Towards Fingerprints as Strings: Secure Indexing for Fingerprint Matching*

Jesse Hartloff Jimmy Dobler Sergey Tulyakov Atri Rudra Venu Govindaraju
University at Buffalo, SUNY
Dept. of CSE , Buffalo, NY, 14260
{hartloff, jdobler, tulyakov, atri, govind}@buffalo.edu

Abstract

This work, for the first time, combines fingerprint matching, security, and indexing in one system. We use the fuzzy vault construct with minutia path information to achieve this. Since we only store translation and rotation invariant information from the paths, we achieve pre-alignment for "free," which is a requirement of the fuzzy vault. We introduce a hash table that allows indexing to be used with our system, which effectively reduces the search space of the database. Our system is secure in the sense that if the database is compromised, it will be computationally infeasible for the intruder to recover the enrolled fingerprint templates.

1. Introduction

Strings are used widely in many identification schemes such as passwords primarily because of three key properties. (1) *Matching* between any two pairs of strings is very efficient. (2) There exist *secure* hashes that can be used to store the strings. (3) Strings allow for efficient *indexing* structures that permit quick matching of a target string against a database of strings. However, in applications such as passwords, strings can be difficult for users to remember.

Biometrics such as fingerprints have the obvious advantage of being hard to "forget." Ideally, we would like to replace strings with fingerprints in authentication applications. To successfully do this, we must be able to replicate the three properties above with fingerprints. Unfortunately (or fortunately for us researchers!), fingerprints are an inherently more difficult medium to work with. In particular, unlike strings where the authentication protocol can demand the same string be reproduced, it is impossible for a given physical finger to produce exactly the same reading each time it is scanned. This has led to a fair amount of research on matching fingerprints [14, 15]. Somewhat more recently, there has been research in designing secure hash functions for fingerprints [16, 8, 18], and indexing for fingerprints has also been considered [9, 7, 22]. However, there has not been much work in achieving more than one of these three goals *simultaneously*. Recently, there was a study about the security and matching performance of the so-called fuzzy vaults of [16] in [13]. However, no existing work has considered all three axes together. Our main contribution in this paper is, to the best of our knowledge, *the first study of performance of fingerprints on the axes of matching accuracy, security and indexing* simultaneously.

Our starting points are two different schemes that have been proposed for secure hashes and indexing. The first is the fuzzy vault, proposed by Juels and Sudan [16], which can match any two *sets* with sufficiently large intersection and has been extended by [2]. This has been implemented in the biometrics literature with the sets being the set of minutia points [25, 23]. One of the major difficulties of this approach has been the inability of the fuzzy vault to handle translations and rotations of fingerprints which has inspired work using geometric hashing with the fuzzy vault [11, 20]. The second scheme is inspired by that of Tan et al. [28] and Mansukhani et al. [22] that index the fingerprints by storing information about paths. In particular, the indexing algorithm uses this path information to narrow down the search space on which it performs the final matching. *Our main technical contribution is a natural combination of these two schemes to attain both security and fast indexing.* In particular, from the hashing perspective, we first convert the minutia points into a set of paths and then apply the fuzzy vault to this latter set. We would like to point out that the path information that we store are distances and certain angles, which are *invariant* under translation and rotation. This automatically takes care of the shortcomings of previous fuzzy vault implementations.

Finally, we present *a thorough experimental evaluation of our system on all three aspects of matching, security and indexing*. To measure the matching performance we use the traditional notions of FAR and FRR to report ROC curves. To measure the efficiency of indexing we measure the *penetration coefficient* [7], which roughly speaking measures on

*Research supported by NSF grant TC-1115670.

average how much indexing reduces the brute-force search space. We use the security parameter from [13] to measure the security we get from the fuzzy vault. (This security parameter is in turn based on a hardness predicate proposed in the cryptography community [17].) We concede that, individually, our matching and indexing numbers do not beat the existing state-of-the-art results. However, we would like to point out that we expect to lose some performance when we gain security. Furthermore, indexing (and matching) inherently seems to require leaking information about the fingerprint, while secure hash functions try to obfuscate the fingerprint. Thus, when considering all three aspects together, we cannot expect to beat the existing results in any axis by itself.

2. Related Work

The matching between a single test template and the set of enrolled biometric templates is possible either by employing a fast matching algorithm, or with the help of some indexing structure reducing the number of attempted matches. For some biometric modalities, the fast matching algorithm and identifying the test template among millions of enrolled ones are feasible. For example, the iris template can be represented as a binary feature vector with the matcher calculating the distance between two templates by counting the number of differing bits. Using this approach, the iris biometric systems performing searches in large databases in real time have been implemented [6].

Some biometric modalities, such as face and hand geometry, allow templates to be represented as feature vectors with fixed dimensions. For such templates the multidimensional indexing in feature space is possible, e.g. by traditional k-dimensional tree structures. For example, in [24] a pyramid method was used to index 24-dimensional hand geometry biometric templates.

But many biometric modalities, e.g. fingerprints, voice, signature, etc., do not have fast matchers or fixed length feature vector representations. As the focus of our research, fingerprints are typically represented as variable size sets of minutia point coordinates. Since there is no well defined center or orientation of the fingerprint, the matching algorithm should investigate all possible affine transformations between two prints. As a result, it takes a relatively long time of up to 1 second to perform a single match between two fingerprint templates [4], and performing searches in large databases of millions of fingerprints without any indexing would be impractical.

The Henry classification system separates fingerprints into 5 classes (left and right arcs or loops and whorl), and many researchers proposed algorithms to perform automatic classification into such classes (e.g. [26]). Since 5 classes do not provide sufficient reduction of search space, it was also proposed to filter database templates by comparing them to the features used for classification, e.g. direction field [21]. Effectively such approaches extract a fixed length feature vector and use it for initial matching or filtering, which is followed later by traditional matching.

A different approach to fingerprint indexing is based on the multidimensional indexing technique [3], such as geometric hashing. Germain et al. applied such indexing to fingerprints using minutia triplets [9]. From each minutia triplet some features are extracted and quantized; depending on concatenated binary vector of features, a triplet is placed in the corresponding bucket. During matching, a test fingerprint will have minutia triplets occupying similar buckets and by finding the list of fingerprints having triplets in the same buckets as a test fingerprint we would be able to find the matching fingerprint. The algorithm was later improved by experimenting with different features and was shown to outperform classification based methods [28].

The multidimensional indexing algorithms should find a balance between the robustness of the feature extraction (so that corresponding features of two fingerprints would belong to the same buckets) and the number of buckets (as discussed in [3], each bucket should contain only a small number of triplets for better reduction of the matching time). In [22], a long vector of features is used for indexing, and matching a single bucket in two fingerprints is sufficient for retrieving an enrolled fingerprint. In this work, we utilize a smaller length feature vector for achieving such balance.

Among the variety of techniques for creating privacy preserving fingerprint templates, the fuzzy vault method relies on set matching and seems to be most compatible with the feature set indexing [9]. In the current paper we attempt to combine both methods. The fuzzy vault has been applied to fingerprints without indexing [25, 23] and attempted to be used along with indexing [1], though we show that the system proposed by Bohm et al. [1] has no provable security with the parameters chosen. Specifically, the secret can be recovered in polynomial time by inputting the entire vault into the Reed-Solomon decoding algorithm by Guruswami and Sudan [12]. The features employed for indexing in [9] should be independent of the affine transformations, and the fuzzy vaults based on such features have been proposed as well, either by explicitly extracting transformation invariant features [19] or by using geometric hashing construction [11, 20]. Since we need to use transformation invariant features for the creation of the indexing hash tables, and have an additional large number of fuzzy vault chaff points for enrollment, the restrictions on the number of buckets [3] suggest the use of complex indexing features different from those used in above papers. Thus we utilized the complex transformation invariant indexing features of [22] extracted from the paths of neighboring minutiae, which proved to be sufficient for our task.

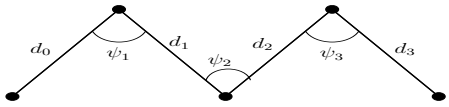

Figure 1. A path is represented by a single value z given by a concatenation such that $z = d_0d_1d_2d_3\sigma_0\sigma_1\sigma_2\sigma_3\psi_1\psi_2\psi_3$. The d_i values are the distances between points and the ψ_i's are inside angles as shown. The σ_i's are the difference in orientation between minutia points and can range from 0 to 180 degrees.

3. System Details

Our system is based on fuzzy vaults locked using path information from the fingerprint minutia points. It populates a hash table by enrolling fingerprints, then uses this table for indexing when searching for a match. We (1) explain how we obtain the paths, (2) how the systems enrolls fingerprints and (3) how it searches the hash table for matches.

Paths. We use a set of paths of minutia points to represent a fingerprint in our system. Paths have the advantage of being rotation and translation invariant which is critical since we are using the fuzzy vault construct which requires the fingerprints to be pre-aligned.

To convert a fingerprint template to a set of paths, we choose each minutia point as the start to a path p_0 exactly once to obtain n paths from n points. The next point in a path p_{i+1} is the Euclidean closest point to p_i not already in the path. This is repeated until a path of length 5 is found.

Each path is then represented in the finite field with 2^{23} elements (denoted by $\mathbb{F}_{2^{23}}$) by concatenating 11 values extracted from the path. These values are d_i for the Euclidean distance and σ_i for the difference in minutia orientation($\Delta\theta$) between p_i and p_{i+1}. The other values are ψ_i which is the inside angle formed by points p_{i-1}, p_i, and p_{i+1} (see figure 1). We quantize and concatenate all of these values to map the paths to values in $\mathbb{F}_{2^{23}}$ as $z = d_0d_1d_2d_3\sigma_0\sigma_1\sigma_2\sigma_3\psi_1\psi_2\psi_3$. Each value is given a predetermined number of bits which defines the amount of quantization. In our implementation, no value is given more than four bits so there are at most sixteen possibilities for a given value. This z value is the bin number in the hash table corresponding to that path. We also quantize in a way that distributes the z values close to uniformly over $\mathbb{F}_{2^{23}}$ as described in [13] which is a requirement for security in fuzzy vaults [17].

Enrollment in the hash table. The hash table is constructed to have one bin for each element in the field, and each bin will contain a set of (ID, γ) pairs where ID uniquely identifies a fingerprint and $\gamma \in \mathbb{F}_{2^{23}}$. A fingerprint is enrolled into the hash table by using its path information to determine which bins it will be inserted into. There is a one-one correspondence between the 2^{23} possible paths and the 2^{23} bins in the hash table.

To enroll a fingerprint f, we lock a secret polynomial P_s of degree $k-1$ over $\mathbb{F}_{2^{23}}$ using the fuzzy vault construct [16] and insert the points of the vault into the hash table. To accomplish this, we first find the z values that correspond to the paths in f. For each of the z values, we insert an (ID, γ) pair into bin z such that $\gamma = P_s(z)$. For testing purposes, P_s is randomly generated, though in practice this would be any sensitive information such as an encryption key.

After all the genuine points are inserted, we add 50,000 chaff points by choosing z and γ uniformly at random under the conditions that $\gamma \neq P_s(z)$. All of the genuine and chaff points for a fingerprint will have the ID of the fingerprint being enrolled. If all the points with the same ID are extracted from the hash table, it would represent a traditional fuzzy vault. The chaff points serve the purpose of obfuscating the data so an intruder can not determine which paths correspond to the actual enrolled fingerprint [16]. The intuition is that the genuine user will be able to recover the paths from a new reading of the same fingerprint to find the bins containing the genuine pairs.

During the enrollment process, if any z values are repeated, the duplicate is rejected so each fingerprint will be in each bin at most once.

In our implementation, we used multiple fingerprint readings for enrollment. We performed testing on the FVC2002 databases which contain eight reading for each fingerprint. We chose six readings for enrollment, then tested on the remaining two. For multiple enrollment, we added all of the paths found from the six readings to the hash table with the same ID. That ID is then that fingerprint's ID and is used to check for genuine matches. We can not enroll each of the six readings with different ID's as this would expose a known weakness of fuzzy vaults [27]. Our testing shows that this results in an average of 245 unique paths for each enrollee.

Searching the hash table. To make use of the indexing structure, we must be able to search the hash table and quickly determine which fingerprints are likely to be genuine. To accomplish this for a test fingerprint reading f', the first step is to compute all its paths to find the set of z values for f', just as we did for enrollment. We then extract all the (ID, γ) pairs from the bins pointed to by these z values. We score each enrolled ID by counting the number of points extracted labeled with that ID and rank all ID's according to this score. This gives a list of fingerprint ID's sorted by the number of matched paths with the test reading. We then compute the r value as how far down this list the genuine fingerprint is. This value is used to find the penetration coefficient of this scheme which we discuss in section 4.

4. Evaluation

To evaluate our system, we present various results to measure the indexing efficiency, matching accuracy, and security of the system. These metrics should all be considered simultaneously to properly assess the overall effectiveness of the system.

Metrics. To measure indexing, we use the time taken to search the hash table for a single test reading, the hit rate vs. penetration rate, and the penetration coefficient. We define the penetration coefficient to be

$$P = r/N$$

where r is the average rank of the genuine fingerprint using our ranking scheme, and N is the total number of enrolled fingerprints. This metric relates to how much the search space is reduced by using our indexing scheme. Observe that given random rankings, the expected value of P is 0.5, not $P = 1$. Therefore, 0.5 should be considered the worst-case measure of P. The penetration coefficient can be thought of as the average penetration rate. Informally, it is the area bounded above the curve in figure 2.

To measure the matching accuracy we use standard ROC, FAR, and FRR curves. To obtain a score between a test reading and an enrolled fingerprint, we find all the bins that the two share and observe whether each point is genuine or chaff. The score is then the number of genuine points minus the number of chaff points in this set. This score directly relates to the ability of that reading to unlock the fuzzy vault corresponding to that fingerprint. Specifically, the Welch-Berlekamp decoder can unlock the vault iff this score is greater than or equal to the size of the secret polynomial (k). These matching results represent the ability to match fingerprints using only path information in the presence of chaff points. To obtain this score in testing, the chaff points are marked beforehand. This will not be the case in practice and our indexing results do not use these markings. We have tested our system by running the Welch-Berlekamp decoder and verified the predicted behavior. Running the decoder also made no significant impact on the runtime of our system by running in less than 1ms on all trials.

The security is measured by a parameter λ which is defined as

$$\lambda = \sqrt{nk} - t$$

where t is the number of genuine minutia points used to lock the vault and n is the total size of the vault (sum of genuine and chaff points). Since t and n will vary between fingerprints, we use the average number of genuine points for both, which is $t = 245$ making $n = 50,245$. We note that λ is a logarithmic measure of security and is somewhat analogous to the number of bits of security. Any attack on the vault will take exponential time in terms of λ. Full justification for the use of λ can be found in our previous work [13] which is based on the hardness of Reed-Solomon decoding which has been proposed to be a cryptographic primitive [17]. In brief, any efficient attack on a fuzzy vault with uniformly distributed genuine points can be used as an efficient Reed-Solomon decoder which is believed to have exponential time complexity in λ. We note that any fuzzy vault implementation with $\lambda \leq 0$ can be cracked in polynomial time by applying the Reed-Solomon decoding algorithm by Guruswami and Sudan [12].

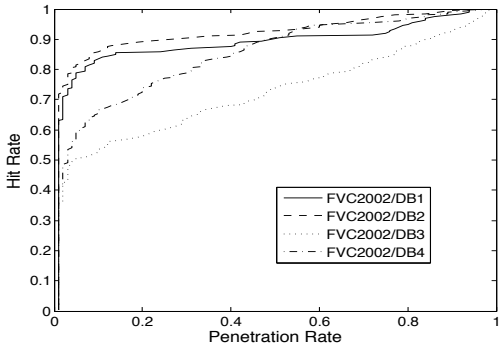

Figure 2. The fraction of the tested fingerprints that find their genuine match (hit rate) for a given fraction of the database searched in order of the ranking results from the hash table (penetration rate).

Results. We conducted experiments on the four databases from the Second International Fingerprint Verification Competition (FVC2002) (http://bias.csr.unibo.it/fvc2002/). The minutia positions were obtained by finding large curvature points on the contours extracted from the binarized fingerprint images [10]. Additionally, to improve the performance of minutia extraction, the fingerprints were enhanced by the short time Fourier transform algorithm [5]. The impostor scores and indexing results were found by enrolling every fingerprint using the first six readings from the databases and testing using the remaining two readings giving a total of 19800 impostor scores. The genuine scores were calculated separately by enrolling each fingerprint 28 times (once for each six reading combination) and testing with all readings. The matching score was taken from every enrollment and test pair from the same fingerprint such

Table 1. The indexing performance of our system on the FVC2002 databases. For DB1 and and DB2, about 10% of the database needs to be searched on average to find the genuine fingerprint. Even on DB3 where our system did not perform as well, it still cuts the search to $\frac{1}{4}$ of the database.

Penetration Coefficient (P)			
DB1	DB2	DB3	DB4
0.1031	0.0898	0.2582	0.1701

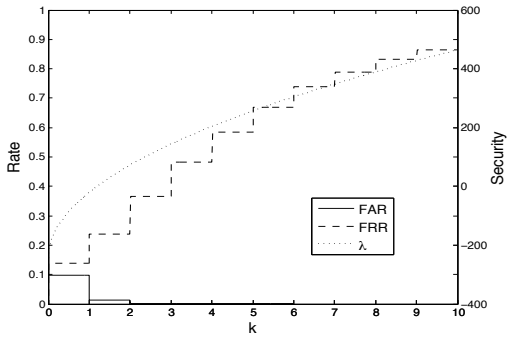

Figure 3. Rates and security for FVC2002 Database 1 with 50,000 chaff points per enrollee. The security parameter λ was computed using the average number of genuine minutia which is $t = 245$.

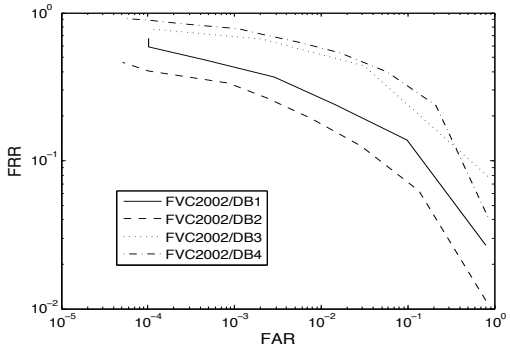

Figure 4. ROC curves for experiments on the four databases from FVC2002. The rates represent the matching performance of this system if it were used directly for fingerprint verification using the path information. The EER's are .132, .090, .313, and .231 for DB1, DB2, DB3, and DB4 resp. We note that these EER values are not very useful since they all occur at $k \leq 0$ meaning that there is no security at this value.

that the test print was not one of the six used to enroll. This results in 5600 genuine scores. Throughout all the tests, 50,000 chaff points were used for each enrollment.

The effectiveness of the indexing aspect of the system can be seen in table 1 and figure 2. Table 1 shows that for DB1 and DB2 about 10% of the database would need to be searched on average to find the genuine match. By figure 2, we can see that a large number of the genuine matches are at the top of the ranking list after searching the hash table. For DB1, 66% of the tests returned the genuine fingerprint as the first one in the ranking. This implies that by simply returning the first result of the rankings every time, the fingerprint would be correctly identified two thirds of the time using DB1.

The security and matching performance of our system can be seen in figures 3 and 4 and table 2. The security parameter λ is shown alongside FAR and FRR to illustrate the tradeoff between security and matching. Since λ is a logarithmic measure of security, a rather secure system can be achieved even with polynomials with only two terms (lines) where $\lambda = 72$. For more security, a polynomial with more terms can be used which also reduces the FAR while increasing FRR.

Figure 5 shows that a single hash table lookup can be performed very fast which is essential for an indexing scheme. Combined with the penetration coefficients from table 1, for DB1, on average we can eliminate 90% of the search space in less than 1ms with 10,000 enrolled fingerprints.

Timing tests were performed on a machine with a 2.4Gh processor and 32Gb of DIMM RAM clocked at 1333 MHz. Currently our program resides exclusively in RAM which does improve data access time. Our future work will test larger databases residing on a hard drive to perform timing tests that include disk access time.

Table 2. Results for common setups in secure systems where k represents the number of term in the secret polynomial. The security parameter λ was computed using 245 genuine points and 50,000 chaff points. We note that λ is a logarithmic measure of security so the security grows exponentially with λ.

		DB1		DB2	
k	λ	FRR	FAR	FRR	FAR
2	72.0	0.2386	0.0145	0.1261	0.0291
3	143.2	0.3654	0.0029	0.1920	0.0077
4	203.3	0.4816	0.0005	0.2648	0.0023

5. Conclusion

In this paper we presented an algorithm for indexing privacy preserving fingerprint templates. In essence, the algorithm combines the geometric hashing indexing technique with the fuzzy vault method for creating privacy preserving templates. Such combination is achieved by filling the hash table with additional chaff elements, which still al-

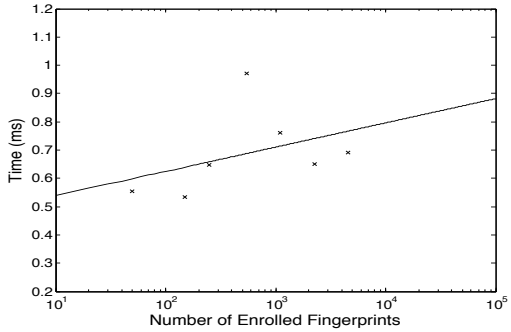

Figure 5. Time taken to search the hash table for a fingerprint template(given as a set of minutia points) and return a list ranking the candidate matches. Actual data points are denoted by 'X' and the best fit line was computed by assuming a logarithmic dependence on the number of enrollees. All tests were conducted with the entire hash table residing in RAM.

lows for the indexing part to function properly, but makes the retrieval of non-chaff elements by the intruder impossible. The experiments show the acceptable performance of the proposed method with respect to indexing, as well as matching accuracy. In addition, it has been shown that the algorithm has proper security characteristics.

Since the search in the hash table involves the matching of test templates simultaneously against all enrolled fingerprints, it does not seem possible to utilize any fingerprint alignment information at this stage. As a result, the transformation invariant features extracted from the paths of neighboring minutia are used in the system. Using such transformation invariant features might not deliver the best possible performance with respect to matching accuracy. Thus, it could be desirable to use a secondary matcher for verifying the results of the indexing, and we see it as a direction for future research. A challenge of constructing such a matcher is ensuring that its features do not reduce security when combined with the information from the hash table.

References

[1] C. Böhm, I. Färber, S. Fries, U. Korte, J. Merkle, A. Oswald, T. Seidl, B. Wackersreuther, and P. Wackersreuther. Efficient database techniques for identification with fuzzy vault templates. In *BIOSIG*, pages 115–126, 2011. 2

[2] J. Bringer, H. Chabanne, and M. Favre. Fuzzy vault for multiple users. In *AFRICACRYPT*, pages 67–81, 2012. 1

[3] A. Califano and R. Mohan. Multidimensional indexing for recognizing visual shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(4):373–392, 1994. 2

[4] R. Cappelli, D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain. Performance evaluation of fingerprint verification systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(1):3–18, 2006. 2

[5] S. Chikkerur, A. N. Cartwright, and V. Govindaraju. Fingerprint enhancement using STFT analysis. *Pattern Recognition*, 40(1):198–211, 2007. 4

[6] J. Daugman and I. Malhas. Iris recognition border-crossing system in the UAE. *Int. Airport Review*, (2), 2004. 2

[7] J. de Boer, A. M. Bazen, and S. Gerez. Indexing fingerprint databases based on multiple features. In *ProRISC Workshop on Circuits, Systems and Signal Processing*, 2001. 1

[8] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. 1

[9] R. Germain, A. Califano, and S. Colville. Fingerprint matching using transformation parameter clustering. *IEEE Computational Sci. and Engineering*, 4(4):42–49, 1997. 1, 2

[10] V. Govindaraju, Z. Shi, and J. Schneider. Feature extraction using a chaincoded contour representation of fingerprint images. In *4th international conference on Audio- and video-based biometric person authentication*, volume 2688, pages 268–275, Guildford, UK, 2003. Springer-Verlag. 4

[11] X. Q. Guo and A. Q. Hu. The automatic fuzzy fingerprint vault based on geometric hashing: Vulnerability analysis and security enhancement. *Int. Conference on Multimedia Information Networking and Security*, 1:62–67, 2009. 1, 2

[12] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *FOCS*, pages 28–39, 1998. 2, 4

[13] J. Hartloff, M. Bileschi, S. Tulyakov, J. Dobler, A. Rudra, and V. Govindaraju. Security analysis for fingerprint fuzzy vaults. In *SPIE Defense, Security and Sensing*, 2013. 1, 2, 3, 4

[14] A. K. Jain and D. Maltoni. *Handbook of Fingerprint Recognition*. Springer-Verlag New York, Inc., 2003. 1

[15] T.-Y. Jea and V. Govindaraju. A minutia-based partial fingerprint recognition system. *The Journal of Pattern Recognition*, 38:1672–1684, 2005. 1

[16] A. Juels and M. Sudan. A fuzzy vault scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006. 1, 3

[17] A. Kiayias and M. Yung. Cryptographic hardness based on the decoding of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 54(6):2752–2769, 2008. 2, 3, 4

[18] G. Kumar, S. Tulyakov, and V. Govindaraju. Combination of symmetric hash functions for secure fingerprint matching. In *20th International Conference on Pattern Recognition (ICPR)*, pages 890 –893, aug. 2010. 1

[19] C. Lee, J.-Y. Choi, K.-A. Toh, and S. Lee. Alignment-free cancelable fingerprint templates based on local minutiae information. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 37(4):980–992, 2007. 2

[20] S. Lee, D. Moon, S. Jung, and Y. Chung. Protecting secret keys with fuzzy fingerprint vault based on a 3d geometric hash table. In *Adaptive and Natural Computing Algorithms*, volume 4432 of *LNCS*, pages 432–439. 2007. 1, 2

[21] A. Lumini, D. Maio, and D. Maltoni. Continuous versus exclusive classification for fingerprint retrieval. *Pattern Recognition Letters*, 18(10):1027–1034, 1997. 2

[22] P. Mansukhani, S. Tulyakov, and V. Govindaraju. A framework for efficient fingerprint identification using a minutiae tree. *Systems Journal, IEEE*, 4(2):126–137, 2010. 1, 2

[23] J. Merkle, M. Niesing, M. Schwaiger, H. Ihmor, and U. Korte. Performance of the fuzzy vault for multiple fingerprints. In *BIOSIG*, pages 57–72, 2010. 1, 2

[24] A. Mhatre, S. Chikkerur, and V. Govindaraju. Indexing biometric databases using pyramid technique. In *5th Int. Conference on Audio and Video-based Biometric Person Authentication*, volume 3546 of *LNCS*, pages 841–849, 2005. 2

[25] K. Nandakumar, A. K. Jain, and S. Pankanti. Fingerprint-based fuzzy vault: Implementation and performance. *IEEE Transactions on Information Forensics and Security*, 2(4):744–757, 2007. 1, 2

[26] N. Ratha, K. Karu, S. Chen, and A. Jain. A real-time matching system for large fingerprint databases. *IEEE Pattern Analysis and Machine Intelligence*, August 1996. 2

[27] W. J. Scheirer and T. E. Boult. Cracking fuzzy vaults and biometric encryption. In *Proceedings of Biometrics Symposium*, pages 1–6, 2007. 3

[28] X. Tan, B. Bhanu, and Y. Lin. Fingerprint identification: classification vs. indexing. In *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.*, pages 151–156, 2003. 1, 2