. After this step, some minutiae are detected and stored in a pattern file, which includes the position, orientation, and type (ridge ending or bifurcation) of the minutiae.

Based on the minutiae, the input fingerprint is compared with the enrolled fingerprint. Actually, minutiae-matching is composed of an *alignment stage* and a *matching stage*. To match two fingerprints captured with unknown direction and position, the differences of direction and position between the two fingerprints are detected and aligned. This alignment stage estimates transformations, such as translation and rotation, between the two fingerprints and aligns two minutiae according to the estimated parameters. If the alignment is accurate, the following matching stage is a simple point pattern matching. The matching stage compares two minutiae based on their position, orientation, and type. A matching score is then computed.

2. Match-on-Card

Table 1 shows the system specifications of the smart card that we developed for the fingerprint-based match-on-card [13].

Table 1. System specifications of the smart card.

CPU	32-bit RISC Processor (ARM7TDMI)
ROM	64 kB
RAM	8 kB
EEPROM	32 kB

To assign the verification steps to either the smart card or the card reader, we first evaluated the resource requirements of each step. Gil et al. [22] reported that the preprocessing and extraction steps could not be executed in resource-constrained environments such as the smart card, so we assigned only the minutiae-matching step to the smart card.

Note that the minutiae matching step (alignment and matching stages) for computing the similarity between the enrolled minutiae and the input minutiae was executed on the match-on-card, whereas the image preprocessing and minutiae extraction steps were executed on the card reader. Figure 2 shows a fingerprint-based match-on-card 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 smart card. Then, the similarity between the enrolled minutiae and the input minutiae is examined in the smart card. Note that the enrolled minutiae and the input minutiae in Fig. 2 consist of minutiae positions, orientations, and types. In the following section, we focus on the minutiae-matching step.

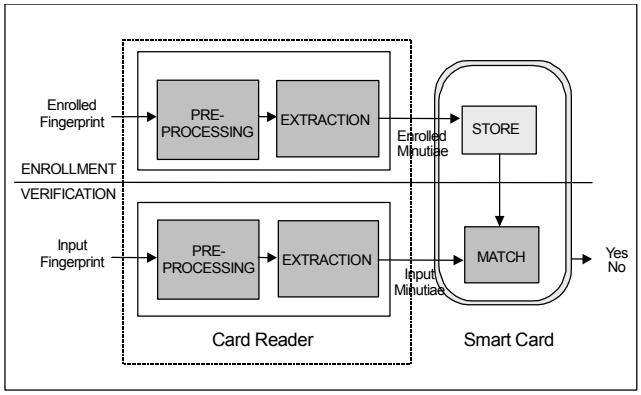

Fig. 2. Fingerprint-based match-on-card.

III. Memory-Efficient Fingerprint Matching Algorithm

1. Baseline Algorithm

The input to the alignment stage consists of two sets of minutiae points P and Q extracted from input and enrolled fingerprint images, respectively. For the purpose of explanation, we define the following notations:

$$P = \{(p_x^1, p_y^1, \alpha^1), \ldots, (p_x^p, p_y^p, \alpha^p)\},$$
$$Q = \{(q_x^1, q_y^1, \beta^1), \ldots, (q_x^Q, q_y^Q, \beta^Q)\}, \quad (1)$$

where (p_x^i, p_y^i, α^i) and (q_x^i, q_y^i, β^i) are the features (spatial position, orientation) associated with the i-th minutia in the set P and Q, respectively. We assume that the second fingerprint image can be obtained by applying a similarity transformation (rotation and translation) to the first image.

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

$$F_{\theta,\Delta x,\Delta y}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}, \quad (2)$$

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 numbers of discretized parameters, i.e., the number of arrays along the rotation and translation parameter axes.

The alignment parameters 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*, all possible transformations that can map *p* to *q* are found. The evidence for these transformations in the array *A* is then incremented.

In this straightforward implementation alignment stage, the required memory space of the accumulator array *A* is *O*(*LMN*). If the numbers of *L*, *M* and *N* are 64, 128 and 128, respectively, the memory space of the accumulator array *A* is 1,048,576 bytes. It cannot be executed on the smart card. Therefore, a different implementation of the accumulator array is needed to reduce the required memory space.

2. Memory-Efficient Algorithm

Figure 3 shows the proposed memory-efficient algorithm using a multi-resolution accumulator array. The following parameters and terminology are defined to describe the computation flow of our algorithm using the accumulator array:

- *d*: the depth of a level
- *ua*: the unit reference angle of the accumulator array at each level
- *ux*: the unit reference distance along the horizontal axis of the accumulator array at each level
- *uy*: the unit reference distance along the vertical axis of the accumulator array at each level
- maximum bin (↑): the most accumulated position in the accumulator array

For simplicity, we assumed that depth is set to 3 and the accumulator array considers the spatial parameters *ux* and *uy* only. In the 3rd level, with coarse resolution considering with range = *Y* and *uy* = *y*, the maximum bin of the accumulator

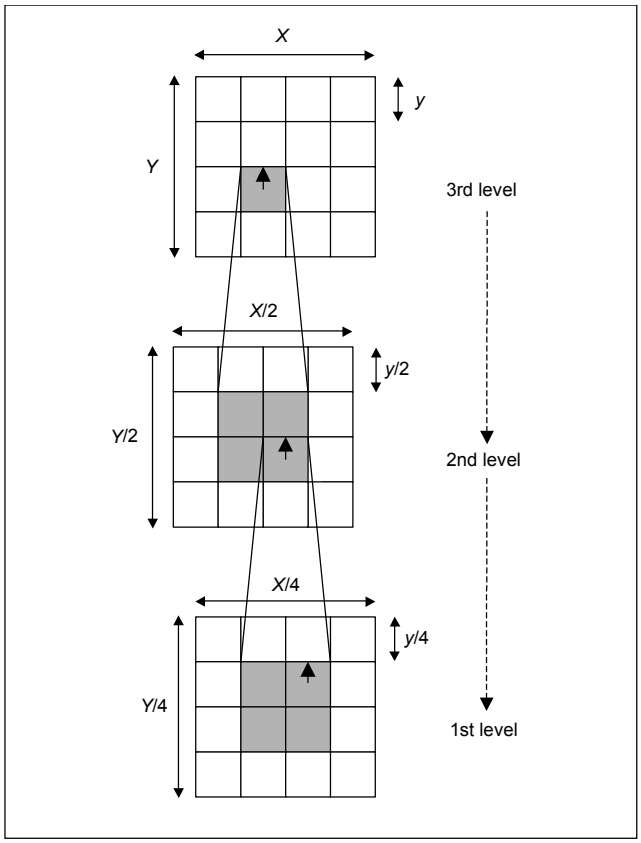

Fig. 3. Computation flow of the proposed algorithm.

array that approximates the alignment parameters is found. To obtain more exact alignment parameters using the same memory space, the proposed algorithm iterates the same process with a finer resolution having range = *Y*/2 and *uy* = *y*/2 around the positions found in the 3rd level. Finally, in the 1st level, with the finest resolution with range = *Y*/4 and *uy* = *y*/4, the exact alignment parameter is found. The alignment procedure for minutiae-matching using a multi-resolution accumulator array is as follows.

Step 1. Set the initial parameters of the accumulator array A by considering a target system. For example, when the available memory space is 10 kB and the number of *L, M,* and *N* are 64, 128, 128, respectively, set *d* = 3.

Step 2. Set the unit size of *L, M* and *N* to 2^{d-1}, i.e., $\theta \in \{\theta_l \mid 1 \le l \le \lfloor 64/2^{d-1} \rfloor \times 2^{d-1}\}, \Delta x \in \{\Delta x_m \mid 1 \le m \le \lfloor 128/2^{d-1} \rfloor \times 2^{d-1}\}$, and $\Delta y \in \{\Delta y_n \mid 1 \le n \le \lfloor 128/2^{d-1} \rfloor \times 2^{d-1}\}$. Fill the accumulator array A in this level, find the positions of maximum of array *A(i,j,k),* and compute the matching score after the alignment according to the estimated parameters.

Step 3. Let *d* = *d* – *1* and repeat the *Step 2* until *d* becomes 1.

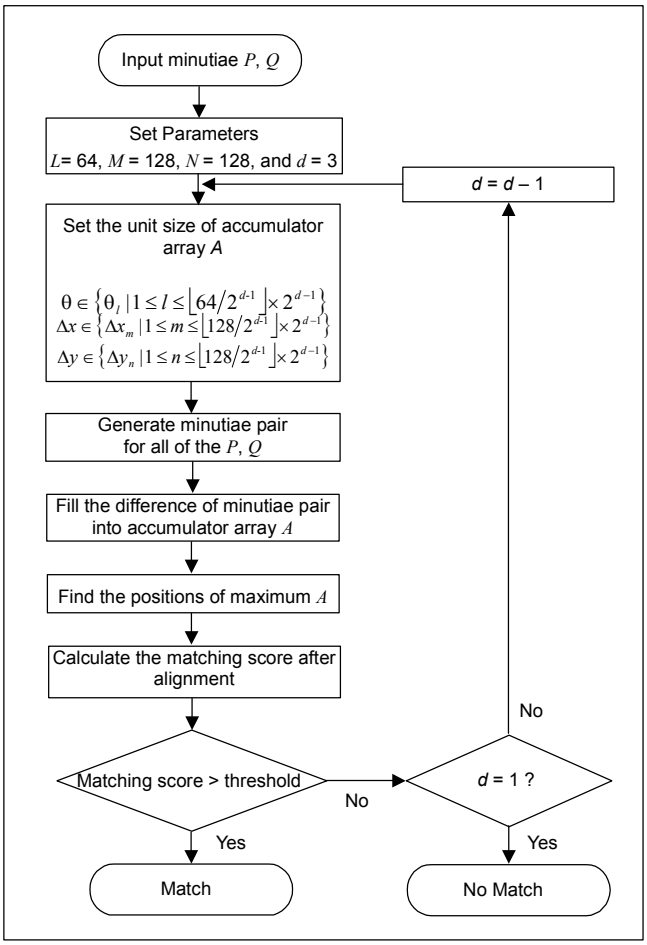

Fig. 4. Flow chart of the proposed algorithm.

Figure 4 shows the flow chart of the proposed memory-efficient algorithm using a multi-resolution accumulator array. Note that the memory requirement of the proposed algorithm is the same at each level. For example, if the numbers for L, M, and N are 64, 128, and 128, respectively, the memory space of the accumulator array A is $(64/2^{3-1})*(128/2^{3-1})*(128/2^{3-1})$ (=16,384) bytes. In levels 2 and 1, our algorithm uses $(32/2^{2-1})*(64/2^{2-1})*(64/2^{2-1})$ (=16,384) bytes and $(16/2^{1-1})*(32/2^{1-1})*(32/2^{1-1})$ (=16,384) bytes. Therefore, the required memory space of the proposed algorithm is $O(LMN/2^{3d})$. Since d can be set so that $2^{3d} = L$, the required memory space of the proposed algorithm is $O(MN)$. The straightforward implementation requires a memory space of $O(LMN)$. Note that the number of instructions of the typical algorithm is $O(|P||Q|)$, whereas that of the proposed algorithm is about $O(d|P||Q|)$.

IV. Experimental Results

In this section, we analyze the performance of the proposed fingerprint verification algorithm in terms of the number of instructions, the memory requirement, and the EER.

1. Test Environment

We tested our fingerprint verification algorithm on fingerprint images captured with an optical scanner manufactured by NitGen [24], which has resolution of 500 dpi. The size of the captured fingerprint images was 248 × 292. The fingerprint test images were provided by NitGen, our project partner, for the performance analysis of our fingerprint verification algorithm. The image set was 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. Moreover, the captured fingerprint images varied in quality.

Each fingerprint in the test image set was tested with the other fingerprints. A matching was labeled genuine if the matched fingerprint image was from the same finger as the template fingerprint image, and imposter otherwise. Six hundred genuines were found, i.e., 6 matchings were achieved per 4-fingerprint image subset from the same finger. One reference fingerprint image was chosen from four fingerprint images to perform imposter, and we used reference images only. Therefore, a total of 9900 imposters was performed, i.e., one reference fingerprint image was matched with the other 99 reference fingerprint images.

We tested for Genuines and Imposters using various parameters and evaluated not only the matching scores but also the required memory and total number of instructions. Because the smart card has very limited resources (memory and processing power), the memory requirement as well as the execution time must be evaluated. Using this information, the possibility of using a match-on-card can be determined.

2. Effects of Depth

Table 2 shows the parameter sets used to figure out how much memory space can be saved, and the tendency for the number of instructions to decrease as depth d increased. Table 3 shows the experimental results. The simulator we used was the iSAVE, which models the behavior of a 32-bit RISC processor (ARM7TDMI) in software as shown in Fig. 5 [25]. According to Table 3, when we applied a higher depth d, we needed more instructions and less memory. Limiting the memory size is critical for match-on-card. Therefore, d should be set to at least 3 to realize the match-on-card system. Note that the memory requirement size includes the required data size for both alignment and matching, and it is obtained by using the Instruction Set Simulator of the iSAVE.

Figure 6 shows the distribution of the matching scores of genuines and imposters. The vertical axis represents the distribution of the matching scores, and the horizontal axis represents the scores ranging from 0 to 100. As the figure

Table 2. Parameter sets used for matching.

	d	*ua*	*ux*	*uy*
Test_1	1	2	4	4
Test_2	2	2	4	4
Test_3	3	2	4	4

Table 3. Number of instructions and memory requirement under various depths.

	Number of Instructions (Estimated Execution Time)	Memory Requirement (bytes)
Test_1	35,377,719	436,916
Test_2	40,891,096	20,486
Test_3	48,129,074	6,836

Fig. 5. Simulation environment for fingerprint match-on-card.

shows, the three graphs have a similar distribution, which means that the matching score is rarely affected by *d*.

3. Effects of Unit Size

Table 4 represents the number of instructions and the required memory space under the various conditions when *d* is set to 3. It indicates that the memory requirement is reduced significantly as the unit size gets larger. However, the error rate is degraded slightly. Therefore, (2,4,4) is a reasonable bin unit satisfying the memory requirement of a match-on-card with an acceptable error rate.

Table 5 shows the EER under the various conditions of unit of angle and coordination when *d* is set to 3. According to Table 5, the error rate does not worsen even if the bin unit is set to (2,4,4). This is because 2 degrees and 4 pixels are more appropriate than 1 degree and 1 pixel. Using 1 pixel as a unit of the accumulator array is not reasonable for an image resolution of 500 dpi. This means that about 0.05 mm is used as one unit. Thus, it is a negligible distance.

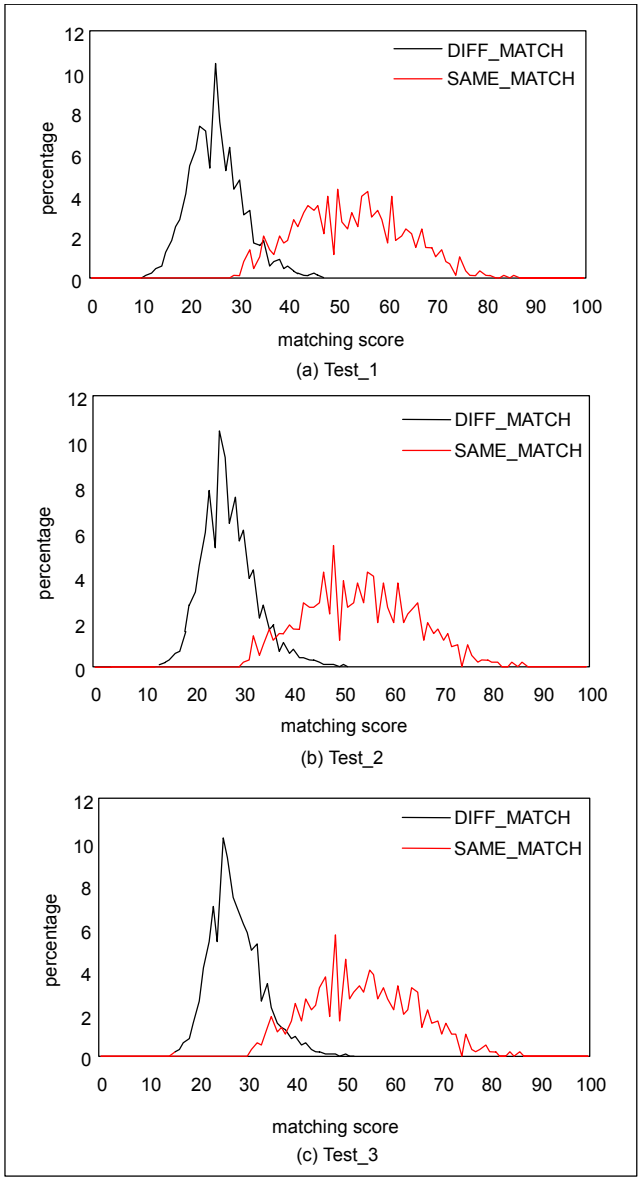

Fig. 6. Distribution of the matching scores.

Table 4. Number of instructions and memory requirement for the various unit sizes.

(*ua*, *ux*, *uy*)	Number of Instructions (Estimated Execution Time)	Memory Requirement (bytes) (Stack size + Heap size)
(1,1,1)	332,685,880	223,924 (2,460 + 221,464)
(1,2,2)	112,348,976	77,828 (2,460 + 75,368)
(2,2,2)	76,263,937	40,144 (2,460 + 37,684)
(2,4,4)	46,129,074	6,836 (2,460 + 4,376)
(4,2,2)	58,183,205	24,338 (2,460 + 21,878)
(4,4,4)	43,873,083	5,420 (2,460 + 2,960)

Table 5. Equal error rate for the various unit sizes.

	(*ua*, *ux*, *uy*)						
	(1,1,1)	(1,2,2)	(2,2,2)	(2,4,4)	(4,2,2)	(4,4,4)	(4,8,8)
EER(%)	7.5	6.10	5.85	6.00	6.11	6.67	7.17

V. Conclusion

The smart card is an example of a secure device, and biometrics is a promising technology for verification. These two can be combined for many applications to enhance both security and convenience. However, typical biometric verification algorithms that have been executed on standard PCs have not been able to be executed in real time in a resource-constrained environment such as a smart card.

In this paper, we have presented a memory-efficient fingerprint verification algorithm that can be executed in real time on a smart card. To meet the processing power and memory space specifications of the smart card, we first evaluated both the processing power and the memory requirement of each fingerprint verification step. We found that the memory requirement of the alignment stage was the most critical factor in implementing the match-on-card. To reduce the memory requirement, we employed a small-sized accumulator array. Then, to compute the alignment parameters more accurately, we performed more computations from a coarse-grain to a fine-grain resolution on the accumulator array.

We performed the experimental evaluations on the iSAVE with the NitGen fingerprint database, and the experimental results were very encouraging. Given 248 × 292 fingerprint images, the matching was completed in 0.9 seconds with 6.8 kB. The same step was performed in 0.3 seconds with 400 kB using the typical algorithm. The accuracy (the EER) of the proposed algorithm was also comparable to that of the typical algorithm (6.0% vs. 3.8%). We are currently improving the accuracy further while maintaining the small-size memory requirement. We believe that the memory-efficient technique developed in this paper offers a general framework for developing other biometric algorithms in resource-constrained environments such as the smart card.

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] Y. Seto, "Personal Authentication Technology using Biometrics," *SICE*, vol. 37, no. 6, 1998, pp. 395-401.

[4] F. Gamble, L. Frye, and D. Grieser, "Real-time Fingerprint Verification System," *Applied Optics*, vol. 31, no. 5, 1992, pp. 652-655.

[5] C. Wilson, C. Watson, and E. Paek, "Effect of Resolution and Image Quality on Combined Optical and Neural Network Fingerprint Matching," *Pattern Recognition*, vol. 33, no. 2, 2000, pp. 317-331.

[6] C. Lee and S. Wang, "Fingerprint Feature Extraction using Gabor Filters," *Electronics Lett*., vol. 35, no. 4, 1999, pp. 288-290.

[7] A. Jain, L. Hong, and R. Bolle, "On-line Fingerprint Verification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, 1997, pp. 302-313.

[8] D. Isenor and S. Zaky, "Fingerprint Identification using Graph Matching," *Pattern Recognition*, vol. 19, no. 2, 1986, pp. 113-122.

[9] K. Fan, C. Liu, and Y. Wang, "A Fuzzy Bipartite Weighted Graph Matching Approach to Fingerprint Verification," *Proc. of The IEEE Int'l Conf. On Systems, Man and Cybernetics*, 1998, pp. 729-733.

[10] Shinyoung Lim et al., "Efficient Iris Recognition through Improvement of Feature Vector and Classifier," *ETRI J*., vol. 23, no. 2, June 2001, pp. 61-70.

[11] Sang Kyun Im et al., "A Direction-Based Vascular Pattern Extraction Algorithm for Hand Vascular Pattern Verification ," *ETRI J*., vol. 25, no. 2, Apr. 2003, pp. 101-108.

[12] H. Dreifus and T. Monk, *Smart Cards*, John Wiley & Sons, 1997.

[13] H. Kim et al., "Specification for the Next-Generation IC Card System(Korean)," *Technical Report*, ETRI, 2000.

[14] G. Hachez, F. Koeune, and J. Quisquater, "Biometrics, Access Control, Smart Cards: A Not So Simple Combination," *Proc. of the 4th Working Conf. on Smart Card Research and Advanced Applications*, 2000, pp. 273-288.

[15] B. Struif, "Use of Biometrics for User Verification in Electronic Signature Smartcards," *Proc. of E-smart* 2001, LNCS 2140, 2001, pp. 220-227.

[16] R. Sanchez-Reillo, "Smart Card Information and Operations using Biometrics," *IEEE AEES Mag*., 2001, pp. 3-6.

[17] R. Sanchez-Reillo and A. Gonzalez-Marcos, "Access Control System with Hand Geometry Verification and Smart Cards," *Proc. of Int'l Carnahan Conf*., 1999, pp. 485-487.

[18] Y. Moon, H. Ho, K. Ng, S. Wan, and S. Wong, "Collaborative Fingerprint Authentication by Smart Card and a Trusted Host," *Electrical and Computer Engineering*, vol. 1, 2000, pp. 108-112.

[19] M. Janke, *FingerCard Project Presentation*, http://www.finger-card.org, 2001.

[20] N. Kaku, T. Murayama, S. Yamamoto, "Fingerprint Authentication System for Smart Cards," *Proc. of IFIP on E-commerce, E-business, E-government*, 2001, pp. 97-112.

[21] R. Sanchez-Reillo and C. Sanchez-Avila, "Fingerprint Verification using Smart Cards for Access Control Systems," *Proc. of Int'l Carnahan Conf*., 2001, pp. 250-253.

[22] Y. Gil, Y. Chung, D. Ahn, J. Moon, and H. Kim, "Performance Analysis of Smart Card-based Fingerprint Recognition for Secure User Authentication," *Proc. of IFIP on E-commerce, E-business,*

E-government, 2001, pp. 87-96.
[23] Samsung, www.samsungelectronics.com.
[24] NitGen, www.nitgen.com.
[25] iSAVE, www.dynalith.com.

Sung Bum Pan received the BS, MS, and PhD degrees in electronics engineering from Sogang University, Korea, in 1991, 1995, and 1999. He joined ETRI in 1999 and he is currently a Senior Member of Engineering Staff at the Biometric Technology Research Team. His current research interests are biometrics, security, and VLSI architectures for real-time image processing.

Youn Hee Gil received the BS and MS degrees in computer engineering from Busan National University, Korea, in 1999 and 2001. After graduation, she joined ETRI as a Researcher for the Biometrics Technology Research Team. Her current research interests include biometrics, especially fingerprint recognition, pattern recognition, and security.

Daesung Moon received his MS degrees from Busan National University, Korea, in 2001. He joined ETRI in 2001 and he has been a Member Of Research Staff at the Biometric Technology Research Team. His research areas are biometrics, image processing, and security.

Yongwha Chung received his BS and MS degrees from Hanyang University, Korea, in 1984 and 1986. He received his PhD degree from the University of Southern California, USA in 1997. He joined ETRI in 1986 and he is currently working as the Team Leader of the Biometric Technology Research Team. His research interests include biometrics, security, and performance optimization.

Chee Hang Park received the BS degree in applied physics from Seoul National University, Korea, in 1974, the MS degree in computer science from Korea Advanced Institute of Science and Technology, Korea, in 1980, and the PhD degree in computer science from University of Paris 6, France, in 1987. During the last 10 years, he has been involved as Project Leader with several large projects such as multimedia computer system development and high speed parallel computer system development. His research interests include security, multimedia systems, distributed systems, middleware, groupware, network virtual computing, and mobile agent architecture. He is currently the Executive Director of the Information Security Research Division of Electronics and Telecommunications Research Institute.