ERROR TOLERANCE TECHNIQUES FOR BINDING CRYPTOGRAPHIC KEY WITH BIOMETRICS

Qinghai Gao
GaoQJ@farmingdale.edu

Department of Security Systems, Farmingdale State College, SUNY
2350 Broadhollow Road, Farmingdale, NY 11735

Abstract: Modern cryptography has one issue to be solved: key management. One proposed solution to the issue is to binding biometrics with cryptographic key. However, the non-reproducible measurements of biometrics make it difficult to bind a key with biometrics due to the exactitude requirement of cryptographic key. To bind a cryptographic key with a biometric, error tolerance technique has to be applied to process the biometric information. In this paper, we briefly survey the error tolerance techniques that have been proposed to minimize the fuzziness of biometric measurements. Since it is typical that different biometrics have to be measured with different instruments, different methods may have to be chosen for best measurement results. Advances in the topic are reported with a few representative biometrics, including keystroke, voice, signature, face, iris, and fingerprint. Since for all biometric applications the central issue to be solved is the fuzzy matching problem. We reported our preliminary testing results in this aspect.

Keywords: Error, Tolerance, Key, Biometrics, Minutiae, Template

1. Introduction

Biometrics is defined as the identification of an individual based on physiological and behavioral characteristics. The common physiological features include face, fingerprint, hand geometry, palm print, hand IR thermogram, iris and retina, ear, skin, odor, denture, and DNA. The common behavioral features include speech, gait, keystroke and signature.

In cryptography, key generation and management is a very important issue [1]. According to Schneier, "*Key management is the hardest part of cryptography and often the Achilles' heel of an otherwise secure system*." The typical practice is that a key would be generated mathematically and then assigned arbitrarily to a user. This approach has two problems. First is the repudiation problem due to the lack of direct physical connection between the key and its owner. Second is that the key has to be saved somewhere because it is too long to be memorized. Easy-to-remembered and easy-to-hacked passcode is then utilized to access the saved key. These problems potentially can be solved by binding a cryptographic key with biometrics, either by generating cryptographic key directly from live biometric measurements or by controlling the access of cryptographic key with biometrics.

A number of biometrics, such as keystroke patterns, voice, handwritten signatures, fingerprints, Iris, DNA and face images, have been studied for cryptographic key binding.

The major problem for binding cryptographic key with biometrics is that biometric measurements are non-exactly reproducible: typically two measurements of a same biometrics will give two similar but different results, which violate the exactitude requirement of cryptographic key. To solve the problem, error tolerance techniques have to be applied. In this paper we survey these techniques proposed in literatures for different biometrics. The rest of the paper is organized as follows. Section 2 briefly summarizes these methods; Section 3 gives

detailed description on how these methods are utilized based on particular biometrics; Section 4 contains our experimental results on matching modified fingerprint minutiae templates; Section 5 concludes and proposes future research.

2. Overview of the error tolerance methods

Biometric system works with two steps: registration and verification. For registration a person provide a live biometric for measurements and the results will be stored. For verification, the person must provide the same biometric for new measurements. The output of the new measurements will be compared to the previously stored results.

Biometric measurements generate noisy data and it is a challenging problem to achieve security with noisy data [2]. Among other things non-reproducibility is the most difficult problem to be solved for biometric application, including cryptographic key binding. In this section the common methods we found in literature are briefly described as follows.

●Averaging/Training

For averaging method a number of biometric samples with some variations were obtained, transformed, and then averaged to get a generic representation of the biometric. The mean and standard deviation of the samples are often needed.

For training method the biometric samples collected during registration will be applied to train a mathematical model, such as Hidden Markov Model (HMM). The parameters obtained at the end of the training will be used for verification.

●Quantization (aka, Tessellation/Discretization)

Individual biometric image will be quantized into a number of small units. The biometric information inside each unit will be assumed to locate at the center of the unit.

●Majority voting

For a number of measurements of a biometric, each will be quantized and binarized into a fixed-length string. For every bit position of the binary strings, majority voting will then be used to determine the value, 0 or 1.

●Error correction Coding and helper data

Error correction coding is often used for noisy data. Two common choices are Reed-Solomon (RS) coding and Hadamard coding, especially for iris encoding. During registration, some redundant information (also called helper data) about the biometric will also be collected and stored to correct the error bits at verification.

●Subsetting

Assume a biometric can be represented as a set of points N. Instead of using N registered points for verification, we only use a subset M of N points in the hope that at least M points of a biometric can be regenerated from new measurements.

To successfully bind a cryptographic key with a biometric, one or a combination of the aforementioned five methods will be applied for error tolerance purpose.

3. Biometrics-exemplified error tolerance techniques

A few biometrics, including keystroke dynamics, voice, handwritten signatures, face, iris, and fingerprint, have been proposed for cryptographic key binding. For different biometrics, different techniques have to be chosen to solve the fuzzy measurement problems.

3.1 Time averaging for keystroke dynamics measurements

Computer user's typing patterns consist of durations for each letter typed and latencies between keytrokes. Monrose, Reiter, Li and Wetzel [3] proposed to harden a user's password with keystroke dynamics.

Let $\phi_1, \phi_2, \ldots, \phi_m$ denote the number of features that are measured during logins. For each feature ϕ_i, let $T_i \in \Re$ be a fixed parameter of the system. Let μ_i and σ_i be the mean and standard deviation of the measurements $\phi_i(j_1),\ldots, \phi_i(j_H)$ where $j_1,\ldots, j_H$ are the last H successful logins and $H \in \mathbf{N}$ is a fixed parameter of the system.

ϕ_i is a distinguishing feature for the last H successful logins if $|\mu_i - T_i| > K \bullet \sigma_i$, where $K \in \Re$ is another system parameter.

Let $b(\phi_i)$ be the bit representation of feature ϕ_i. Then

$b(\phi_i)=0$, if $T_i > \mu_i + K \bullet \sigma_i$ means the user consistently measures below T_i on feature ϕ_i (fast);

$b(\phi_i)=1$, if $T_i < \mu_i - K \bullet \sigma_i$ means the user consistently measures above T_i on feature ϕ_i (slow)

$b(\phi_i)=\varnothing$, Otherwise

An m-bit string is derived and then combined with password to form a hardened password. The system parameter K plays an important role for the error tolerance. A large K means high error tolerance and high false acceptance.

3.2 Sequence segmentation for voice

Monrose, Reiter and Wetzel [4] [5] proposed another system based on voice biometrics.

A user's utterance is represented as a sequence of 30ms frames. A feature descriptor is constructed by segmenting the sequence of frames into contiguous subsequences, starting from a segmentation of the frame sequence into m, roughly equal-length segments. The segmentation is an iterative process, converging to a near-optimal segmentation of the user's utterance. One feature descriptor bit will then be generated from each subsequence. The relative position of the point obtained by averaging the frames in a subsequence to the closest matching centroid determines whether the bit is 1 or 0.

A larger number of centroids and shorter subsequences will increase the differentiating ability of the system (high false rejection rates) but tolerate fewer errors for a particular user.

3.3 Feature quantization for handwritten signatures

Hao and Chan [6] made use of handwritten signatures to generate private key. They defined forty-three signature features extracted from dynamic information like pen-down time, velocities for different directions, pressure, height-to-width ratio, etc. The measurements for each feature will be quantized into a number of segments, each of which will be assigned to a different decimal number *D*. During enrollment, ten samples will be collected from each user and the mean and standard deviation are used to calculate the user boundary:

$$\text{User boundary} = (\mu - b \bullet \sigma, \mu + b \bullet \sigma)$$

The "b" is a system parameter to be adjusted. A bigger "b" value corresponds to more error tolerance and easier forgeries.

The bit information for each feature can be obtained as $\log_2^D$. Bit representations for all the features of a user are concatenated to form a binary string *S*.

3.4 Basis projection for face images

Goh and Ngo [7] outline cryptographic key-computation from two-dimensional frontal speaking face bit-map images. Each image was then represented as a vector and Principle component analysis (PCA) was utilized to reduce the dimensions (less than 100 eigenbasis).

A face is represented with an image vector IV = (s_1, s_2, … , s_i, …, s_n), where s_i is the value for a pixel.

Binarize each point as the following (mean μ and standard deviation σ obtained at registration):

$$b(s_i) = 0 \text{ if } s_i < \mu-\sigma$$
$$=1 \text{ if } s_i > \mu+\sigma$$
$$=\varnothing \text{ if } s_i \in [\mu-\sigma\ \mu+\sigma]$$

In their experiments, 20 to 80 bits were extracted from each image. The results showed that 40 to 60 eigenfaces gives best results: FAR=0% and FRR<3%. 20 bits is insufficient and 80 bits is sensitive to noise.

3.5 Hadamard and Reed-Solomon coding for Iris

Daugman [8] proposed a practical and secure way to integrate iris biometrics into cryptographic applications. A random binary string, i.e. the key for encryption, is XORed and therefore locked with genuine iris codes from enrollment templates. To extract the key, the live iris has to be measured again to get the iris code and then XORed with the result obtained from the previous step. To deal with the 10 to 20% of error bits within an iris code and derive an error-free key, they carefully studied the error patterns within iris codes, and devised a two-layer error correction technique that combines Hadamard and Reed-Solomon codes.

3.6 Image averaging, shifting and majority coding for fingerprint

Soutar et al. [9-11]proposed a method to link and retrieve a digital key by using fingerprint.

At registration, a series of Fourier-Transformed fingerprint images are averaged and then multiplied with a random phase-only array. The phase-phase product of the result, termed Stored Filter Function $H_{stored}(u)$, is saved as the First part of the Bioscrypt$^{TM.}$. The inverse FT of the result, a combined image, will be used to link with an N-bit Random Number k_0. The linking algorithm will generate a Look-up Table, which is the Second part of the Bioscrypt^{TM}. The first S bits of $H_{stored}(u)$, will be encrypted with key k_0; the encryption result will then be hashed to create an identification code id_0, which is the Third part of the Bioscrypt^{TM}.

At verification, A series of fingerprint images are captured, Fourier-Transformed and then combined to generate the magnitude information, which is combined with the saved $H_{stored}(u)$ and then inversely Fourier-Transformed to generate a new image, which will be used with the stored Look-up table to retrieve the key k_1.

A few procedures were proposed for error tolerance:

- Pre-align finger for image capturing;
- Use more than one images. As the number of fingerprint images increases, the average of the FT's of the images converges;
- Only the central portion of the images is used for key linking and retrieval;
- During the process of key retrieval, the selected portion will be shifted horizontally, vertically and diagonally by a number of pixels (1 to 16);
- Majority coding.

To check the validity of the newly generated key k_1, using it as an encryption key, encrypt the same S bits of the stored filter function $\mathbf{H_{stored}(u)}$, then hash the encrypted text to produce $\mathbf{id_1}$. If $\mathbf{id_1 = id_0}$ **,** then $\mathbf{k_1 = k_0}$; if $\mathbf{id_1 \; != id_0}$ **,** then $\mathbf{k_1 != k_0}$**,** the retrieval algorithm continues with the next pixel offset.

In Table 1, we listed some implementations of biometric key binding. However, even for a same biometric it is difficult to compare the results from one report with the results from another since they often use different methods and different databases. Standardization can help to solve the problem.

Table 1 Some Representative Implementations of Biometric Key Binding

In Table 1 the Bit Lengths vary from 12 to 224 and the FRRs are high.
Since the widely used symmetric-key encryption method AES uses 128 bits and public-key encryption method RSA uses ~1024 bits, current research on binding cryptographic key with biometrics has yet to reliably generate bit strings that can be used practically.

3.8 Fuzzy vault scheme

Juels and Sudan [12] proposed a fuzzy vault scheme. First, Alice selects a polynomial *p* to encode *k* with the coefficients of *p*. She constructs a vault set *R* containing the polynomial projections with her own set *A* and some chaff points. Bob can reconstruct k using his own set *B* if $A \cap B$ is greater than the threshold determined by *p*. The problem with the scheme proposed in [12] is that even small variations for fingerprint minutia, which are common for biometric measurements, will render the failure of reconstructing *k*.
Uludag [13] et al. improved the scheme by introducing Cyclic Redundancy Check (CRC) bits in the polynomial and using square tessellation of 2D image. The example given in [13] for encoding and decoding a 128-bit cryptographic key follows.

Encoding

●Apply the 16-bit primitive polynomial $a^{16} + a^{15} + a^2 + 1$ to a 128-bit key to get 16-bit CRC and construct a 144-bit secret ***SC*** that would be divided into 9 non-overlapping 16-bit segments and each segment is declared as a specific coefficient c_i, i=0, 1, 2, ..., 8.

●Concatenate 8-bit x and 8-bit y coordinates of minutiae: u=x|y. Hence, N minutia set A={u_1, $u_{2,}$.., u_N}and polynomial $p(u)=c_8u^8 + c_7u^7 + ... + c_1u + c_0$ are used to calculate genuine set G={(μ_1, $p(\mu_1)$), (μ_2, $p(\mu_2)$),..., (μ_N, $p(\mu_N)$)} which is then combined with chaff set C={(c_1, d_1), (c_2, d_2), ..., (c_M, d_M)}, where $d_j \neq p(c_j)$, j=1, 2, ..., M, to get the Vault Set ***V***=G∪C.

Decoding

●Given N query minutiae, u_1^*, u_2^*, ..., u_N^*. The points to be used in polynomial reconstruction are found by comparing u_i^* , i=1,2,..., N, with the *x* values of points in vault *V*. If there is a match, the corresponding vault point is added to a candidate list.

●Assume that the list has K points, K≤N. To decode a D-degree polynomial, (D+1) unique points are needed. So we end up with C(K, D+1) combinations. For each of these combinations, we construct the Lagrange interpolating polynomial. The coefficients are mapped back to the secret ***SC****, which will be divided by the primitive polynomial to test the validity of the combination: it is valid if the remainder is zero.

4. Experimental results on matching modified fingerprint minutiae templates

To test the capacity of error tolerance of biometric matching, we choose to use the fingerprints from DB1 (Optical Sensor, 640x480, 500 dpi) of the FVC2004 [14] and the NIST fingerprint software [15] since both of them are freely available and well-known.

The main procedures are followed:

- Extract fingerprint minutiae templates from fingerprint image. Each minutia is a 4D vector.
- Randomly modify the coordinates of the minutiae in a template
- Randomly delete certain numbers of minutiae from a template
- Match the original template against the modified template

Our testing results are given in Figure 1.

Figure 1 Matching modified fingerprint minutiae templates

The template modification details are given in Table 2.

Table 2 Fingerprint minutiae template modification steps

Since the threshold for matching decision is 40 [15], the modified fingerprint template with 36 minutiae, whose coordinates have been randomly altered by 10 can still match it original template that has 106 minutiae.

5. Conclusion

Binding biometrics with cryptographic key could improve the security of modern cryptography. Since biometric templates have unreliable bits, error tolerance techniques must be adopted to meet the exactitude requirement of cryptographic keys. Different techniques should be considered for different biometrics. Possible methods include Averaging/Training, Quantization, Majority Voting, Error Correction Coding and Subsetting. Today it is difficult to compare the results from different authors due to the lack of standard.

The central issue of biometric application is the fuzzy matching problem. Our testing results show that NIST fingerprint software has good error tolerance capacity.

Research in binding cryptographic key with biometrics is still in its early stage. More advanced error tolerance techniques may have to be developed to prove the usefulness of biometrics on enhancing cryptographic key management.

References

[1] B. Schneier, *Applied Cryptography*, 2nd Edition, John-Wiley, New York, 1996.

[2] P. Tuyls, B. Skoric, and T. Kevenaar. *Security with noisy data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*, 1st Edition, Springer, 2007.

[3] F. Monrose, M. Reiter, and S. Wetzel, *Password Hardening Based on Keystroke Dynamics. Proc. of the ACM Conference on Computer and Communications Security*, 1999, pp. 73-82.

[4] F. Monrose, M.K. Reiter, Q. Li and S. Wetzel. *Cryptographic key generation from voice. Proc. of the 2001 IEEE Symposium on Security and Privacy*, May 2001.

[5] F. Monrose, M. K. Reiter, Q. Li, D. P. Lopresti, and C. Shih. *Toward speech-generated cryptographic keys on resource constrained devices. Pro.c of 11th USENIX Security Symposium*, 2002, pp. 283-296.

[6] F. Hao and C. Chan. *Private key generation from on-line handwritten signatures*. *Information Management & Computer Security*, 10(2), 2003, pp. 159-164.

[7]A. Goh and D. Ngo. *Computation of cryptographic keys from face biometrics*. *Lecture Notes in Computer Science,* 2828, 2003, pp. 1-13.
[8] F. Hao, R. Anderson, and J. Daugman, *Combining cryptography with biometrics effectively*. *IEEE Transactions on Computers*, 55(9), 2006, pp. 1081-1088.
[9] R. Nichols*, ICSA Guide to Cryptography*, McGraw-Hill, 1998.
[10] C. Soutar, D. Roberge, A. Stoianov, A. Gilroy, and B. Kumar. *Biometric Encryption™ - Enrollment and Verification Procedures. Proc. of SPIE,* 3386, 1998, pp. 24-35.
[11] C. Soutar, D. Roberge, A. Stoianov, A. Gilroy, and B. Kumar. *Biometric Encryption™ using image processing. Proc. of SPIE,* 3314, 1998, pp.178-188.
[12] A. Juels and M. Sudan. *A Fuzzy Vault Scheme. Designs, Codes and Cryptography*, 38(2), 2006, pp. 237-257.
[13] U. Uludag, S. Pankanti, and A. Jain. *Fuzzy Vault for Fingerprints*. *Proc. of Audio and Video-based Biometric Person Authentication*, 2005, pp. 310-319.
[14] Available online: http://bias.csr.unibo.it/fvc2004/.
[15] NIST Biometric Image Software (NBIS). Available online: http://fingerprint.nist.gov/

Table 1 Some Representative Implementations of Biometric Key Binding

Biometrics	Feature	Error Correcting	FRR %	FAR %	Database	Bit Length	Author
Keystroke	Duration, latency	Discretization	48	NA*	20 users	12	[3]
Voice	Cepstrum coefficient	Discretization	20	NA	10 users	46	[4][5]
Signature	Dynamic	Averaging	28	1.2	25 users	40	[6]
Face	Eigen-projections	Discretization	0~5.0	0	153 users	20~80	[7]
Iris	Gabor wavelet	RS coding Hadamard	0~12	0	70 users	42~224	[8]
Fingerprint	Fourier Transform	Majority coding	NA	NA	NA	NA	[9-11]

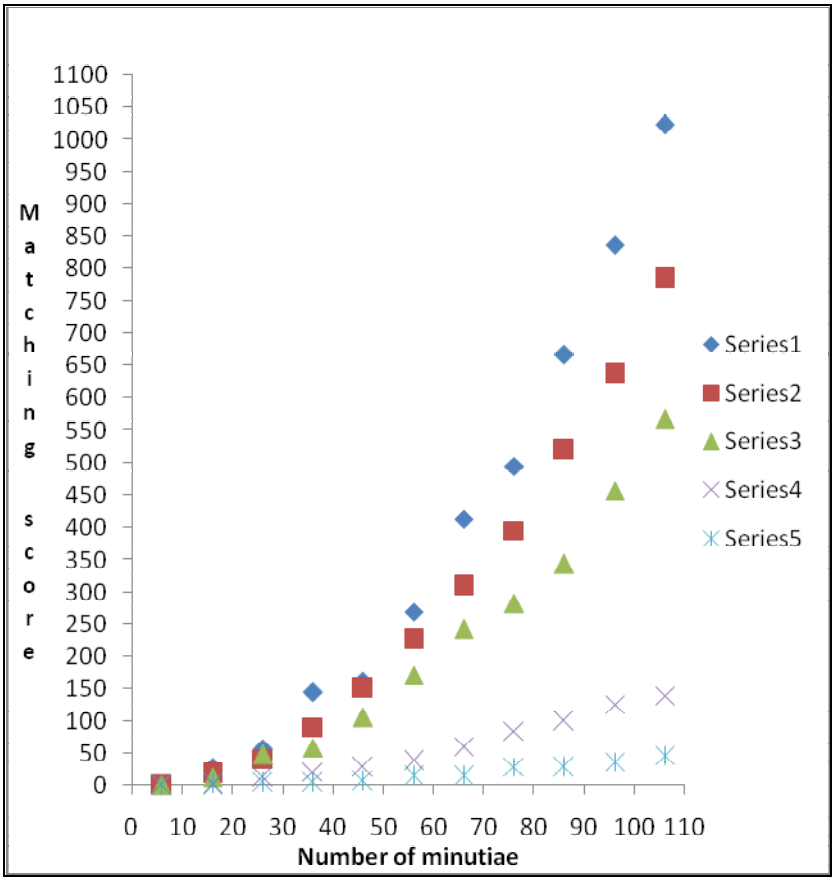

Figure 1 Matching modified fingerprint minutiae templates

Table 2 Fingerprint minutiae template modification steps

Symbol	Modification
◆	ModA(Series 1): Randomly delete 10 minutiae at each step
■	ModB(Series 2): Randomly change the 4 coordinates by 0~3, then ModA
▲	ModC(Series 3): Randomly change the 4 coordinates by 0~5, then ModA
×	ModD(Series 4): Randomly change the 4 coordinates by 0~10, then ModA
✳	ModE(Series 5): Randomly change the 4 coordinates by 0~15, then ModA