8

Fingerprint Authentication System For Smart Cards

Nikkou Kaku, Takahiko Murayama and Shuichiro Yamamoto
NTT Information Sharing Platform Laboratories, NTT Corporation

Abstract: Among the several methods that enable fingerprint authentication of card holders, the one that executes the matching process on the smart card chip is the most important. The Pattern matching method, such as [1], is of no use for smart cards, because it takes too much time to adjust the position between the input image and template image. So we have developed an algorithm (FTA: Free Turning Algorithm) using minutiae that does not need the position correcting process. In this paper, we describe the basic effectiveness of this algorithm and a prototype system that implements it on a smart card.

1. INTRODUCTION

Smart cards have attracted a great deal of public attention because of their high security and the variety of services they enable, and they have been introduced in various fields. Although the personal identification number (PIN) is widely used as a method of authenticating card holders (for example, for magnetic-stripe ATM cards), biometrics (fingerprints and irises etc.) are expected to replace it because biometrics overcome the problems of users forgetting their PINs or having them stolen[2]. Fingerprint-based methods are especially practicable, because the error rate is comparatively low and the hardware is cheap. The three models shown in Table 1 are considered typical models that can be used for a fingerprint authentication system. They are the Server Authentication Model (SAM), Fingerprint Scanner Authentication Model (FSAM), and Smart Card Authentication Model (SCAM).

Table 1. Typical Models of a Fingerprint Authentication System for Smart Cards

Model	Recognition engine location
SAM	Host server
FSAM	Fingerprint scanner
SCAM	Smart card chip

Comparing them, SCAM has several advantages.

1. It is user-friendly in that card holders can manage their fingerprints with their smart cards by themselves.
2. It can reduce the cost of administering and maintaining the database of template fingerprints.

However, it is not practical for implementing a pattern-matching algorithm such as [1] in smart cards, because the algorithm requires too much time to adjust the input and template image positions. Therefore, we have developed an algorithm (FTA: Free Turning Algorithm) that does not need positional correction between the input and template images. FTA can omit this step because it uses a feature vector (the Relative Vector) composed of components describing the relationship between two minutiae (e.g., distance). Consequently, a card holder can be authenticated by touching the fingerprint scanner at any angle. If chip-sized fingerprint scanners are embedded in contactless smart cards in the future, the features of FTA will be particularly effective because card holders are likely to present their cards at various angles.

Section 2 explains the problems of implementing the algorithm [1] in smart cards. Section 3 describes our algorithm, FTA, which is evaluated in Section 4. Section 5 describes a prototype system implementing our algorithm in smart cards. Section 6 concludes this paper.

2. THE PROBLEM WITH THE THINNED IMAGE PATTERN-MATCHING ALGORITHM

The pattern-matching algorithm [1] adjusts the positions of a binarized input image and a thinned template image by moving them relative to each other in the vertical and horizontal directions and rotating them. The two images are then compared by matching the corresponding pixels at each position. Consequently, the computational complexity of the pattern-matching algorithm [1] is high (Fig. 1). We found that executing the algorithm for matching the binarized input image and the thinned template image (size: 256×256 pixels), without correcting the position, took 8 seconds on a smart card with a 5-MHz processing speed, a 4-kbit ROM, and a 512-kbit Flash memory. Therefore, the algorithm would need about 53 minutes to match those two images by the usual procedure including positional correction on the smart card [15].

[15] The algorithm [1] adjusts the relative positions by 10 pixels vertically, 10 pixels horizontally, and 4 degrees of rotation.

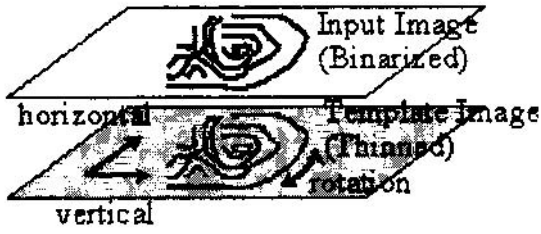

Figure 1. Thinned Image Pattern Matching Algorithm

3. FREE TURNING ALGORITHM (FTA)

3.1 Relative Vector

We define the following vector which forms the basis for the Relative Vector. We call the vector Extension, because it can be regarded as an extension of the ordinary feature vector $P(x, y, \theta)$.

Definition 1: Extension

We define a coordinate system C whose origin is at the center point of the fingerprint image[16]. We define the extraction region ER as the region enclosed by a circle of radius r centered on the origin. And we number the minutiae counterclockwise in order of increasing distance from the origin (Fig.2).

[16] For example, if the image size is 256×256, then the origin of C is at (128,128).

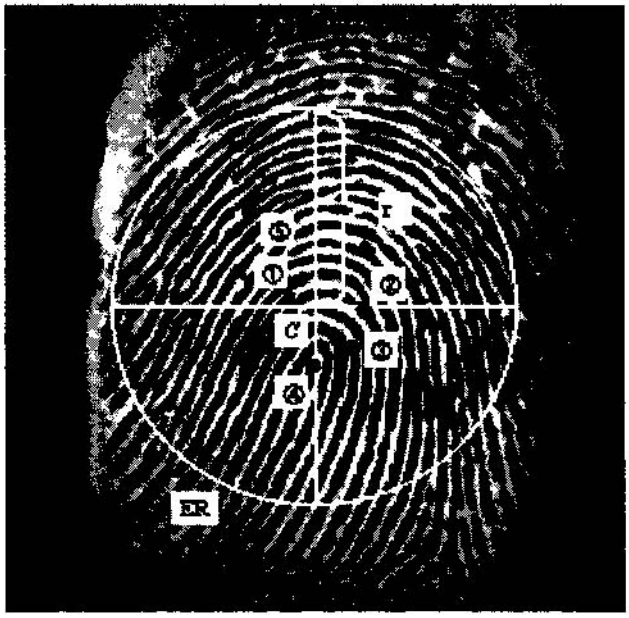

Figure 2. Extraction Region

Now we define the Extension m_i (Fig.3 [17]).

$$m_i = (x_i, y_i, \theta_{(i,trace(1))}, \ldots, \theta_{(i,trace(N))}, kind_i, i)$$

Here □

a) x_i, y_i are coordinates of C.

b) $trace(n)(n = 1, \ldots, N) \in \{1, 2, \ldots\}$ and $trace(1) < trace(2) < \ldots < trace(N)$. The $trace(n)$ is the number of pixels when tracing the ridge. And n is the ID number of a tracing point [18].

c) $\theta_{(i,trace(n))}(0 \leq \theta_{(i,trace(n))} < 2\pi)$ is the angle of the vector starting at the location of m_i and ending at the location found by tracing the ridge $trace(n)$ pixels, to C. If m_i is a ridge bifurcation, we trace the ridge that has obtuse angles on both sides.

d) $kind_i$ indicates the kind of minutiae. If m_i is a ridge bifurcation, then $kind_i = 0$. If m_i is a ridge ending, then $kind_i = 1$.

e) i is the ID number of m_i.

[17] Figure 3 shows an example of the Relative Vector of the ridge bifurcation described in Fig.2 No. □. For ease of explanation, we move the origin of C to the location of the ridge bifurcation.

[18] A tracing point is a point that can be found by tracing the ridge.

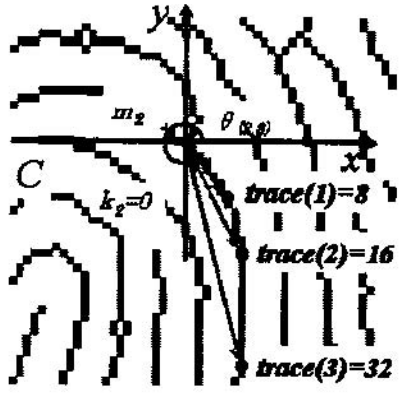

Figure 3. Extension

Definition 2: Relative Vector

Using Extension m_i, m_j, we define the Relative Vector m_{ij} (Fig.4).

$$m_{ij} = (len_{ij}, d\theta_{(ij,trace(1))}, \ldots, d\theta_{(ij,trace(N))}, kind_{ij}, i, j)$$

Here,

a) $\mathrm{len}_{ij} = ((x_i - x_j)^2 + (y_i - y_j)^2)^{1/2}$

b) $d\theta_{(ij,trace(n))} = |\theta_{(i,trace(n))} - \theta_{(j,trace(n))}|$

c) if $\pi < d\theta_{(ij,trace(n))} < 2\pi$ then

d) $d\theta_{(ij,trace(n))} = 2\pi - |\theta_{(i,trace(n))} - \theta_{(j,trace(n))}|$

e) $kind_{ij} = 0$ (if $kind_i = kind_j = 0$)

f) = 1(if $kind_i = kind_j = 1$)

g) = 2(if $kind_i = 0$, $kind_j = 1$ or $kind_i = 1$, $kind_j = 0$)

h) This indicates a kind of Relative Vector.

i) i is the ID number of m_i and j is the ID number of m_j.

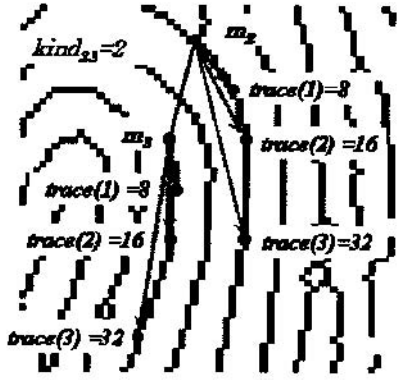

Figure 4. Relative Vector

3.2 FTA Overview

We present the complete FTA algorithm in Appendix. Here, we give an overview of it.

First, let us define some notation. Define the set of Relative Vectors of the template image as $R = \{m^r_{ij}\}$,

$$m^r_{ij} = (len^r_{ij}, d\theta^r_{(ij,trace(1))}, \ldots, d\theta^r_{(ij,trace(N))}, kind^r_{ij}, i^r, j^r)$$

and define the set of Relative Vectors of the input image as $S = \{m^s_{ij}\}$.

$$m^s_{ij} = (len^s_{ij}, d\theta^s_{(ij,trace(1))}, \ldots, d\theta^s_{(ij,trace(N))}, kind^s_{ij}, i^s, j^s)$$

The notation |A| describes the number of components of the set *A*. The basic idea of FTA is matching the Relative Vectors of the input image with the Relative Vectors of the template image (Fig.5).

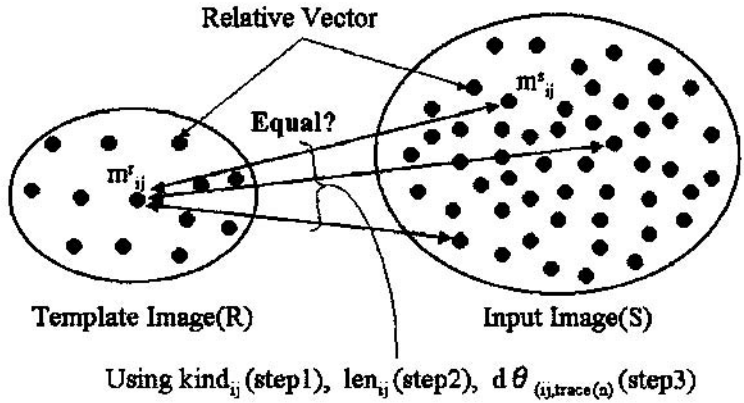

Figure 5. FTA Overview

The steps of the algorithm are as follows.

[step 0] Using len_{ij}, sort the Relative Vectors of the template image in ascending order and assign a number to each Relative Vector.

[step 1] Using $kind_{ij}$, find the Relative Vector from the input image that corresponds to the Relative Vector of the template image. Make set A_1 composed by such Relative Vectors.

[step 2] Using len_{ij}, find the Relative Vector from A_1 corresponding to the Relative Vector of the template image. Make set A_2 composed by such Relative Vectors. Here, the difference in length of the Relative Vectors of the template and input images has to be under the threshold (*THR_LEN*).

[step 3] Using *d_ang*[*n*], sort the Relative Vectors of A_2. This sort is executed *N* times. If m^s_{ij}, which have a minimum *d_ang*[*n*] in each sorting process are identical to each other and all minimum *d_ang*[*n*] in each sorting process are under the corresponding threshold *THR_ANG*[*n*], then add one to the *count*. Otherwise, go back to **step 1.**

[step 4] If *RATE* (*count* /|*R*|), which is the matching rate of the Relative Vectors of the input and template images is over the threshold(*THR*), then the input image is identical to the template image. Otherwise, the input image is not identical to the template image.

4. EXPERIMENT

We conducted three experiments to evaluate the False Acceptance Rate (FAR), False Rejection Rate (FRR), and Equal Error Rate (EER) of FTA. We used a fingerprint database openly available on the Worldwide Web[19], which contains scanned fingerprint images from twenty-one subjects, who each provided eight fingerprint images from the same finger. The database is composed of 168 (21×8) fingerprint images and the resolution of each image is about 350 dpi. We conducted 147 (=(8-1)×21) matching tests between the same subjects, and 3360 (=8×20×21) matching tests between different subjects. An example of the database is shown in Figs. 6 and 7[20]. We applied three types of preprocessing: smoothing, binarizing, and thinning.

Figure 6. fp1-4

Figure 7. fp17-5

[19] http://bias.csr.unibo.it/research/biolab/bio_tree.html

[20] Here, fp1-4 denotes the fourth fingerprint image of subject no. 1 and fp 17-5 the fifth fingerprint image of subject no. 17.

Table 2 shows the parameters of the three experiments.

Table 2. Parameters

	|R|	|S|	N	trace(1)	Trace(2)	trace(3)
Experiment 1	45	190	2	5	10	-
Experiment 2	45	190	2	18	20	-
Experiment 3	45	190	3	5	10	15

Here, $|R|$ and $|S|$ are the numbers of Relative Vectors of the template and input images, respectively. N is the number of tracing points. And $trace(1)$, $trace(2)$, and $trace(3)$ are the numbers of pixels of the first, second, and third tracing points, respectively. In these experiments, we defined the maximum of $trace(n)$ as 20 pixels, because few minutiae were detected when it was over 20.

4.1 Preliminary experiment

We decided the template image of each subject and the FTA threshold by the following methods.

4.1.1 Deciding the template image

Using eight fingerprint images of the same subject, we conducted a round-robin matching on the condition that the threshold of the length difference (THR_LEN) and the threshold of the angle difference ($THR\_ANG[n]$) were maximum[21]. We then take the template image to be the image having the maximum average matching score ($count/|R|$).

Table 3 [22] shows the template image of each experiment decided by this method.

[21] The difference in length, i.e., $|len^s_{ij} - len^s_{ij}|$, did not exceed 160 (pixels), because we defined the radius of the Extraction Region as 80 (pixels). And the difference in angle, i.e., $|d\theta^s_{(ij,trace(n)} - d\theta^r_{(ij,trace(n)}|$ $(n = 1, \ldots, N)$, did not exceed π . Therefore, we defined THR_LEN and $THR\_ANG[n]$ as 160 (pixels) and 4 (radians), respectively.

[22] We abbreviate fp in Table 3. And because of a lack of Relative Vectors, we could not define a template image for some conditions.

Table 3. Template Images

Subject	Experiment 1	Experiment 2	Experiment 3
1	1-1	1-3	1-3
2	2-8	2-5	2-1
3	3-3	3-1	3-3
4	4-2	4-1	4-2
5	5-7 or 5-8	5-4	5-4
6	6-4	6-6	6-4
7	7-5	7-1	7-1
8	8-3	8-3	8-8
9	9-5	9-5	9-5 or 9-6
10	10-4	10-1	10-4
11	-	-	11-5
12	12-5	12-8	12-1
13	13-6	13-7	13-6
14	14-4	14-8	14-4
15	15-3	15-4	15-4
16	16-7	16-6	16-2 or 16-5
17	17-2	17-6	17-2
18	18-3	18-3	18-6
19	-	-	-
20	20-3	20-4	20-8
21	-	-	-

4.1.2 Deciding the threshold

Using the template images shown in Table 3, we conducted matching tests for the same subjects and among different subjects on condition that *THR_LEN* and *THR_ANG*[*n*] were both maximum. Then we plotted their cumulative frequency distributions of $|len^s_{ij} - len^s_{ij}|$, which were calculated in the matching tests for the same and among different subjects, respectively. We define the interval maximizing the difference of these cumulative frequencies as the threshold *THR_LEN*.

Similarly, we plotted the cumulative frequency distributions of $|d\theta^s_{(ij,trace(n)} - d\theta^r_{(ij,trace(n)}|$, which were calculated in the matching tests for the same subject and among different subjects, respectively. We define the interval maximizing the difference of these cumulative frequencies as the threshold *THR_ANG*[*n*].

Table 4 shows the threshold for each experiment decided by these procedures.

Table 4. Thresholds

	THR_LEN	*THR_ANG* [0]	*THR_ ANG*[1]	*THR_ANG*[2]
Experiment 1	1.6	0.2	0.216	-
Experiment 2	2.1	0.162	0.222	-
Experiment 3	2.1	0.221	0.206	0.242

4.2 Experimental Results and Evaluation

Figures 8, 9 and 10 show the FAR-FRRs of experiments 1, 2, and 3, respectively.

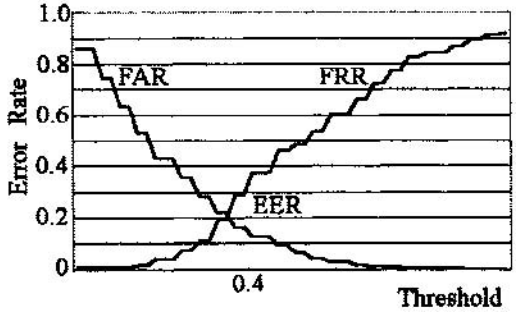

Figure 8. FAR-FRR of Experiment 1

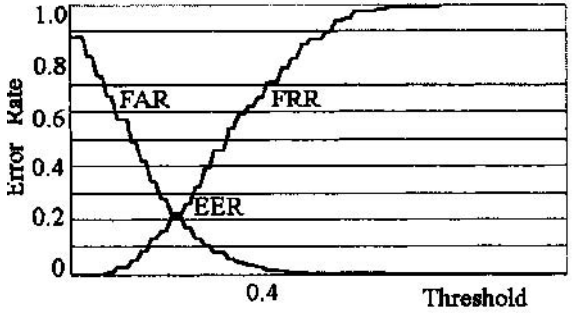

Figure 9. FAR-FRR of Experiment 2

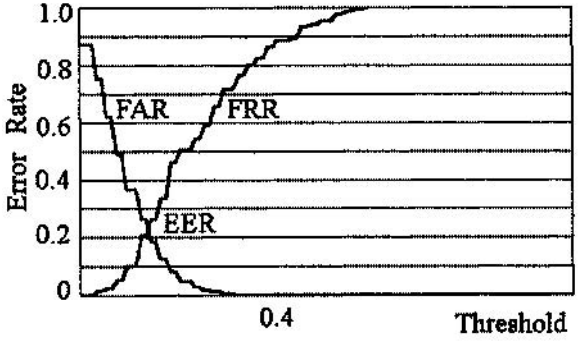

Figure 10. FAR-FRR of Experiment 3

Comparing Figs. 8 and 9, we can see that the EER does not change with the number of pixels of the tracing point. Similarly, comparing Figs. 8 and 10, we can see that it does not change with the number of tracing points, either. Therefore, we can conclude that the EER of FTA is approximately 0.2.

Next we consider the possibility of improving the EER of FTA by analyzing the results of experiment 1. Figure 11 shows that the individual EERs of the subjects varied widely in experiment 1. To determine the cause of this variation, we examined every thinned image of each subject. We found many false minutiae caused by interrupted ridge lines in the thinned images of subjects who had a high EER, but few false minutiae in the thinned images of subjects who had a low EER [23].

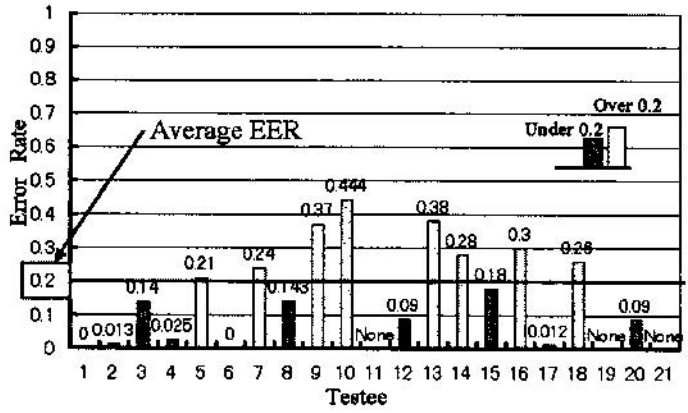

Figure 11. EER for each Subject in Experiment 1

Table 5 shows the rate of false minutiae occurring in the thinned images of typical subjects. Figures 12 and 13 show the thinned images of fp1-1 and fp10-4, respectively. In Figs. 12 and 13, the symbol ■ represents false minutiae and • represents true minutiae.

Table 5. Average Rate of False Minutiae and EER

Subject	Average Rate of False Minutiae	EER
1	0.2	0
3	0.6	0.14
5	0.825	0.21
10	0.85	0.444

[23] We examined the minutiae detected by the FTA manually.

Figure 12. Thinned Image (fp 1-1)

Figure 13. Thinned Image (fp 10-4)

From the above discussion, we conclude that the authentication accuracy of FTA can be improved by reducing the occurrence of false minutiae caused by interrupted ridge lines. We have demonstrated the basic effectiveness of FTA.

5. PROTOTYPE SYSTEM

In this section we describe the prototype system used to implement our algorithm on a smart card. Figure 14 shows the processing procedure of the prototype system, and Figure 15 shows the prototype itself.

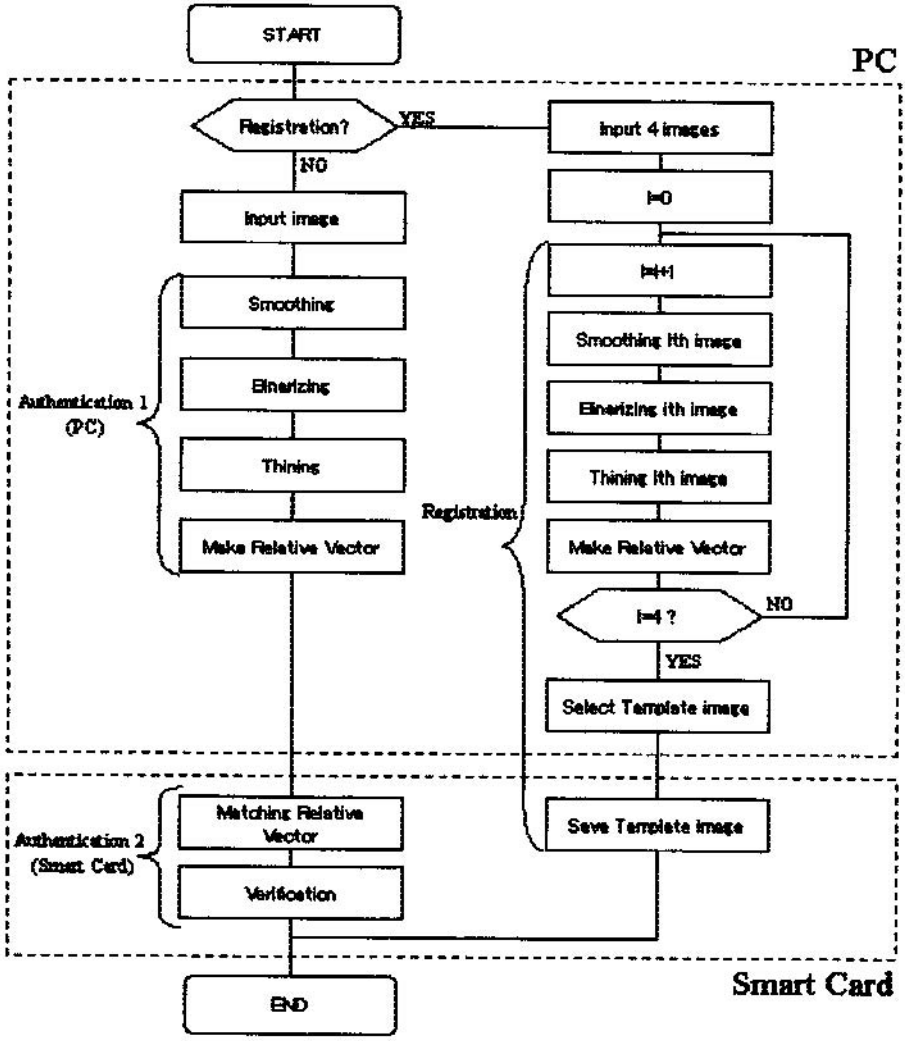

Figure 14. Prototype System Procedure

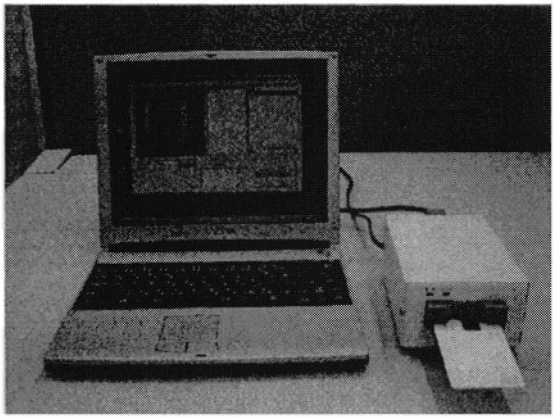

Figure 15. Prototype System

5.1 Overview of each process

Here, we briefly explain each process of the prototype system (Fig.14).

Smoothing. The 256×256 pixel gray-scale image was smoothed with a 3×3 mask. The brightness of the center point was defined as the average brightness of the eight neighbors.

Binarizing. We refer to a 4×4 pixel area as a block. We calculated the average brightness for every block, and the average brightness for a 3×3 block area. Using the latter, we binarized the 3×3 block area.

Thinning. We used the Hilditch algorithm[3].

Make Relative Vector. After detecting the minutiae, we made the Relative Vector as explained in Section 3.1.

Choose template image. We conducted round-robin matching for four input images. We used the image with the highest average matching score as the template image.

Matching. The Relative Vector of the input image was matched to the Relative Vector of the template image.

5.2 Evaluation of prototype system

We evaluated the processing speed of our prototype system under the conditions shown in Table 6. Table 7 shows the evaluation results.

Table 6. Evaluation Conditions

PC	266MHz
Smart Card	CPU:5MHz,ROM:4kbits,RAM:512kbits
Image	256×256,gray-scale image

Table 7. Results(See Fig. 14)

Procedure	Time
Registration	8s
Authentication 1(PC)	2s
Authentication 2(Smart Card)	6s

6. CONCLUSION

We have developed an algorithm, FTA, that has low computational complexity and does not need position correction between input and template images in order to execute fingerprint authentication on a smart card chip. We have demonstrated the basic effectiveness of FTA and made a prototype system.

With the FTA implemented on the smart card, the FTA matching process takes six seconds. Our goal is to reduce the FTA matching time to within one second through an optimum implementation.

References

[1]Tetsuji KOBAYASHI, "A Fingerprint Verification Method Using Thinned Image Pattern Matching," IEICE TRANSA-CTIONS, Vol.J79-D-□,No.3,pp.330-340,March 1996.

[2]Youichi SETO, "Personal authentication technology using biometrics", SICE Vol.37,No.6, pp.395-401,1998.

[3]J. Hilditch,"Linear skeletons from square cupboards", Machine Intelligence 6, Edinburgh Univ. Press, pp. 404-420, 1969.

Appendix

Define the set of Relative Vectors of a template image as $R = \{m^r_{ij}\}$,

$m^r_{ij} = (len^r_{ij}, d\theta^r_{(ij,trace(1))}, \ldots, d\theta^r_{(ij,trace(N))}, kind^r_{ij}, i^r, j^r)$

and define the set of Relative Vectors of an input image as $S = \{m^s_{ij}\}$,

$m^s_{ij} = (len^s_{ij}, d\theta^s_{(ij,trace(1))}, \ldots, d\theta^s_{(ij,trace(N))}, kind^s_{ij}, i^s, j^s)$

and, $N \geq 2$.

main(){

(*step 0*)

Using len^r_{ij}, sort the set R in ascending order;

Give a number to each Relative Vector in that order

$m^r_{ij\text{-}order} = (len^r_{ij\text{-}order}, d\theta^r_{(ij\text{-}order,trace(1))}, \ldots, d\theta^r_{(ij\text{-}order,trace(N))}, kind^r_{ij\text{-}order}, i^r, j^r)$;

count=0;

for(*order* = 1; *order* ≤|*R*|; *order*++){

(*step 1*)

Make set $A_1 = \{ m^s_{ij} \mid kind^s_{ij} = kind^s_{ij\text{-}order} \}$;

(*step 2*)

```
        Make set A1.5 = { m^s_ij | m^s_ij ∈ A1 ,  | len^s_ij - len^r_ij-order | ≤ THR_LEN }
        for(x=0;x<3;x++){
            Search m^s_ij from the set A1.5
  which has minimum |len^s_ij - len^r_ij-order| ;
            Add m^s_ij to set A2 ;
            Delete m^s_ij from set A1.5 ;
        }
        ( step 3 )
        for(n=1;n ≤ N;n++){
            Search for m^s_ij from set A2 which has
  minimum |dθ^s_(ij-order,trace(n))  - dθ^r_(ij-order,trace(n))| ;
            Substitute d_ang[n] for such
               |dθ^s_(ij-order,trace(n))  - dθ^r_(ij-order,trace(n))| ;
            Substitute i^x , j^x for i^s , j^s , respectively,
               where i^s , j^s are ID numbers of such m^s_ij ;
            if( d_ang[n] ≤ THR_ANG[n] ){  i^s[n] = i^x ; j^s[n] = j^x ;
               if( i^s[1] =...= i^s[n] && j^s[1] =...= j^s[n] && n = N ){
  /* Final Loop */
                    count++ ;
                }
            } else {
                break ;
            }
        }
    }
    ( step 4 )
    if( count /|R| ≥ THR ){
        Matching OK ;
    } else {
        Matching NG ;
    }
}
```