ISSN No. 0976-5697

Volume 4, No. 4, March-April 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Matching of Fingerprint Geometry by Advanced Divide and Conquer Technique

Sanjukta Pal*
Student
M.Tech (CST) Dr. B. C. Roy Engineering College,
Durgapur, West Bengal, India
e-mail: sanjukta_2008pal@rediffmail.com

Sucharita pal
Assistant professor
Electrical Engeneering, Asansol engineering College,
Asansol, West Bengal, India
e-mail: sucharita_biet@rediffmail.com

Prof.(Dr) Pranam Paul
Professor
Computer Application, Narula Institute of Technology,
Agpara, West Bengal, India
e-mail: pranam.paul@gmail.com

Abstract: There exist so many techniques for matching two finger print geometry. But this is an algorithm in which divide and conquer technique is used in an advanced manner. Here at first, the whole image are fragmented into four segments, and finally the four image segment would be chosen from previous different four segments, at first those segments will check the DB Image individually one by one. If the matched value would be greater than the threshold value then neighboring pixel matching would proceed further. After checking the whole image, if we get the result above threshold value then the two fingerprint images would matched.
Here the main advantage is that, in the first checking the DB Image would be traversed by four segments, so there exists a better probability to get result above threshold value, because if anyone or any two does not fulfill the criteria then the rest can.

Keywords: Biometric Authentication, DB image, Final image, image segment.

I. INTRODUCTION

One main aspect for matching two fingerprint geometry is that two fingerprint images of same finger may not be exactly same to same in every time. For this reason there must be a certain threshold value. If the matched result would be greater than the threshold value then it will give positive result that means the two images are matched. But the threshold value must be assigned by the coder.
In this algorithm first the image would be divided into four segments. Then another four different image segments would be chosen from previous different image segments which must satisfy certain given criteria, then those would be called finally selected image segments. This segments would traverse one DB Image one by one.
Here the four segments are taken from nearby centre position, and have the better probability to get result above threshold value because four segments will check the larger area of DB Image individually. This is the advantage for using this algorithm.
One main thing is that, after traversing the DB Image when we get the fixed up pixel regions, they cannot be overlapped further else. Because the four image segments from Final Image are taken from different area from final image. The fixed up image segments can intersect each other at a point or at a line.
After fixing up pixel regions which are matched above threshold value, we can proceed further else, that means matching of neighboring pixels would be happened. It will decide the result which can give positive result, means the two fingerprint images (one is DB Image and another is Final Image) are matched.

II. ALGORITHM

1st, Read 5 or 6 fingerprint images from different angle (say DB Images).
1a: convert images into gray scale image.
1b: convert gray scale image into binary image.

Finally for matching purpose one fingerprint image would be taken (say, Final Image)
2a: convert final image into gray scale image
2b: convert this gray scale image into binary image.

The DB Images and Final Image may not be taken from same device, so the images may not be of same size.

Then for matching,

Step-1: calculate the number of pixels along width and in length (say m and n) from final image.

Step-2: Using m, n derive the centre position (p, q) of Final Image by (m+1/2, n/2) or by (m/2, n+1/2) or by (m+1/2, n+1/2) or by (m/2, n/2).

Step-3: Using (p, q) as centre position break the image into four segment.

Step-3a:Identify the bottom right image segment as A, bottom left corner image segment as B, upper left corner image segment as C and the upper right corner image segment as D.

Step-4: If the number of pixel along width and length of the image segments is greater than 10 then go to step1 and repeat step2 and step3(up to first part of step 3).

Step-5: Take the upper left corner image segment (say A_1) of the previous bottom right image segment A, upper right corner image segment (say B_1) of the previous bottom left image segment B, bottom right corner image segment (say C_1) of the previous upper left image segment C and bottom left corner image segment (say D_1) of the previous upper right image segment D.

Step5a: If the number of pixel along width and length of previously selected image segments (i.e. $A_{1,} B_{1,} C_{1,} D_1$) is greater than 10 then go to step4.

Step5b: If the number of pixel along width and length of previously selected image segments (i.e. $A_{1,} B_{1,} C_{1,} D_1$) is less or equal to 10 then go to step6.

.

Step-6: Using the finally selected image segments (i.e. $A_{n,} B_{n,} C_{n,} D_n$), traverse one db image One by one, pixel by pixel in row wise and column wise.
(Here $A_{n,}$ $B_{n,}$ $C_{n,}$ D_n are said to be finally selected image segments because after breaking an image n+1 times, we will get those segments which can full fill the above criteria, that means after breaking an image 6 times, if it full fill the criteria then we can select one of the finally selected image segment say A_5)

6a: For each traverse with first, finally selected image segment A_n store upper left most pixel position (a) and bottom right most pixel position (b) of DB Image and percentage of pixel match (c) during that traverse.

6b: For each traverse with second, finally selected image segment B_n store bottom left most pixel position (d) and upper right most pixel position (e) of DB Image and percentage of pixel match (f) during that traverse.

6c: For each traverse with third, finally selected image segment C_n store upper left most pixel position (g) and bottom right most pixel position (h) of DB Image and percentage of pixel match (i) during that traverse.

6d: For each traverse with fourth, finally selected image segment D_n store upper right most pixel position (j) and bottom left most pixel position (k) of db image and percentage of pixel match (l) during that traverse.

Step-7: Store (a, b, c), (d, e, f), (g, h, i) and (j, k, l) of maximum matching of any two simultaneous traverses with $A_{n,} B_{n,} C_{n,} D_n$.

Step-8: If any one of the maximum matching of pixel(i.e. any one of c, f, i or l) is greater than the threshold value, then go to step 9,
Else go to step15.

Step-9: Using the pixel positions of those image segment/s fix up the pixel chunk/s in the DB Image (those pixel chunks which match over threshold value). If there exists only one chunk then go to step11 directly. Else go to step11.

Step-10: Check those image segments to find the intersecting point/s between them (it may be an intersecting point or intersecting line).

10a: If the intersecting point/s or line is not found, then check those image segments individually.

10a1: If any two or three image segments are above threshold value then take the segment with maximum matching an go to step11.

10b: If the intersecting point/s or line is found, then check

10b1: if the image segments (those who have intersecting point/s or line) are all above threshold value then combine those image segments and go to step13.
(after this step, direct checking of neighboring pixel checking would be started because the four segments are already combined)

10b2: if any two or three image segments (those who have intersecting point/s or line) is above the threshold value and the rest are below threshold value, then combine only those image segments which are above threshold value then go to step11.

Step-11: Check the DB Image with the total area of four segment of one image segment (the segment may be single segment just like step 9 or at step 10a1, or may be combined segment like step 10b1 or step 10b2) Then go to step 12.

Step-12: After checking of four segments if the percentage of pixel matches above threshold value then go to step13. Else go to step 15.

Step-13: Next match the neighbor pixels row wise and then column wise.

Step-14: If the total pixel matching is greater than the threshold value, then continue and go to step 13.
Else go to step 15.

Step-15: Go to the next DB Image, and again check it from step1.

III. EXAMPLE

For matching two fingerprint images, first we need to convert images (both DB Image and Final Image) into gray scale image and then binary image, then we can enter into step 1,because this algorithm is applicable on the binary format of images.

Step-1:
Derive number of pixel along length and width.
Here number of pixel along width= m = 40
Here number of pixel along length= n = 36

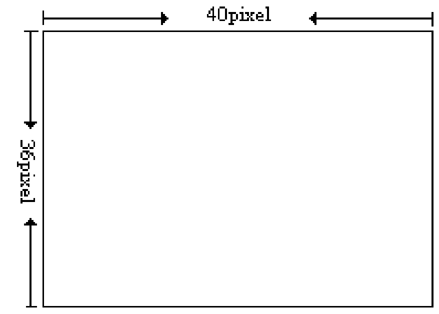

Fig.1. Derive number of pixels along width and length from final image

Step-2:
Derive the centre position of the image.
Here centre position= (p, q) = (m/2, n/2) = (40/2, 36/2)
= (20, 18).

Step-3:
Using centre position divides the image into four segments.

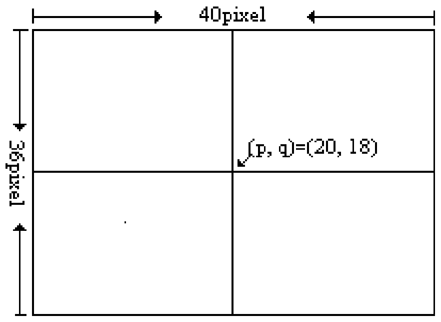

Fig.2. Devide the image into four segment using centre position

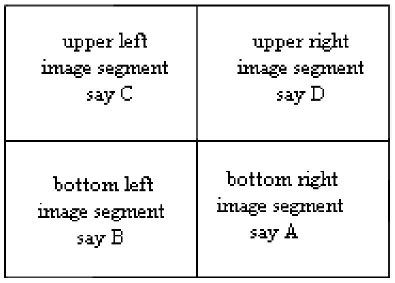
upper left image segment say C
upper right image segment say D
bottom left image segment say B
bottom right image segment say A

Fig.3. Identification of four image segments

Step-4:
Derive number of pixel along length and width of the image segments.

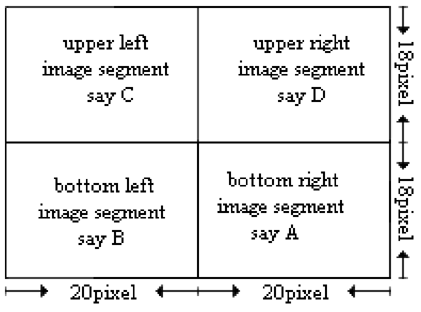
upper left image segment say C
upper right image segment say D
bottom left image segment say B
bottom right image segment say A
18pixel
18pixel
20pixel
20pixel

Fig.4. Derive number of pixels along length and width of segments

If the number of pixels along width or length is greater than 10 then go to step1 and repeat step2 and step3.
Here both in width and length number of pixel is greater than 10, so, repeat step1, step2 and step3. That means break the image segments into four segments again.

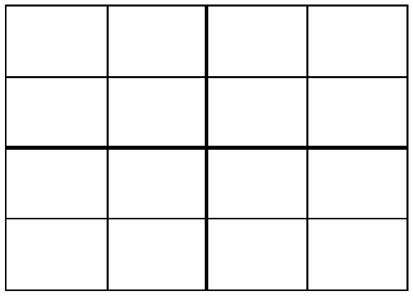

Fig.5. Break the image segments again into four segments

Step-5:
Take the upper left corner image segment (say A_1) of the previous image segment A, upper right corner image segment (say B_1) of the previous image segment B, bottom right corner image segment (say C_1) of the previous image segment C, bottom left corner image segment (say D_1) of the previous image segment D.

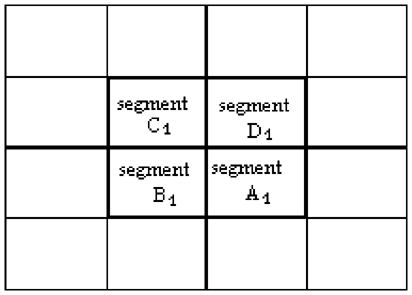

Fig.6. Break the segments again and take segment A_1 from A, B_1 from B, C_1 from C, D_1 frim D

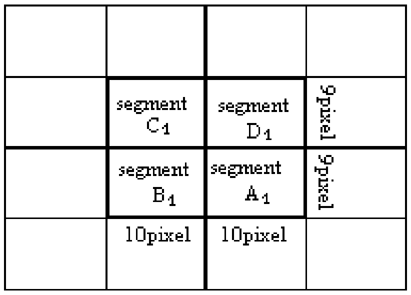

Fig.7. Check the number of pixel along width and length of segment A_1, B_1, C_1, D_1

Step-5a:
Here number of pixel along width and length of the image segments $A_{1,}$ $B_{1,}$ $C_{1,}$ D_1 is not greater than 10.
So we don't have to go to the step 4.

Step-5b:
Here number of pixel along width and length of the image segment $A_{1,}$ $B_{1,}$ $C_{1,}$ D_1 is less than and equal to 10.
i.e. for segment A_1 m=10 pixel, n=9pixel
So fix up the segment A_1 as final image segment.
Same technique for segment $B_{1,}$ $C_{1,}$ D_1 also.
So we have to go to the step 6.

Step-6:
Using final image segment $A_{1,}$ traverse one DB Image pixel by pixel in row wise and then column wise.

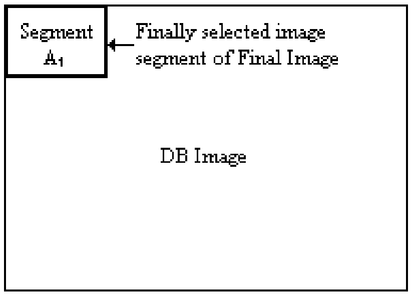

Fig.8. Traverse one DB Image by finally selected first image segment A_1 (first step of traverse)

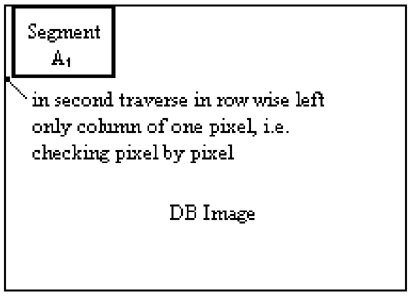
Segment A_1
in second traverse in row wise left only column of one pixel, i.e. checking pixel by pixel
DB Image

Fig.9. The second step of traverse by segment A_1

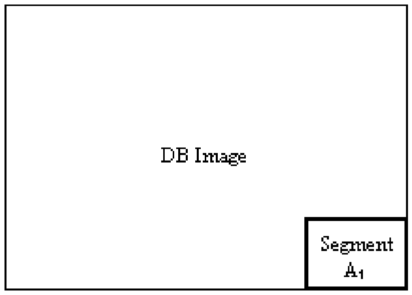

Fig.10. Final step of traverse by segment A_1

The same technique would be applied for segment $B_{1,}$ $C_{1,}$ D_1 also and one by one.

Step-6a:
For each traverse by segment $A_{1,}$ store (a, b, c), where
a= upper left most pixel position,
b= bottom right most pixel position and
c= percentage of pixel match at that traverse.

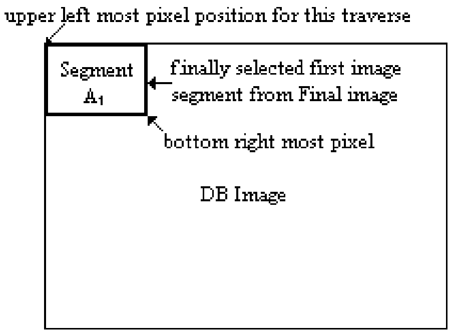
upper left most pixel position for this traverse
Segment A_1
finally selected first image segment from Final image
bottom right most pixel
DB Image

Fig.11. Store upper left most and bottom right most pixel position at that traverse by segment A_1

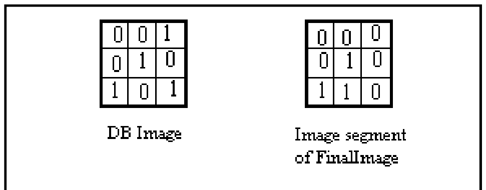

Fig.12. For calculating the pixel match

Here percentage of pixel match c = (6/9) * 100.
Step-6b:
For each traverse by segment $B_{1,}$ store (d, e, f), where
d= bottom left most pixel position and
e=upper right most pixel position,
f= percentage of pixel match at that traverse.

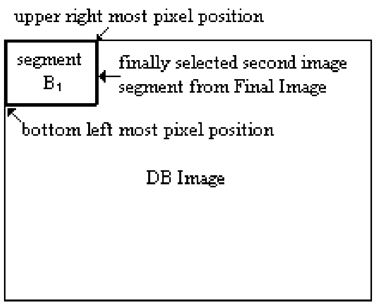
upper right most pixel position
segment B_1
finally selected second image segment from Final Image
bottom left most pixel position
DB Image

Fig.13. Store upper right most and bottom left most pixel position at that traverse by segment B_1

Step-6c:
For each traverse by segment $B_{1,}$ store (g, h, i), where
g= upper left most pixel position,
h= bottom right most pixel position and
i= percentage of pixel match at that traverse.

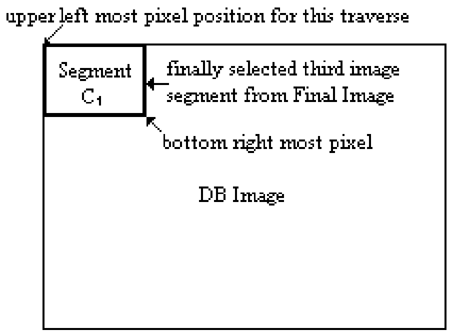
upper left most pixel position for this traverse
Segment C₁
finally selected third image segment from Final Image
bottom right most pixel
DB Image

Fig.14. Store upper left most and bottom right most pixel position at that traverse by segment C_1

Step-6d:
For each traverse by segment $B_{1,}$ store (j, k, l), where
j= upper right most pixel position,
k= bottom left most pixel position and
l= percentage of pixel match at that traverse.

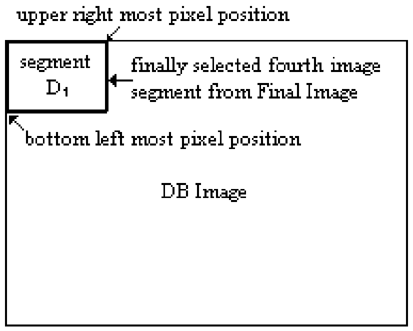
upper right most pixel position
segment D₁
finally selected fourth image segment from Final Image
bottom left most pixel position
DB Image

Fig.15. Store upper right most and bottom left most pixel position at that traverse by segment D_1

Step-7:
For any two traverse with A_1 store (a, b, c) where maximum pixel matching c is received,
For any two traverse with B_1 store (d, e, f) where maximum pixel matching f is received,
For any two traverse with C_1 store (g, h, i) where maximum pixel matching i is received,
For any two traverse with D_1 store (j, k, l) where maximum pixel matching l is received,
I.e. if for first step of traversing by A_1 the value of c=80% and for second step of traversing the value of c=86%,then store the left most pixel position of second step traverse, and also store the maximum percentage of pixel match.
In this way the maximum matched area from DB Image would be gained.

Step-8:
If any one of the maximum matching of pixel (i.e. any one of c, f, i or l) is greater than the threshold value which is previously given by coder, then go to step 9 that means next step.
Else no need to proceed further to check that Db Image so go to step15.

Step-9:
Using the pixel positions of those image segment/s (whose percentage of pixel matching is greater than the threshold value), fix up the pixel chunk/s in the DB Image. If there exists only one chunk then go to step11 directly. Else go to step11.

Step-10:
Check those fixed up image segments in the DB Image to find the intersecting point/s or line between them.

Step-10a:
If the intersecting point/s or line is not found, then check those image segments individually and let all the segments are above threshold value.

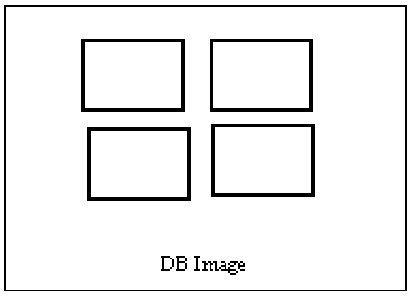

Fig.16. Intersecting point is not found from fixed up pixel regions

Step-10a1:
If all image segments are above threshold value then take the segment with maximum matching and go to step11.

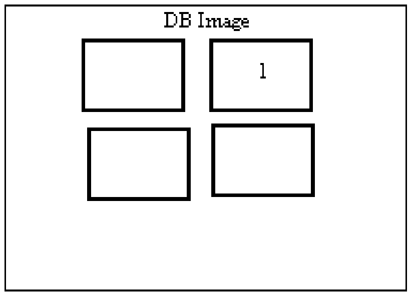

Fig.17. 1 labled image segment has maximum threshold value,so take only labled 1

Step-10b:
If the intersecting point/s or line are found, then check the intersecting image segments.

10b1: if all the image segments (those who have intersecting point/s or line) are above threshold value then combine those image segments and go to step11.

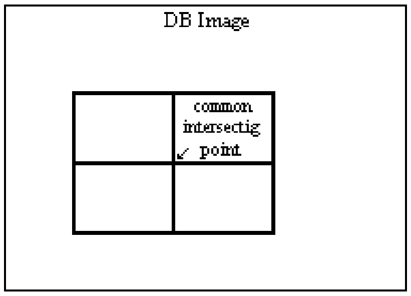

Fig.18. All image segments having threshold value and having at least one common point

10b2: if any two image segments (those who have intersecting point/s or line) is above the threshold value and the rest are below threshold value, then combine only those image segments which are above threshold value then go to step11.

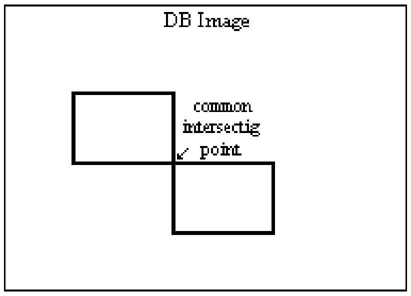

Fig.19. Two image segments above threshold value and having common intersecting point

Or

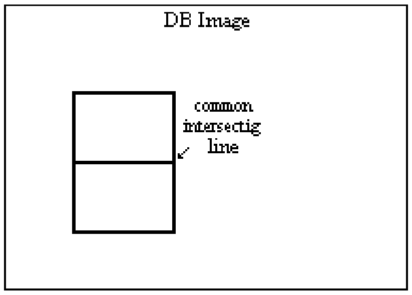

Fig.20. Two image segments above threshold value and having common intersecting line

10b3: if one of the image segments (those who have intersecting point/s or line) is above the threshold value then no need to combine that image segment with another, take only that segment above threshold value. Then go to step 11.

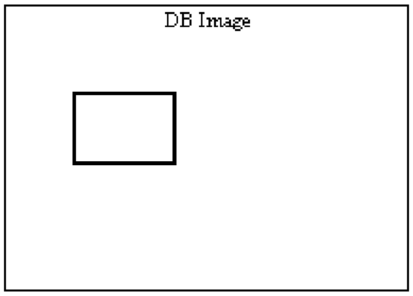

Fig.21. One image segment above threshold value, then take only that segment

Step-11:
Check the DB Image with the total area of four segment of the image segment (the segment may be single segment just like step 10a1 and step 10b3, or may be combined segment like step10a2 or step10b1 or step10b2) which has maximum. Then go to step 12.

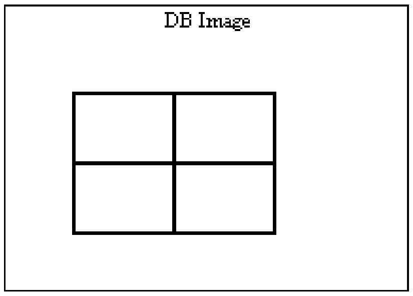

Fig.22. Check total area of four segment of one image segment

Step-12:
After checking of four segments if the percentage of pixel matches above threshold value then go to next step. Else go to step 15.

Step-13:
Next match the neighbor pixels row wise and then column wise.

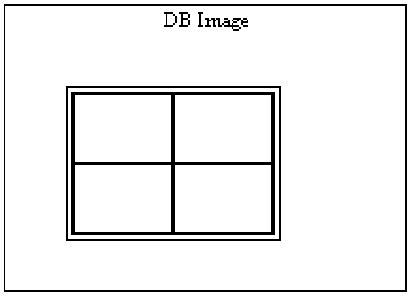

Fig.23. Check the DB Image by neighbouring pixel by row wise and then column wise

Step-14:
If the total pixel matching is greater than the threshold value, then continue and go to step 13.
Else go to step 15.

Step-15: If the percentage of pixel matching does not match above threshold value at Step8 or at Step12 or at Step14 then go to the next db image, and again check it from step1.

IV. ANALYSIS AND CONCLUSION

Failure during the traverse time of DB Image using the image segment, cannot lead to proceed for further checking of that DB Image. So here in this algorithm, we selected four image segments from different position nearby centre area instead of one image segment to improve the efficiency. It is actually helpful than one image segment, because, if one or two or any three image segments does not match then the rest may be matched above threshold value during the traverse of DB Image. So there exists better probability of matching during traverse time.

V. REFERENCES

[1] J. K. Mandal, S. Dutta, "A 256-bit recursive pair parity encoder for encryption", Advances D -2004, Vol. 9 nº1, Association for the Advancement of Modelling and Simulation Techniques in Enterprises (AMSE, France), www. AMSE-Modeling.org, pp. 1-14

[2] Pranam Paul, Saurabh Dutta, "A Private-Key Storage-Efficient Ciphering Protocol for Information Communication Technology", National Seminar on Research Issues in Technical Education (RITE), March 08-09, 2006, National Institute of Technical Teachers' Training and Research, Kolkata, India

[3] Pranam Paul, Saurabh Dutta, "An Enhancement of Information Security Using Substitution of Bits Through Prime Detection in Blocks", Proceedings of National Conference on Recent Trends in Information Systems

(ReTIS-06), July 14-15, 2006, Organized by IEEE Gold Affinity Group, IEEE Calcutta Section, Computer Science & Engineering Department, CMATER & SRUVM Project-Jadavpur University and Computer Jagat

[4] Dutta S. and Mandal J. K., "A Space-Efficient Universal Encoder for Secured Transmission", International Conference on Modelling and Simulation (MS' 2000 – Egypt, Cairo, April 11-14, 2000

[5] Mandal J. K., Mal S., Dutta S., A 256 Bit Recursive Pair Parity Encoder for Encryption, accepted for publication in AMSE Journal, France, 2003

[6] Dutta S., Mal S., "A Multiplexing Triangular Encryption Technique – A move towards enhancing security in ECommerce", Proceedings of IT Conference (organized by Computer Association of Nepal), 26 and 27 January, 2002, BICC, Kathmandu.

[7] Paul Reid, 2003, Biometrics for Network Security, Prentice Hall PTR, chapter-5

[8] A white paper by the University of Southern California and VeriSign 2005 Building a Security Framework for Delivery of Next Generation Network Services United States.

[9] L. Podio and Jeffrey S. Dunn 2002, Biometric Authentication Technology: From the Movies to Your Desktop, National Institute of Standards and Technology (NIST), Information Technology Laboratory 497

[10] Edited by Lori Ayre, Infopeople Project, 2003, Library Computer and Network Security Infopeople Project, http://infopeople.org/howto/security/.

[11] Sarbari Gupta, 2004, Identity Authentication Identity Authentication using the using the PIV Token PIV Token, National Institute of Standards and Technology, India.

[12] Secure Computing Corporation, 2001, Authenticating with one of the safest devices: the biometric Sony Puppy, Secure Computing Corporation, 4810 Harwood Road, San Jose, CA 95124 USA.

[13] Biometric Consortium web site:
http://www.biometrics.org 2006

[14] International Biometric Industry Association, http://www.ibia.org 2005

[15] Bioenable Technologies Pvt. Ltd. 2004-2005 http://www.bioenabletech.com/biometrics_india_pune_contact.htm

[16] Securitex Electronic Systems Engineering, 2006, Fingerprint Identification system http://www.securitex.com.sg/

[17] Manvish Embedded Services, 2006, Finger print sensors technology overview http://www.manvish.com/embedded/miFAUN/techoverview.php

[18] TopAZ Solutions Pte Ltd, 2006, Biometric Fingerprint Security, http://www.topazsol.com/bio_door_access.htm

[19] Sanjukta Pal, Dr. Pranam Paul, "Cryptographic protocol Depending on Biometric Authentication", published in International Journal of Engineering Science and Technology (IJEST), Vol. 5 No.02 February 2013, page no 354

VI. BOIGRAPHY OF AUTHORS

Sanjukta Pal is MTech (CST) final year student of Dr. B. C Roy Engineering College, Durgapur. She had completed MCA at West Bengal University of Technology.

Sucharita pal is an Assistant Professor of Asansol Engineering College, Asnasol. She had completed her M.Tech in the field of Electrical Engineering at National Institute of Technology, Durgapur.

Dr Pranam Paul is an Associate Professor of Narula Institute Technology, Agarpara. He had completed his Ph.D from Electronic and Communication Engineering department of National Institute of Technology, Durgapur in the field of Cryptography and Network Security and master degree in Computer Application in 2005 under West Bengal University of Technology, INDIA. He has total 50 International Journal publications among total 60 publications, except this one.