



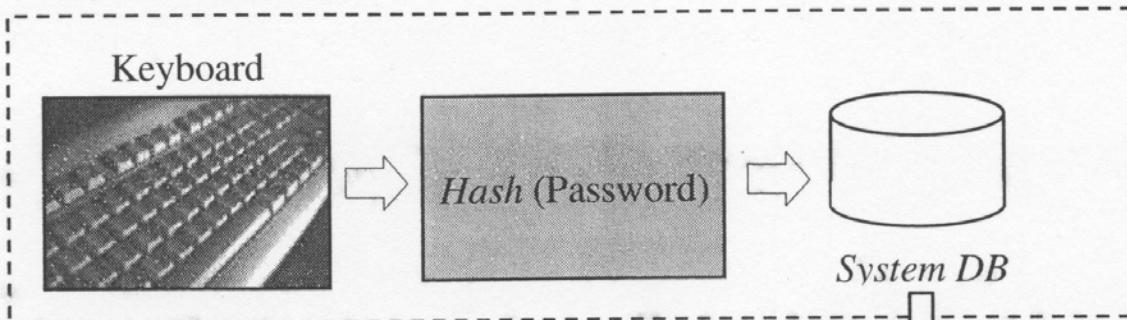
Symmetric hash functions for fingerprint minutiae

S. Tulyakov, V. Chavan and V. Govindaraju
Center for Unified Biometrics and Sensors
SUNY at Buffalo, New York, USA

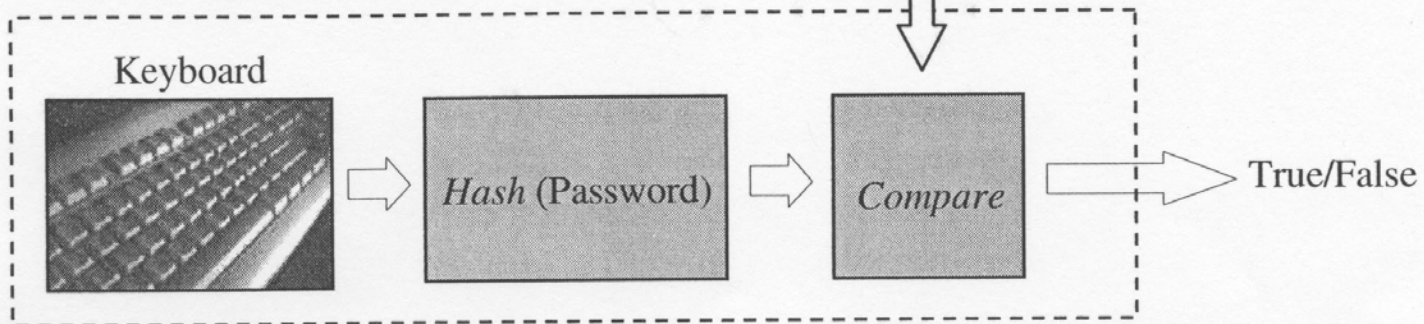


Securing password information

Password enrollment



Password verification



a)

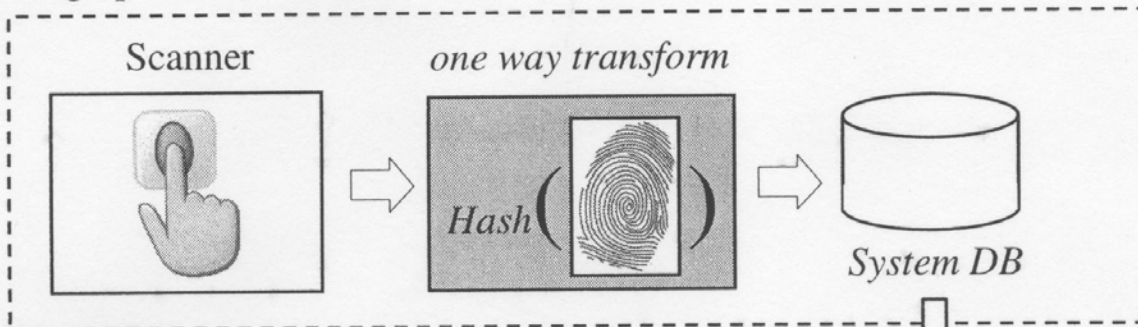
It is impossible to learn the original password given stored hash value of it.



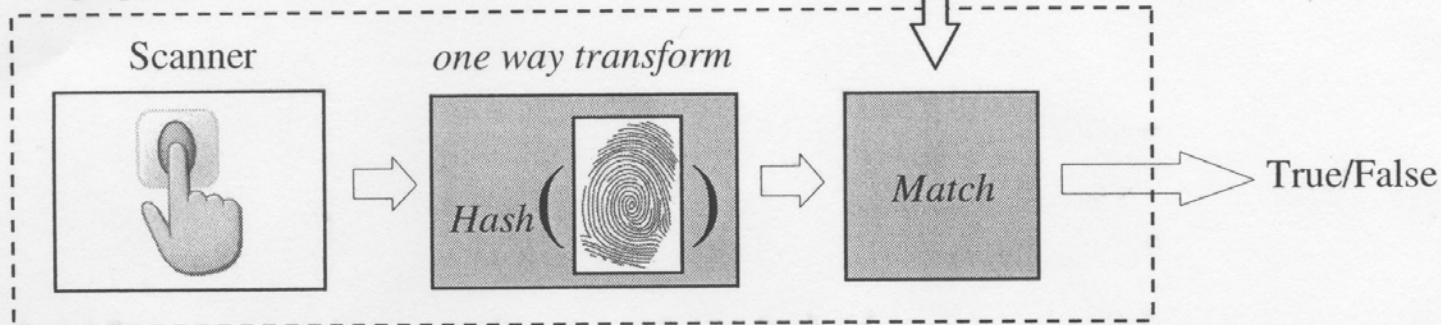
Securing fingerprint information

Wish to use similar functions for fingerprint data:

Fingerprint enrollment



Fingerprint verification

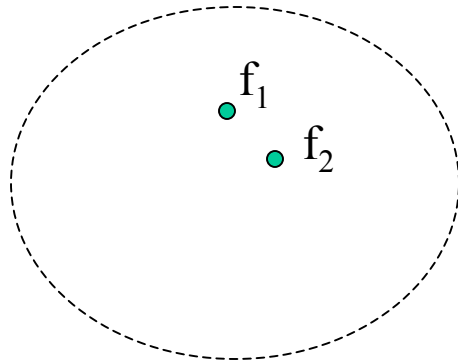


b)



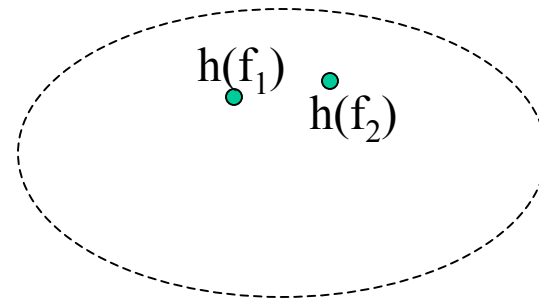
Obstacles in finding fingerprint hash functions

Fingerprint space



h

Hash space



Since match algorithm will work with the values of hash functions,

- similar fingerprints should have similar hash values
- rotation and translation of original image should not have big impact on hash values
- partial fingerprints should be matched



Minutia points of the fingerprint



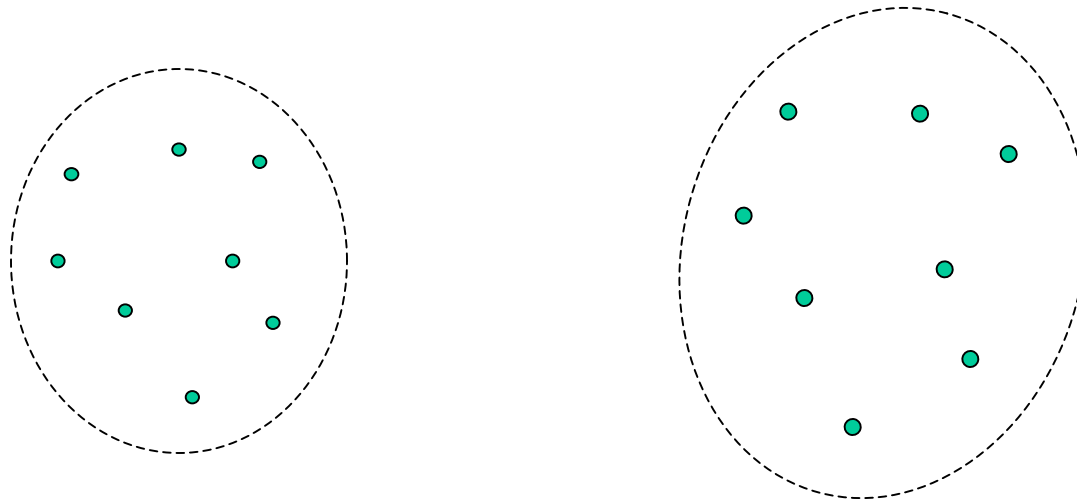
Minutia points - points where ridge structure changes: end of the ridge and branching of the ridge.

The positions of minutia points uniquely identifies the fingerprint.



Assumptions on minutiae sets extracted from the same finger

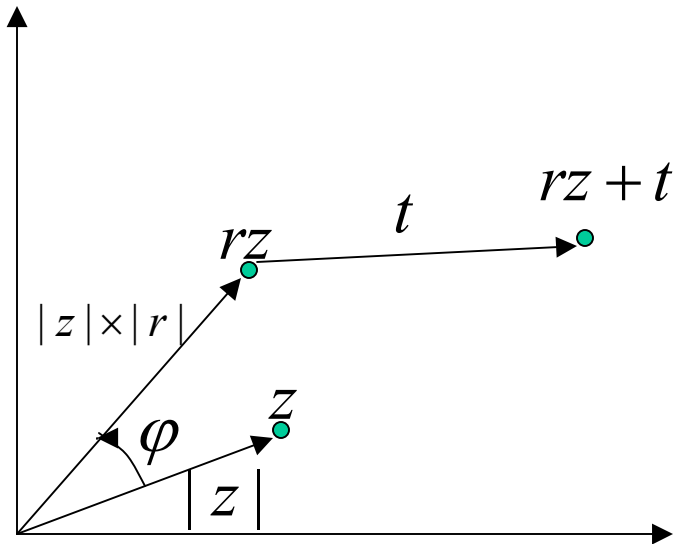
Assume that two fingerprints originating from one finger differ by scale and rotation. Thus the set of minutia points of one fingerprint image can be obtained from the set of minutia points of another fingerprint image by scaling and rotating.





Transformation of minutiae set

If we represent minutia points as points on a complex plane, then scaling and rotation can be expressed by function: $f(z) = rz + t$ where r is the complex number determining rotation and scaling, and t is the complex number determining translation of minutia point.



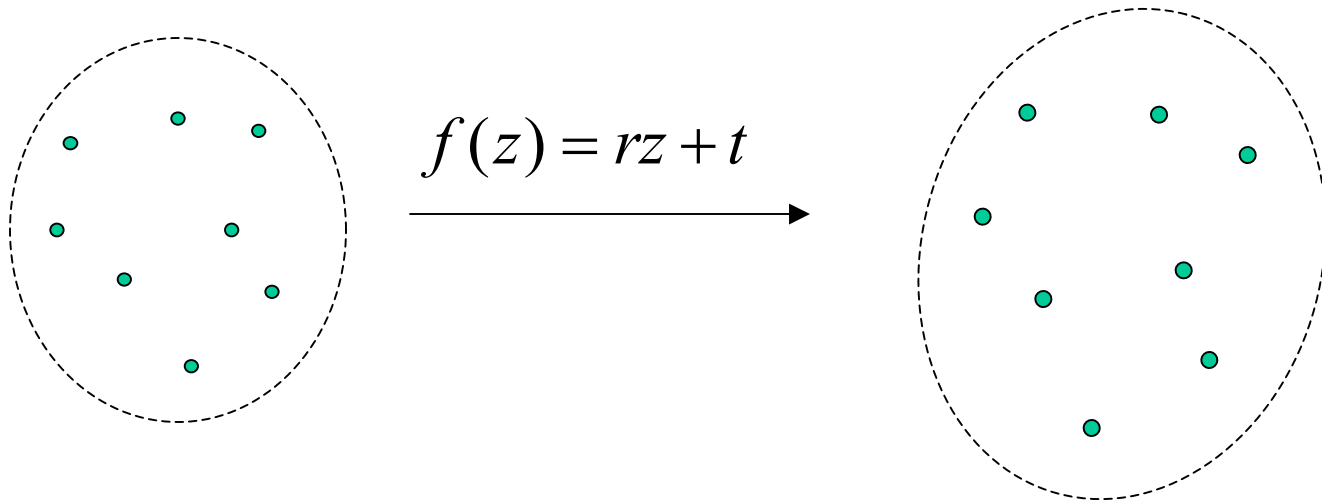
$$r = |r| (\cos \varphi + i \sin \varphi)$$

Multiplying by r means rotating by angle φ and scaling by factor $|r|$.



Transformation function

If (c_1, c_2, \dots, c_n) is a set of minutia points of first fingerprint and $(c'_1, c'_2, \dots, c'_n)$ is a set of minutia points of second fingerprint (same finger), then we assume that there is a transformation $f(z) = rz + t$ such that $c'_i = f(c_i) = rc_i + t$ for any $i = 1, \dots, n$.





Hash functions of minutia points

Consider following functions of minutia positions:

$$h_1(c_1, c_2, \dots, c_n) = c_1 + c_2 + \dots + c_n$$

$$h_2(c_1, c_2, \dots, c_n) = c_1^2 + c_2^2 + \dots + c_n^2$$

⋮

$$h_m(c_1, c_2, \dots, c_n) = c_1^m + c_2^m + \dots + c_n^m$$

The values of these symmetric functions do not depend on the order of minutia points.



Hash functions of transformed minutiae

What happens with hash functions if minutia point set is transformed?

$$\begin{aligned}h_1(c'_1, c'_2, \dots, c'_n) &= c'_1 + c'_2 + \dots + c'_n \\ &= (rc_1 + t) + (rc_2 + t) + \dots + (rc_n + t) \\ &= r(c_1 + c_2 + \dots + c_n) + nt = rh_1(c_1, c_2, \dots, c_n) + nt\end{aligned}$$

$$\begin{aligned}h_2(c'_1, c'_2, \dots, c'_n) &= c'^2_1 + c'^2_2 + \dots + c'^2_n \\ &= (rc_1 + t)^2 + (rc_2 + t)^2 + \dots + (rc_n + t)^2 \\ &= r^2(c^2_1 + c^2_2 + \dots + c^2_n) + 2rt(c_1 + c_2 + \dots + c_n) + nt^2 \\ &= r^2h_2(c_1, c_2, \dots, c_n) + 2rth_1(c_1, c_2, \dots, c_n) + nt^2\end{aligned}$$



Finding transformation parameters from hash function values

Thus $h_i(c'_1, c'_2, \dots, c'_n)$ can be expressed as a linear combinations of $h_j(c_1, c_2, \dots, c_n)$, $j \leq i$ with coefficients depending on transformation parameters r and t .

Denote:

$$h_i = h_i(c_1, c_2, \dots, c_n)$$
$$h'_i = h_i(c'_1, c'_2, \dots, c'_n)$$

Thus

$$h'_1 = rh_1 + nt$$
$$h'_2 = r^2 h_2 + 2rth_1 + nt^2$$

And r, t can be calculated given h_1, h_2, h'_1, h'_2



Verifying fingerprint match using hash functions

When r and t are found we can use higher order hash functions to check if fingerprints match.

For example, if extracted minutia set is identical to the stored in the database, then for the hash function of third order we should

get:

$$h'_3 = r^3 h_3 + 3r^2 t h_2 + 3r t^2 h_1 + n t^3$$

The difference between two parts of above equation can serve as a confidence measure for matching two sets of minutia points. Subsequently, similar confidence measures for hash functions of other orders should be combined in one confidence measure.



Fingerprint matching algorithm

- Enrollment:
 1. Minutia positions are extracted (n positions)
 2. K symmetric hash functions are evaluated and results are stored in the database.
- Matching:
 1. Minutia positions are extracted (n positions)
 2. K symmetric hash functions are evaluated and passed to the server for matching.
 3. Using values of first two hash functions (stored in the server database and just extracted) the transformation parameters r and t are found.
 4. Remaining $K-2$ hash function values are used to verify minutia set matching.



Privacy issues

- If the number of stored hash functions h_i is less than the number of minutia points, it is not possible to find the positions of minutia points C_j . This also implies that it is possible for two different sets of minutia points to have same hash function values. To prevent such mismatches, we might want to consider multiple subsets of minutia points for matching.
- If fingerprint database is compromised, the different set of symmetric hash functions should be chosen. Need more research here.
- Similarly, different set of hash functions can be chosen for each individual, resulting in cancelable fingerprint templates.



Practical considerations for matching localized hash values

- The small changes in locations of minutia points result in big changes of symmetric functions of higher orders. Thus we limited ourselves to the symmetric functions of 1st and 2nd orders.
- Since direction of the minutia (direction of the ridge where minutia is located) is also important in fingerprint matching, we consider unit direction vectors and same hash functions of that vectors:
$$hd_i(d_1, d_2, \dots, d_n) = d_1^i + d_2^i + \dots + d_n^i$$

- The matching of hash values can be reformulated as a problem of minimizing (with respect to r and t) some distance function:

$$\begin{aligned} dist(h_i, h'_i, r, t) = & \alpha_1 | h'_1 - rh_1 - nt | \\ & + \alpha_2 | h'_2 - r^2h_2 - 2rth_1 - nt^2 | + \dots \end{aligned}$$



Subsets of minutia points.

- Since it is rare situation when two fingerprint images contain same minutia points, consideration of subsets of minutia points is required.
- Due to the privacy issues we need to have less symmetric functions than points in the minutia subset
- Considered configurations: 2 minutiae & 1 function
3 minutiae & 1 function
3 minutiae & 2 functions
- Matching of two fingerprints consists in matching all possible localized minutiae subsets, seeing how many subsets are matched and whether values of r and t are similar.

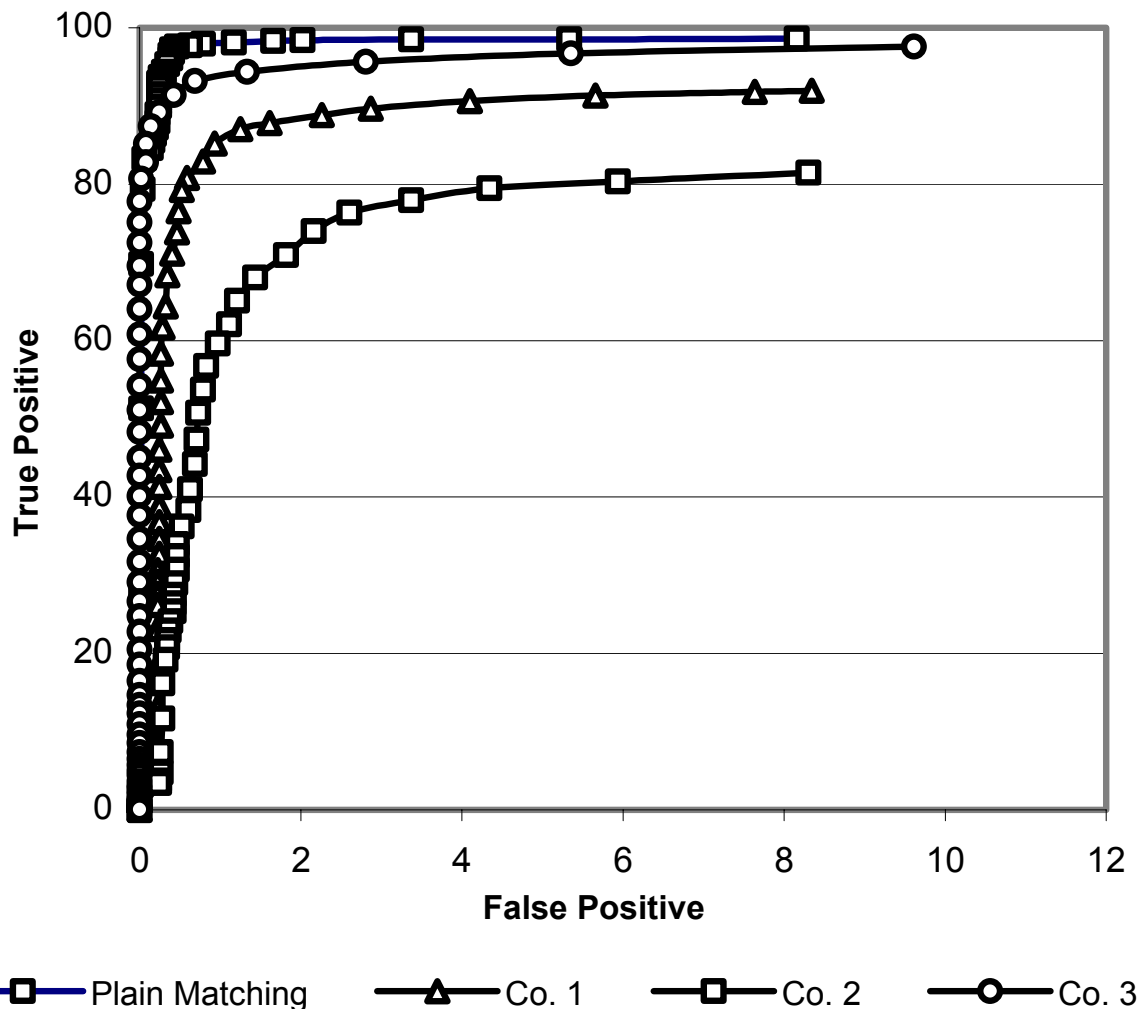


Partial fingerprint matching algorithm

- Enrollment:
 1. For each minutia point k , $1 \leq k \leq K$ of the fingerprint find its 2 nearest neighbors.
 2. Evaluate hash functions $(h_1(k), h_2(k), hd_1(k), hd_2(k))$ and store results in the database.
- Matching:
 1. As for enrollment, evaluate $(h'_1(l), h'_2(l), hd'_1(l), hd'_2(l))$ hash functions for all minutia points l , $1 \leq l \leq L$
 2. For each pair of hash values find the confidence of their match together with rotation angle best fit for this match.
 3. Consider the set of matching angles and determine if there are any clusters in this set. The presence of the cluster would indicate that there were many matches with particular rotation angles, hence same fingerprints.



Experimental results



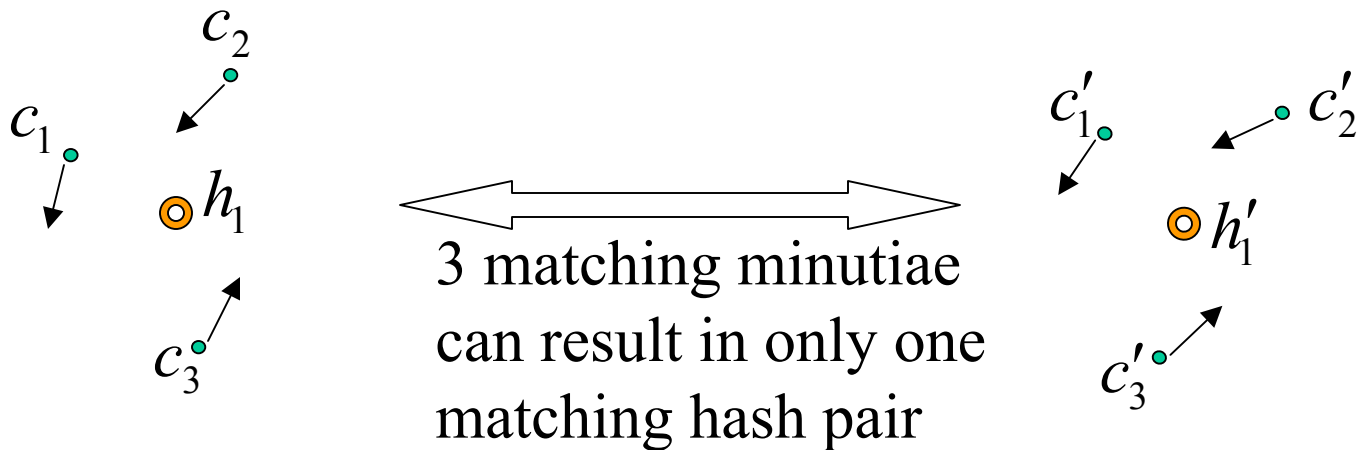
Using 3 minutiae in a subset and 2 hash functions gives the best performance so far with equal error rate of 3%, while original no-hash matching has equal error rate of 1.7%,

(tested on FVC2002 set, with 2800 genuine tests and 4950 impostor tests)



Algorithm limitations

Usually there are less matching hash values than matching minutiae. This means bigger difficulty in producing good match score, and setting match thresholds.





Thank you !

References

1. Davide Maltoni, Dario Maio, Anil K. Jain and Salil Prabhakar, *Handbook of Fingerprint Recognition*, Springer-Verlag, New York, 2003
2. Colin Soutar, Danny Roberge, Alex Stoianov, Rene Gilroy and B.V.K. Vijaya Kumar, “Biometric Encryption”, in *ICSA Guide to Cryptography*, R.Nichols, ed. (McGraw-Hill, 1999)
3. G.I. Davida, Y. Frankel, and B.J. Matt. “On enabling secure applications through offline biometric identification”. In *IEEE Symposium on Privacy and Security*, 1998.
4. Tsai-Yang Jea, Viraj S. Chavan, Venu Govindaraju and John K. Schneider, “Security and matching of partial fingerprint recognition systems”, In *SPIE Defense and Security Symposium*, 2004.