SECURE BIOMETRIC SYSTEMS

By

Umut Uludag

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science & Engineering

2006

Abstract

SECURE BIOMETRIC SYSTEMS

By

Umut Uludag

Traditional personal authentication systems that are based on knowledge (e.g., password) or physical tokens (e.g., ID card) are not able to meet strict security performance requirements of a number of modern applications. These applications generally make use of computer networks (e.g., Internet), affect a large portion of population, and control financially valuable and privacy-related tasks (e.g., e-commerce). Biometrics-based authentication systems that use physiological and/or behavioral traits (e.g., fingerprint, face, and signature) are good alternatives to traditional methods. These systems are more reliable (biometric data can not be lost, forgotten, or guessed) and more user-friendly (there is nothing to remember or carry). In spite of these advantages of biometric systems over traditional systems, there are many unresolved issues associated with the former. For example, how secure are biometric systems against attacks? How can we guarantee the integrity of biometric templates? How can we use biometric components in traditional access control frameworks? How can we combine cryptography with biometrics to increase overall system security?

In this dissertation, we address these issues and develop techniques to eliminate associated problems. Firstly, we analyze attack robustness of fingerprint matchers, and develop algorithms for circumventing them. The proposed approach is shown to be very successful in bypassing the security associated with fingerprint systems. Further, we develop methods to counter this attack. Secondly, we develop algorithms for increasing the security of image-based (e.g., fingerprint and face) biometric templates, via embedding additional information in them. We show that these algorithms do not reduce biometric matching performance. Thirdly, we develop a secure multimedia content distribution framework that includes fingerprint matching. This provides another line of defense against the piracy of copyrighted data. Finally, we develop a hybrid system that combines traditional cryptography with fingerprint biometrics. The security associated with cryptographic algorithms and the user-friendliness of biometrics coexist in such systems.

To my family

Acknowledgments

I would like to thank my advisor Dr. Anil Jain for all his guidance and support throughout the course of my Ph.D. program. I have been privileged to work under his supervision, and I truly appreciate his help. I am grateful to other members of my Ph.D. committee, Dr. George Stockman, Dr. Abdol-Hossein Esfahanian, and Dr. Sarat Dass for all their guidance.

I am grateful to Dr. Sharath Pankanti and Dr. Salil Prabhakar for their excellent comments on my research. Many thanks to the members of the PRIP Lab at MSU for providing such a productive working atmosphere: Dr. Anoop Namboodiri, Dr. Arun Ross, Dirk Colbry, Hong Chen, Karthik Nandakumar, Martin Law, Meltem Demirkus, Dr. Michael Farmer, Miguel Figueroa-Villanueva, Mrityunjay Kumar, Dr. Paul Albee, Pavan Mallapragada, Shailesh Saini, Silviu Minut, Steve Krawczyk, Unsang Park, Dr. Vincent Hsu, Xiaoguang Lu, Yi Chen, and Yongfang Zhu.

I appreciate the administrative help and support of Adam Pitcher, Cathy Davison, Debbie Kruch, Kim Thompson, Linda Moore, Norma Teague, and Starr Portice. Special thanks to Dr. Miroslav Trajkovic, Dr. James Reisman, Michael Indovina, and Dr. Rainer Lienhart for their guidance during my summers at Symbol Technologies, Siemens Corporate Research, National Institute of Standards and Technology, and Intel Corporation, respectively. Many thanks to Goksel Dedeoglu & Susan Rossbach and Gokhan Gokoz & Aysecan Boduroglu Gokoz for their friendship and support.

Finally, I would like to thank my parents, sister and Yesim for their love and encouragement. I dedicate this thesis to them.

Table of Contents

List of Tables

List of Figures

Images in this dissertation are presented in color.

Chapter 1

Introduction

1.1 Biometrics and Security

With the proliferation of large-scale computer networks (e.g., Internet), the increasing number of applications making use of such networks (e.g., e-commerce, e-learning), and the growing concern for identity theft problems, the design of appropriate personal authentication systems is becoming more and more important. Systems that have the ability to authenticate persons (i) accurately, (ii) rapidly, (iii) reliably, (iv) without invading privacy rights, (v) cost effectively, (vi) in a user-friendly manner, and (vii) without drastic changes to the existing infrastructures are desired. Note that some of these requirements conflict with the others. The traditional personal authentication systems that make use of either a (secret) piece of knowledge (e.g., password) and/or a physical token (e.g., ID card) that are assumed to be utilized only by the legitimate users of the system are not able to meet all of these requirements.

Biometrics-based personal authentication systems that use physiological and/or behavioral traits (e.g., fingerprint, face, iris, hand geometry, signature, voice ...) (see Fig. 1.1) of individuals have been shown to be promising candidates for either replacing or augmenting these traditional systems [22, 34]. They are based on entities (traits) that are actually bound with the individual at a much deeper level than, for example, passwords and ID cards. As a result, they are more reliable since biometric information can not be lost, forgotten, or guessed easily. They lead to increased user convenience: there is nothing to remember or carry. They improve the authentication accuracy: the system parameters can be tuned so that the probability of illicit use of the system can be reduced. Further, the cost of incorporating biometric components into an authentication system is continually decreasing, whereas the cost of relying on traditional authentication mechanisms is increasing. It has been reported that around 25% of all help desk calls are related to password resets, with a cost of around $20 per reset [2].

As a comparison of traditional and biometrics-based systems, Figures 1.2, 1.3 and 1.4 show the block diagrams of these three types of authentication. Note that verification refers to the authentication procedure when the user claims an identity (I) and the output is a (Yes/No) decision. On the other hand, during identification the user does not claim an identity: the authentication system searches the entire database of enrolled users for a match, and if there is a match, it outputs the identity of the user I.

In the password-based scheme the user submits a password P, which is generally passed through a one-way hash function (e.g., MD5 [56]) resulting in P' (this assures that even users with super-user privileges can not access passwords). This, along

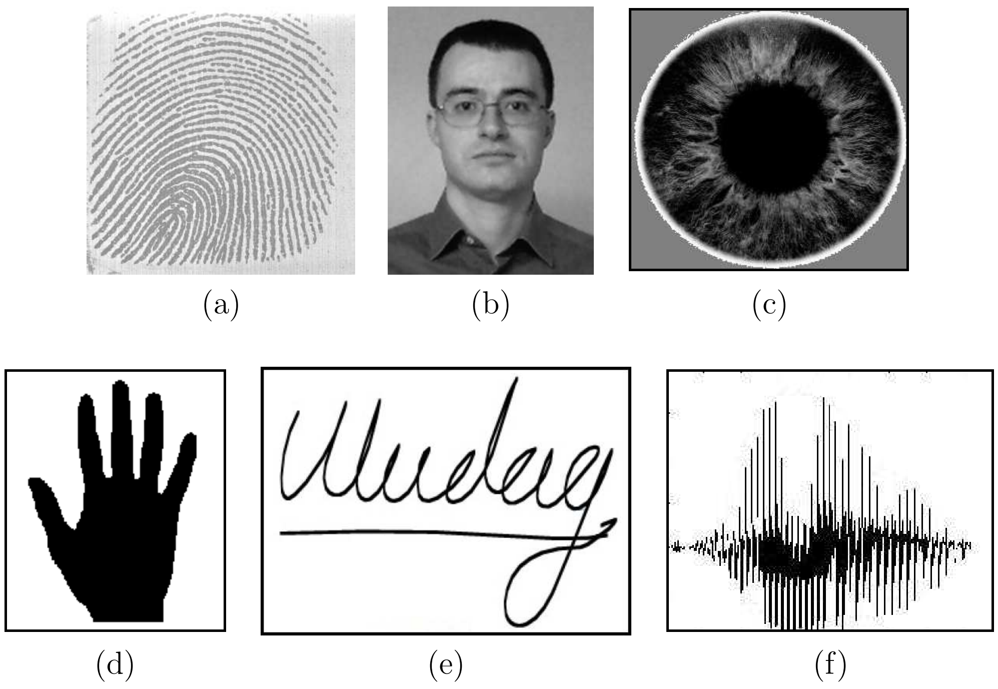


Figure 1.1: Sample biometric traits: (a) fingerprint, (b) face, (c) iris, (d) hand geometry, (e) signature, and (f) voice.

with the identity I is saved in a database during enrollment. During verification, the user submits the password P_v, whose hash P'_v is compared to P', the hash value retrieved from the database. A "Yes" decision is the output if and only if the two hashes are the same (which indicates that, with very high probability, $P_v = P$ due to the characteristics of hash functions).

In the ID card-based scheme, the user's identity I and credentials C (e.g., her birth certificate, diploma, signature) are checked by the authenticating institution (e.g., a university registrar's office), which stores this information in a database. It also generates the token $ID_{I,C,A}$ that can include the identity, credential (e.g., signature) and access privileges A (e.g., string "Graduate Assistant" printed on the card, an RFID (Radio Frequency Identification) tag with a code to open laboratory doors).

This token can also contain entities to bind the token only with the authenticating institution, such as an embossed seal, specific logo, or bar code to eliminate illegal reproduction of the token. In a supervised authentication application (where a human attendant is available), the user's card $ID_{I,C,A}$ and her credentials (e.g., signature) are checked for consistency by a (human) supervisor: if these data match, the supervisor accepts the user. In an unsupervised application, just the presence/validity of the card is checked (e.g., checking whether it has a specific RFID tag).

In the fingerprint-based scheme, during enrollment the user presents her finger F to the sensor, whose output F_s (e.g., fingerprint image) is passed through a feature extractor to arrive at the template F_t, which, along with the identity I of the user, is saved in a database (note that this database can be central, such as a law enforcement database or local, such as a smart-card issued to an individual). During verification, the user's fingerprint is captured again, and the generated template $F_{v,t}$ is matched against the database template F_t corresponding to the claimed identity I. If these two representations are "close enough", the matcher outputs a "Yes" decision. This decision is generally based on a similarity (dissimilarity) measure: if the similarity (dissimilarity) score between two representations is higher (lower) than a specific threshold T, a "Yes" decision is output, otherwise, a "No" decision is output. Conversely, during identification, the user's template generated online, $F_{i,t}$, is matched against *all* the database templates. If there is a match, the matcher outputs the associated identity I of the user.

Authentication accuracy and user convenience can be quantified using the metrics False Accept Rate (FAR) and False Reject Rate (FRR), respectively. FAR is the

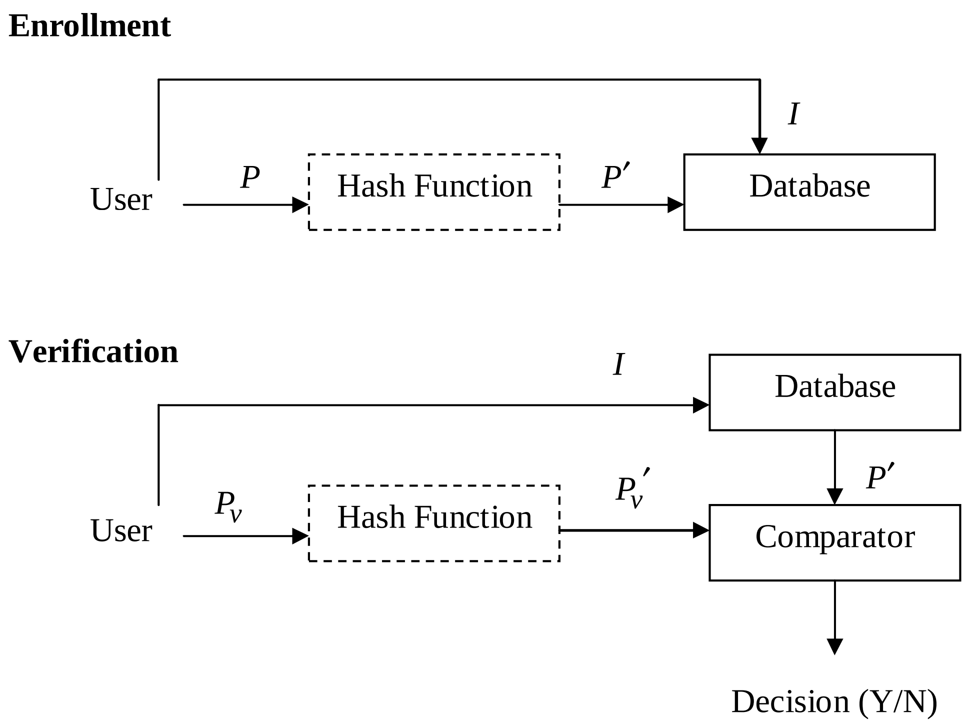


Figure 1.2: Knowledge-based (password) authentication.

probability that an imposter access attempt will be successful; FRR is the probability that a genuine access attempt will fail. Equal Error Rate (EER) denotes the point where FAR = FRR. Another commonly used metric is Genuine Accept Rate (GAR), which is the probability that a genuine access attempt will be successful. Hence, $GAR = 1 - FRR$. All these metrics are dependent upon the decision threshold T, so we can label them for a specific threshold T_1 as FAR_{T_1} and GAR_{T_1}. By varying the decision threshold T, e.g., $T = T_1, T_2, \ldots, T_K$, we can obtain multiple (K) *operating points* of the system. The resulting plot of GAR versus FAR is called the Receiver Operating Characteristics (ROC) curve, which is commonly used to evaluate the performance of biometric systems. Fig. 1.5 shows the ROC curve of a fingerprint verification system; we see a sample operating point as $(GAR, FAR) = (0.87, 0.001)$.

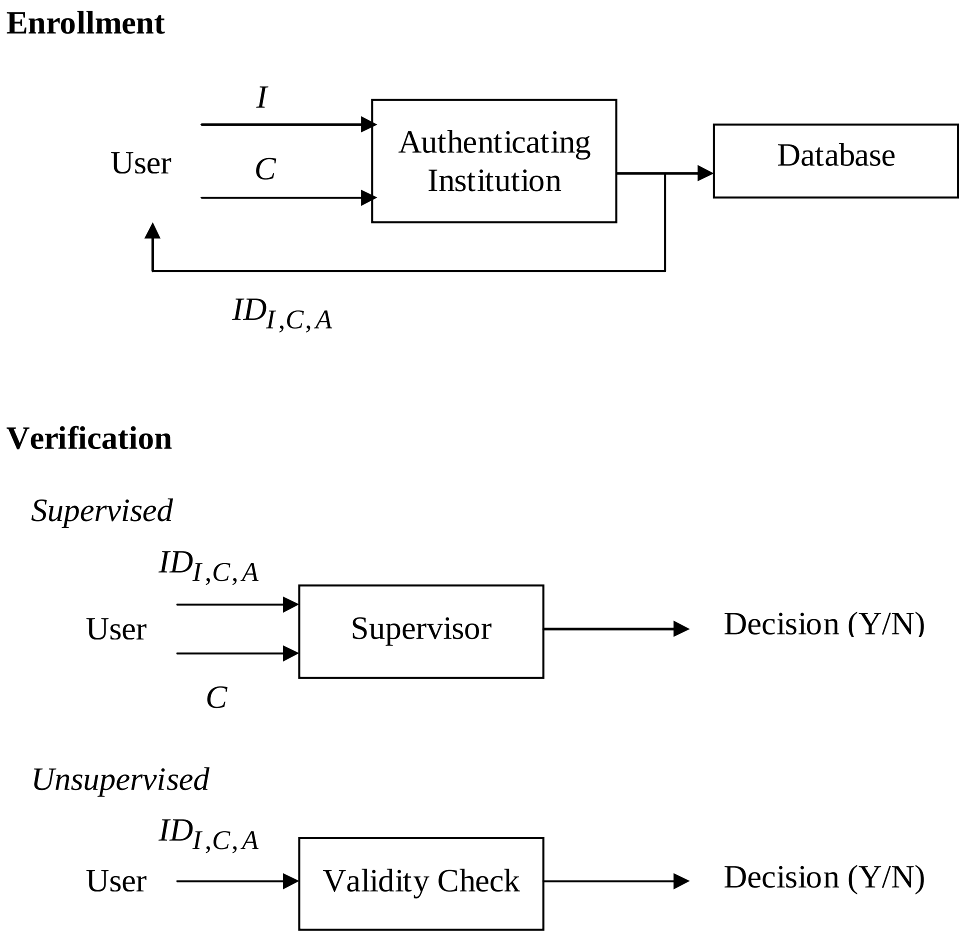


Figure 1.3: Token-based (ID card) authentication: I denotes user identity and C denotes the associated credential.

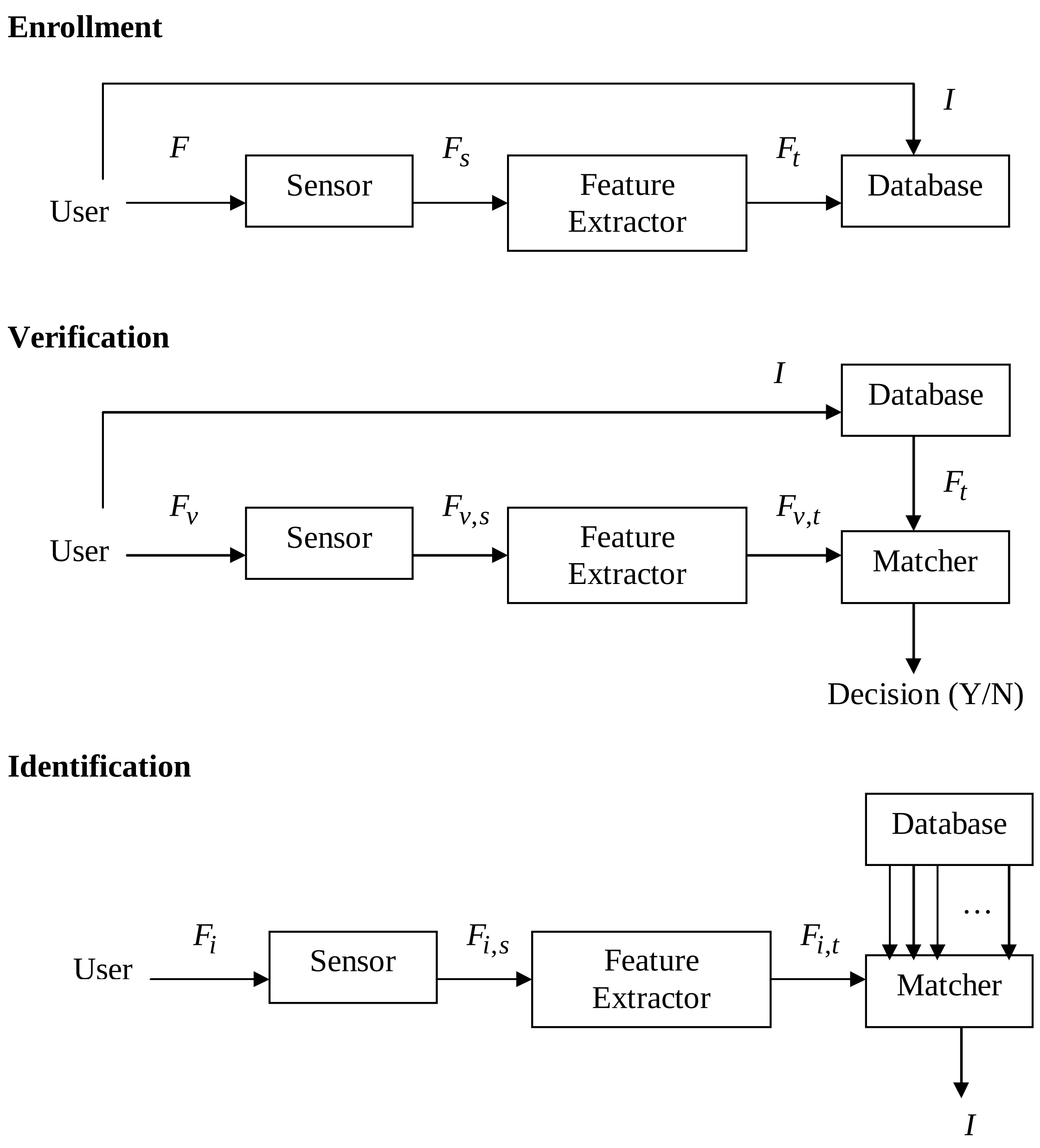


Figure 1.4: Biometrics-based (fingerprint) authentication.

Biometric system administrators can analyze this curve in light of the requirements for their specific application (e.g., "GAR should be greater than 0.85 to assure user-friendliness of the system", and "FAR should be less than 0.001 to assure security of the system") and decide on the operating point to use.

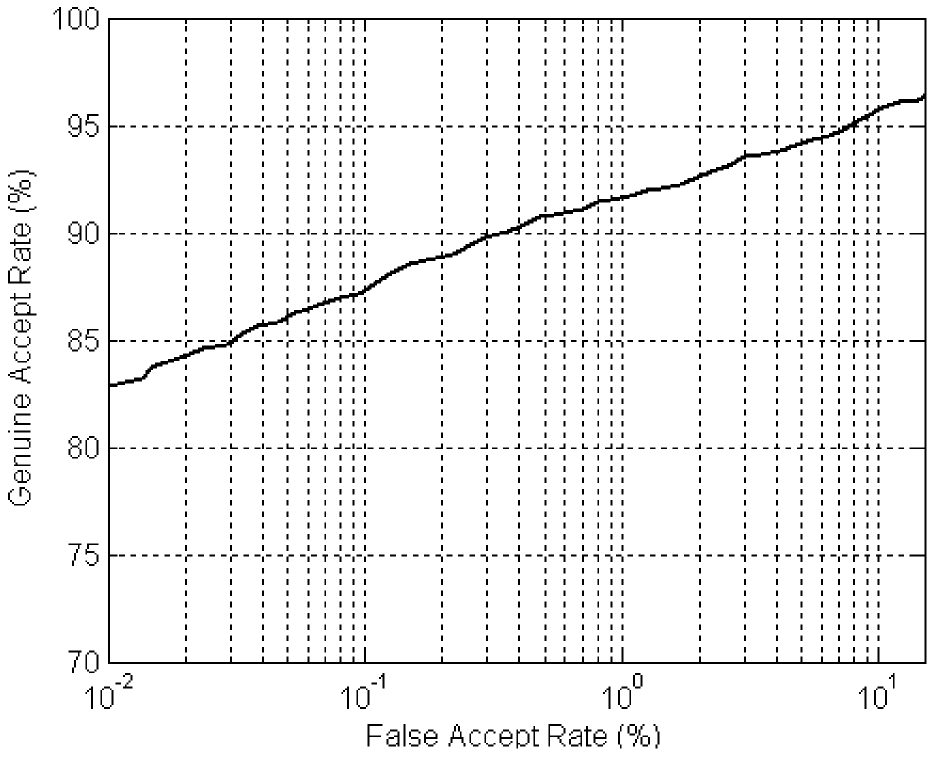


Figure 1.5: A typical ROC curve of a fingerprint verification system.

In spite of the advantages of biometrics-based authentication systems compared to traditional authentication schemes, there are still unresolved problems associated with the former. These problems generally emerge from the *security* characteristics of the biometrics-based systems. Here, the term security is used to denote the overall reliability of the system, rather than just the simplistic notion of increased authentication accuracy (decrease of FAR and FRR) brought about by the use of biometrics for verification/identification. While it is true that any increase in such authentication accuracy increases the security of the system, there are many other issues (cited below) that need to be taken into account before arriving at a truly secure biometric system.

One security-related issue is the robustness of the biometric systems against attacks devised specifically to thwart their operation. The security analysis of traditional password-based authentication schemes can be based on simple parameters, such as the minimum length of passwords (e.g., minimum of 8 characters), the password change frequency (e.g., at least twice a year), and the complexity of the passwords (e.g., it must include upper and lowercase letters, numbers and special characters such as #, &, *). Note that the number of possible passwords with 8 characters is theoretically very large (e.g., when characters are from 7-bit ASCII code, this number is $128^8 \approx 7.2 \cdot 10^{16}$). But most people use easily guessed passwords, thus, the space of useful passwords is relatively small.

Similarly, the security of token-based systems can be analyzed based on the probability of illegal utilization of the token (e.g., the probability that a valid ID card is lost and later found by an attacker), the feasibility of illegally replicating/generating a token (e.g., how easy it is to forge an ID card that includes a signature, a magnetic strip, and a face image embossed with the seal of the authenticating institution) and the feasibility of illegally mimicking the characteristics included in the token (e.g., how easy it is for an attacker to change her facial appearance and signature with the aim of bringing them closer to the data residing in the token).

Biometric systems, on the other hand, are intrinsically much more complicated than these traditional authentication schemes, due to (i) nature of the data they need to operate on (e.g., biometric data of individuals are not identical in each acquisition, compared to either true/false (present/non-present) for a password (ID card)), (ii) the number and complexity of the associated modules (e.g., complex image enhancement operations may be necessary for poor quality fingerprints) and (iii) the overall architectural design (e.g., the need to securely access stored biometric data). As a result, there are many critical points in a biometric system that can be compromised, which are inherently absent in traditional authentication schemes [52].

A specific problem within this general scope of system robustness against attacks is guaranteeing the security of biometric templates. As an example, consider a fingerprint authentication system. During enrollment, fingerprint images are captured from the individual and stored in a central (e.g., authentication server) or local (e.g., smart-card issued to the individual) database as her template. The verification module compares the image acquired at the time of the associated transaction (e.g., when she wants to purchase a book online) to her template. Verification succeeds if and only if the similarity between the two exceeds a pre-specified threshold. How can the attacks that aim to modify or delete existing templates, or introduce new templates into the database be eliminated?

The issue of seamless integration of biometric components into existing authentication systems is also important. Due to the rather independent development of traditional and biometrics-based authentication systems (and hence the differences in their applicability and the amount of associated knowledge-base), many of the current applications (e.g., access to computing facilities, financial records, secure areas, multimedia data (e.g., image, audio, video)) and frameworks (e.g., cryptography) are designed tightly around traditional systems. Trying to incorporate biometric components into such applications brings along problems (e.g., temporal variation in biometric data) that need to be resolved. For example, using biometric data in traditional encryption/decryption architectures for copyright protection of multimedia data is an unresolved problem.

Considering that both the traditional cryptographic systems and biometrics-based authentication systems aim at the common goal of reliable access control, there have been attempts to combine these seemingly disjoint domains. Basically, these approaches can be grouped into two classes: (i) the biometric component acts as a wrapper around the cryptographic component, and (ii) the biometric component is merged with the cryptographic domain at a much deeper level, where biometric matching essentially takes place within the realm of cryptography [68]. These biometric crypto (or crypto-biometric) systems, provided their elements are combined intelligently, have the potential to merge the proven security of cryptographic systems with the ease-of-use property of biometric systems.

These problems are pictorially represented in Fig. 1.6, which also depicts the domains of *biometrics* and *security* areas.

1.2 Thesis Contributions

In this dissertation, the problems cited in Section 1.1, namely, (i) robustness of biometric systems against attacks, (ii) protecting biometric templates, (iii) integrating

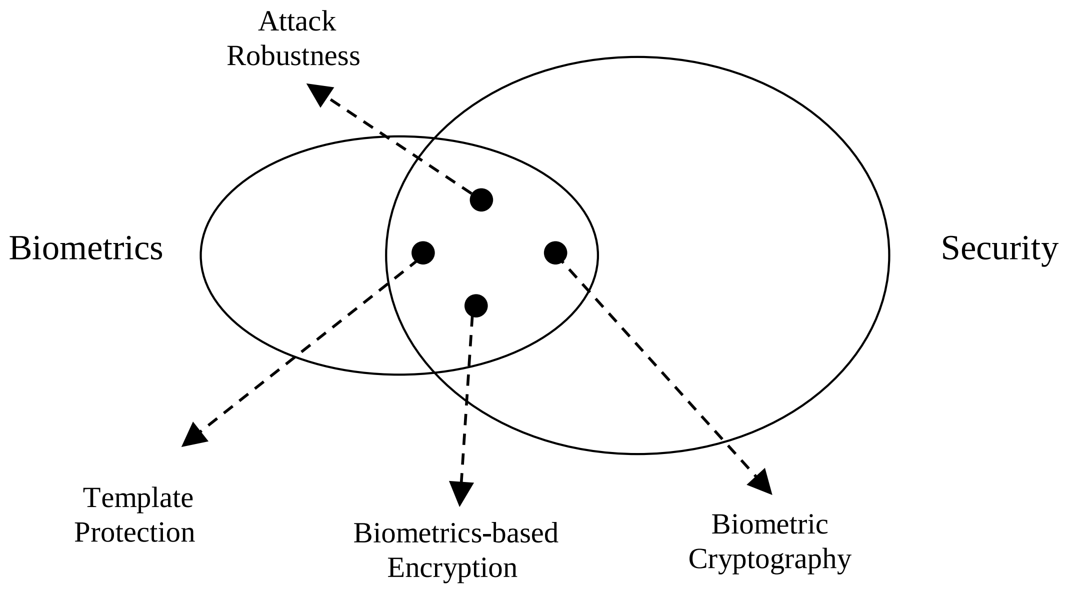


Figure 1.6: Problems at the intersection of biometrics and security.

biometric components into existing computing architectures, and (iv) combining cryptography with biometrics, will be analyzed and appropriate solutions for them will be proposed and developed. Specifically, the major contributions of this dissertation include:

1. Design and development of a hill-climbing based attack system for the fingerprint biometric, with the aim of showing the vulnerable points in fingerprint matching systems, and the development of a defense mechanism against this attack.

2. Design and development of watermarking techniques to increase the security of image-based biometric templates, such as fingerprints and faces.

3. Design and development of a multimedia content distribution system using biometric matching for increased security and user convenience.

4. Design and development of a practical biometric cryptosystem utilizing fingerprint features that combines the benefits of biometric systems with the security of cryptographic systems.

Chapter 2

Attacks Against Biometric Systems

2.1 Introduction

Biometrics-based personal authentication systems that use physiological (e.g., fingerprint, face, iris) or behavioral (e.g., speech, handwriting) traits are being increasingly utilized in many applications to enhance the security of physical and logical access systems. Even though biometric systems offer several advantages over traditional token (e.g., key) or knowledge (e.g., password) based authentication schemes (e.g., increased user convenience and robustness against imposter users), they are still vulnerable to attacks.

Ratha et al. [52] analyzed these attacks and grouped them into eight classes. Fig. 2.1 shows the locations of these attacks in a generic biometric system. A Type 1 attack involves presenting a fake biometric (e.g., finger made from silicon, face mask, lens including fake iris texture) to the sensor. The second type of attack is called a replay attack, because an intercepted biometric (with or without the cooperation of the genuine user) data is submitted to the feature extractor, bypassing the sensor. In the third type of attack, the feature extractor module is replaced with a Trojan horse program that functions according to its designer's specifications (henceforth, these users that try to break into systems protected by biometric authentication will be collectively called "Trudy"). In the fourth type of attack, genuine feature values are replaced with values (synthetic or real) selected by the attacker. In the fifth type of attack, the matcher is replaced with a Trojan horse program. The attacks on the template database (e.g., addition, modification, or removal of templates) constitute the sixth type of attack. In the seventh type of attack, the templates are tampered with (stolen, replaced, or altered) in the transmission medium between the template database and matcher. Lastly, the matcher result (accept or reject) can be overridden by the attacker.

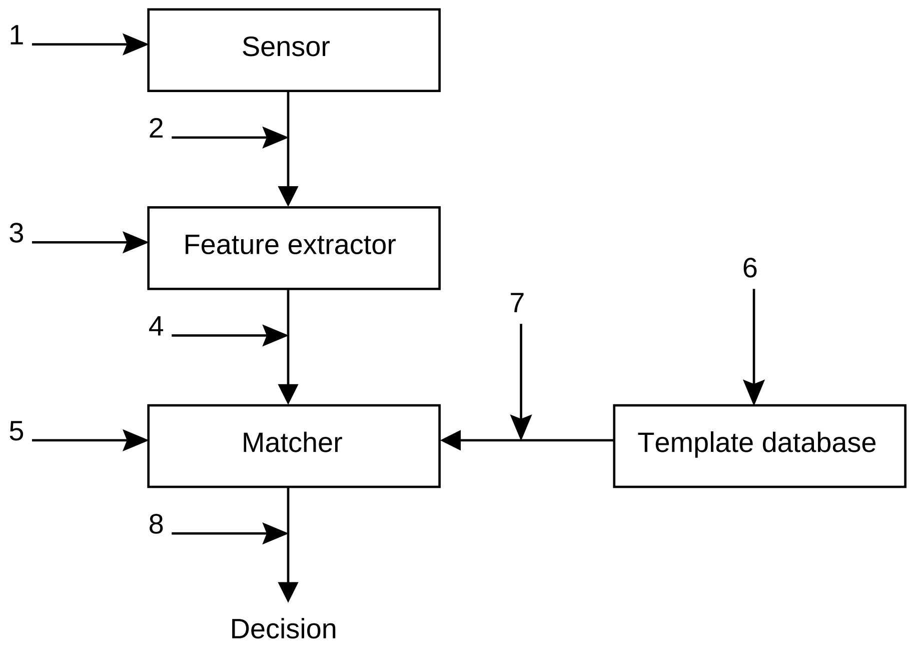


Figure 2.1: Locations of possible attacks in a biometric system.

Note that the relative threats of these attacks depend on (i) biometric modality (e.g., it is harder to physically replicate a retina scan than it is to forge a signature), (ii) type of sensor (2D vs. 3D face sensors), (iii) type of matcher operating on the same biometric (face matchers based on texture vs. geometry), and (iv) the security settings (reflected via False Accept Rate) of the biometric system.

Schneier [57] compared traditional security systems with biometric-based ones. The lack of secrecy (e.g., leaving fingerprint impressions on the surfaces we touch, face images being captured by hidden cameras), and non-replaceability (e.g., once the biometric data is compromised, there is no way to return to a secure situation, unlike replacing a key or password) are identified as the main problems of biometric systems.

Maltoni et al. [34] described typical threats for a generic authentication application. In *Denial of Service* (DoS), an attacker corrupts the authentication system so that legitimate users cannot use it. An online biometric authentication server that processes access requests (via retrieving templates from a database and performing matching) can be bombarded with a multitude of bogus access requests, to a point where its computational resources cannot handle valid requests any more. In *circumvention*, an attacker gains access to parts of the system protected by the authentication application. This threat can be cast as a privacy attack, where the attacker accesses some data that she was not authorized to access (e.g., medical records of another user), or as a subversive attack, where the attacker manipulates the data (e.g., changing records, submitting bogus insurance claims, etc.). In *repudiation*, the attacker denies accessing the system. For example, a corrupt bank clerk who modifies some financial records illegally may claim that her biometric data was “stolen”, or she can argue that the False Accept Rate (FAR) of the biometric system allowed a different user to access her account. In *contamination* (covert acquisition), an attacker can surreptitiously obtain biometric data of legitimate users (e.g., lifting a latent fingerprint and constructing a three-dimensional mold) and use it to access the system. Further, the biometric data associated with a specific application can be used in another unintended application (e.g., using a fingerprint for accessing medical records instead of the intended use of office door access control). This becomes especially important for biometric systems since we have a limited number of useful biometric traits, compared to a practically unlimited number of traditional access identities (e.g., keys and passwords). Cross-application usage of biometric data becomes more probable with the growing number of applications using biometrics (e.g., opening car or office doors, accessing bank accounts or medical records, locking computer screens, gaining travel authorization, etc.). In *collusion*, a legitimate user with super user privileges (e.g., system administrator) is the attacker who illegally modifies the system. In *coercion*, attackers force the legitimate users to access the system (e.g., using a fingerprint to access ATM accounts at gunpoint).

The problems that may arise from the above-mentioned attacks on biometric systems are raising concerns as more and more biometric systems are being deployed [8]. This, combined with the increase in the size of the population using these systems may lead to significant financial, privacy, and security related breaches.

2.2 Background

This section summarizes several studies that show the vulnerability of biometric systems and provide solutions to some of the attacks presented in Section 2.1.

The success of fake biometric submission to sensors (type 1 attack) has been demonstrated by several researchers. Note that this attack does not need anything more than a fake biometric; hence, the feasibility of this type of attack compared to others is rather high. For example, no knowledge of the matcher, template specification, or template database access privileges (generally limited to system administrators) is necessary. Also, since this attack operates in the analog domain, outside the digital limits of the biometric system, digital protection mechanisms such as encryption, digital signature, and hashing are not applicable.

Putte and Keuning [50] tested several fingerprint sensors to check whether they accepted an artificially created (dummy or fake) finger instead of a real one. They described methods to create dummy fingers with or without cooperation (Fig. 2.2) of the real owner of the biometric (say, Alice). When the owner was cooperative (namely, Alice is helping Trudy), the quality of the dummy fingers was better than those produced without cooperation (namely, Alice is a victim of Trudy). In both cases, the authors used inexpensive and easily accessible material for creating dummy fingers. Five out of six sensors (including both optical and solid state) tested by the authors accepted a dummy finger as a real finger in the first attempt; whereas the sixth accepted it in the second attempt. The authors argued that physical properties (e.g., temperature, conductivity, heartbeat, dielectric constant) claimed to be used by the sensor manufacturers to distinguish a dummy finger from a real finger may not perform well since the detection margins of the system need to be adjusted to operate in different environments (e.g., indoor vs. outdoor) or different environmental conditions (e.g., hot summer vs. cold winter). Wafer thin silicon dummy fingers may lead to changes that are still within the detection margins of the systems.

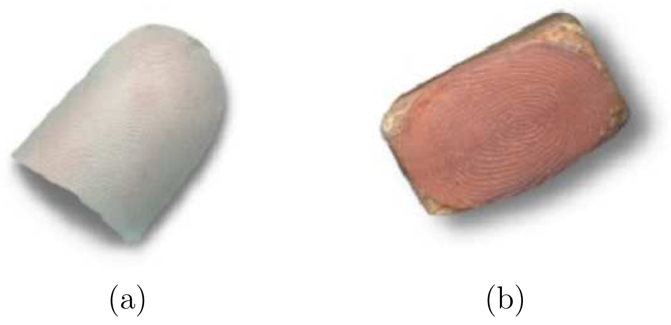


Figure 2.2: Dummy fingers: (a) finger created with cooperation of the user, (b) finger created without cooperation of the user [50].

Matsumoto et al. [35] attacked 11 different fingerprint verification systems with artificially created gummy (gelatin) fingers. For cooperative users, it was found that the gummy fingers could be enrolled in all of the 11 systems, and they were accepted with a probability of $68 - 100\%$. For non-cooperative users, all of the 11 systems enrolled the gummy fingers and they accepted the gummy fingers with more than 67% probability.

To overcome such fake biometric attacks, Derakhshani et al. [14] proposed two software-based methods (in contrast to hardware-based solutions that measure temperature, conductivity, spectroscopy [55], etc.) for fingerprint liveness detection. They used a commercially available capacitive sensor and the sole input to the liveness detection module was a 5-second video of the fingerprints. In their static method, the periodicity of sweat pores along the ridges is used for liveness detection. In the dynamic method, sweat diffusion pattern over time along the ridges is measured. A small database consisting of live fingers, fingers from cadavers, and dummy fingers made of play dough (18 sets for each of the three) were used in the experiments. A back-propagation neural network (BPNN) based classifier was used to distinguish live fingers from cadaver/dummy fingers. The static method led to an Equal Error Rate (EER) of nearly 10%, whereas the dynamic method led to an EER in the range of $11-39\%$ (a false accept event is a cadaver/dummy finger being classified as live, and a false reject event is a live finger being classified as cadaver/dummy).

Chen and Jain [72] proposed a spoof detection method based on a fingerprint deformation model using Thin-Plate Splines (TPS). They showed that, the magnitude of fingerprint deformation model can be used to differentiate between deformations originating from genuine fingers and those originating from gummy fingers made of gelatin. The physical characteristics of the material used for synthetic biometric production (e.g., gelatin) and its *inability* to mimic the genuine fingertip surface completely are used as cues for spoof detection. Experimental results including 32 gummy fingers show that a correct spoof detection rate of 82% is achievable. Fig. 2.3 shows impressions of a live finger and the associated gummy finger made from gelatin.

We can see that fake biometric attacks can be quite successful in fooling commercial fingerprint systems, and no perfect (in either hardware or software) solution is currently available. As noted previously, these attacks aim at a location in the

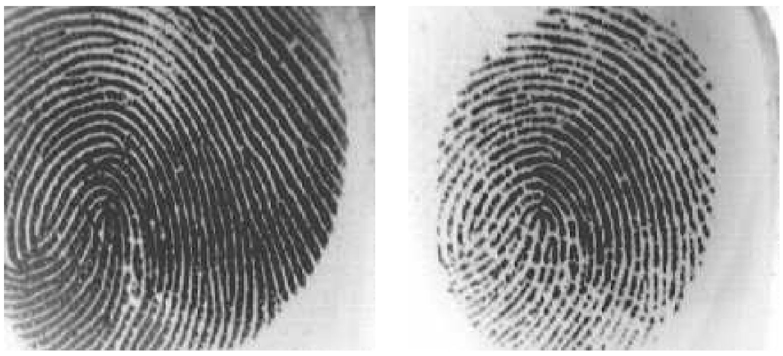


Figure 2.3: Impression from a gummy finger (a), and impression originating from the corresponding live finger (b) [72].

biometric system that is very close to the end user (in the sense that a physical replica is used). One other problem is that the means to detect (and deter) them are limited: Trudy can operate without revealing any trace of information about her identity, which may not be possible for other attacks (e.g., her computer's IP address and access statistics can be logged for digital domain attacks).

For eliminating type 2 attacks, in which a previously intercepted biometric is replayed, Ratha et al. [53] proposed a challenge/response based system: a pseudo-random challenge was presented to the sensor by a secure transaction server. At that time, the sensor acquired the current biometric signal and computed the response corresponding to the challenge (for example, specific pixel values at locations indicated in the challenge). The acquired signal and the corresponding response were sent to the transaction server where the response was checked against the received signal for consistency. An inconsistency revealed the possibility of the replay attack.

Soutar [60] proposed a hill-climbing attack against image recognition systems relying on filter-based correlation. Synthetic templates were gradually input to a biometric authentication system. Using the scores returned by the matching system, Soutar showed the system could be compromised until the point of incorrect positive identification. Outputting only quantized matching scores, not absolute scores, was proposed as a way to increase the time needed for an incorrect positive identification, thereby decreasing the practicality of this attack. This hill climbing attack can be cast as either a type 2 or a type 4 attack.

As an example of the former, Adler [1] proposed an attack on a face recognition system where the account of a specific user was attacked via synthetically generated face images (Fig. 2.4). An initial face image was selected randomly. Using the matching scores returned from the matcher that were generated for each of the successive face images, this initial image was modified. At each step, a weighted sum of eigen-images (that can be generated from public domain face databases) was added to the current candidate face image. The modified face image that led to the highest matching score was input as the new candidate image. These iterations were repeated until no improvement in matching score was observed. Experimental results on three commercial face recognition systems showed that after about 4,000 iterations, a sufficiently large matching score could be obtained, which corresponds to a very high (nearly 99.9%) confidence of matching scores.

When hill climbing is applied as a type 2 attack (before the feature extractor), the information about the template format (essential for a type 4 attack) is not necessary. Synthetic images are input to the matching algorithm, which in turn converts them into a suitable representation before matching. However, for a fingerprint-based bio-

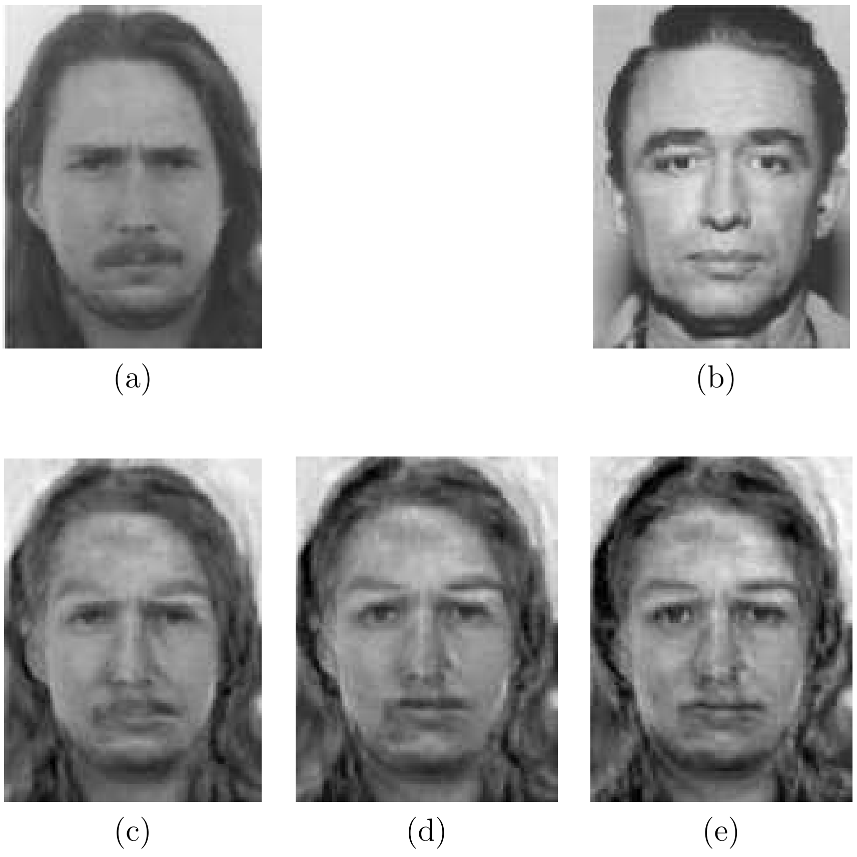


Figure 2.4: Face image synthesis: (a) initial image, (b) target image, (c) synthetic image at 200th iteration, (c) synthetic image at 500th iteration, (c) synthetic image at 4000th iteration. Figure reproduced from [1].

metric system, such an approach presents challenges not found in a face-based system: the discriminating information in fingerprints is not tied to specific geometrical relationships, as it is in face-based systems (e.g., distances between eyes, nose, mouth, etc.), and methods that are inherently linked to the correct registration of image pixels (e.g., eigen-image analysis used in [1]) seem unsuitable.

A study related to template database security (type 6 attack) is given in [20]. Using a commercial fingerprint matcher, the minutiae template data were reverse engineered by the author and the corresponding synthetic fingerprint images were generated. Although generated images were not very realistic and few experimental results are provided, the possibility of this masquerading implies that raw biometric templates may need to be secured, using, for example, techniques such as encryption.

Ross et al. [54] propose another technique to elicit the fingerprint structure from the minutiae template. Each minutia is assumed to be represented by its 2D spatial location and its local orientation. The authors first identify minutia triplets which are used to estimate the underlying orientation map: the orientation values *within* each triangular patch (whose vertices are the selected minutiae) are found by a weighted average of associated minutiae orientations. The estimated orientation map is observed to be consistent with the flow of ridges in the original fingerprint. Then, the authors use *streamlines* and *Line Integral Convolution (LIC)* for generating ridge patterns: basically, streamlines model the curves that are tangent to a vector field (orientation map) at every point. By selecting starting points of these lines as the observed minutiae locations, the method guarantees that the generated fingerprint will have minutiae at these specific coordinates. Then, LIC is used to introduce texture-like gray-level appearance to the ridges found using streamlines. Fig. 2.5 shows a fingerprint with overlaid minutiae, estimated orientation map, and the final fingerprint image generated using the minutiae information. The authors also quantitatively evaluated the similarity between original fingerprint images and the images generated using the outlined algorithm for a subset of NIST-4 database comprised of 2,000 fingerprints: the synthetically generated images were accepted as "real" fingerprint images by a commercial authentication system with a probability of around 23%. This is indicating a serious security breach for the current systems and the authors conclude that further research for increasing the security of minutiae templates is necessary.

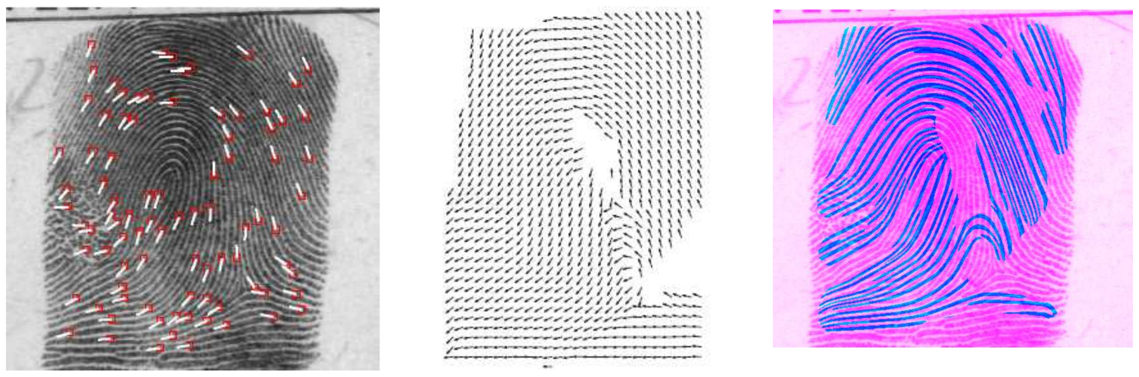


Figure 2.5: Fingerprint regeneration using minutiae: (a) Input minutiae, (b) estimated orientation map, (c) regenerated fingerprint overlaid on original fingerprint [54].

A method to protect templates from fraudulent usage involves using a distorted (but non-invertible) version of the biometric signal or the feature vector [53]; if a specific representation of template is compromised, the distortion transform can be replaced with another one from a transform database. Every application (e.g., health care, visa, e-commerce) can use a different transform so that the privacy concerns of subjects related to database sharing between institutions can be addressed. Data hiding and watermarking techniques have also been proposed as means of increasing the security of fingerprint images (for more information, see Chapter 3), by detecting modifications [48], by hiding one biometric into another [24] and by hiding messages (authentication stamps such as personal ID information) in the compressed domain [53].

In the following section, we propose a system that uses hill-climbing as a type 4 attack that bypasses the feature extractor and uses synthetically generated feature sets in the realm of a minutiae-based fingerprint matcher. A preliminary version of this research appeared in [67].

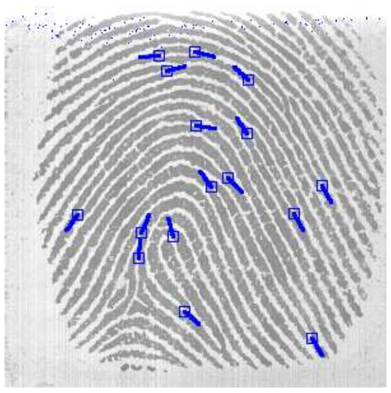
2.3 Architecture of the Proposed Attack System

2.3.1 Basic Structure

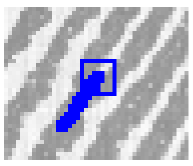
The proposed system attacks a minutiae-based fingerprint authentication system. A minutiae-based system is chosen as the test bed because minutiae information is used in most of the commercial fingerprint authentication systems. Hence, observed results can provide insights for securing them.

In typical minutiae-based fingerprint authentication systems, discontinuities in the flow of ridges (ridge bifurcations and endings) constitute the minutiae. Fig. 2.6 shows a sample fingerprint image with overlaid minutiae, and close-up views of these two types of minutiae. Generally, the minutiae type information is not used as a feature in the fingerprint matchers since the changes in finger pressure on the sensor can change one type of minutia into the other. The majority of minutiae based systems use the location (c, r) (denoting column and row indices, respectively) of the minutiae and orientation θ associated with the minutiae as features; but some systems use additional information such as ridge flow around the minutiae and ridge counts between the minutiae.

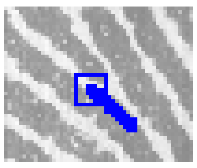
Consistent with this approach, we use (c, r, θ) attributes for each minutia. This is also in accordance with the proposed minutiae template exchange format of Bolle et al. [4], which excludes proprietary features, and encompasses only the location, orientation, and type of the minutia.



(a)



(b)



(c)

Figure 2.6: Minutiae features: (a) fingerprint image with overlaid minutiae, (b) ridge bifurcation, (c) ridge ending.

The proposed attack system inputs synthetic minutiae sets (comprised of (c, r, θ) triplets) to the fingerprint verification system with the aim of gaining access to the system in place of a genuine user. It is assumed that the format of the user's template is known, but the actual contents of this template are not known to the attacker.

The attacker inputs a single synthetic fingerprint template and observes the matching score between this input template and the target template in the database. By repeating this for successively generated templates, the attacker program tries to generate a synthetic template that results in a matching score higher than the system threshold. The block diagram of the proposed system is given in Fig. 2.7.

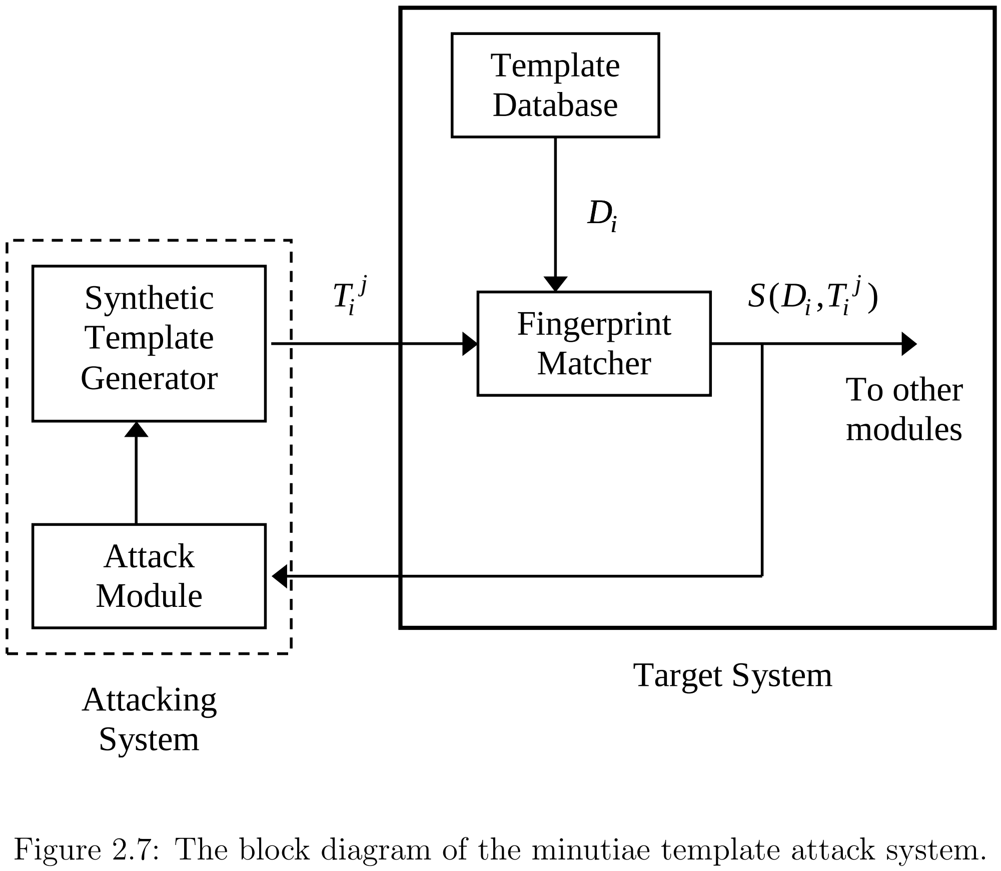


Figure 2.7: The block diagram of the minutiae template attack system.

2.3.2 Notation

Here, the symbols used in subsequent sections are defined:

- D_i: The database template corresponding to user i, $i = 1, 2, ..., N$, where N is the total number of users registered in the system.

- n_i: The total number of minutiae in D_i. Note that the attacker does not know this value.

- $T_i{}^j$: The jth synthetic template generated by the attacker for user i. In the proposed system, it is represented as

$$T_i^j = \begin{pmatrix} {}^1c_i^j & {}^1r_i^j & {}^1\theta_i^j \\ {}^2c_i^j & {}^2r_i^j & {}^2\theta_i^j \\ \vdots & \vdots & \vdots \\ {}^{n_{ij}}c_i^j & {}^{n_{ij}}r_i^j & {}^{n_{ij}}\theta_i^j \end{pmatrix} \tag{2.1}$$

  where each row represents column index (pixels), row index (pixels), and orientation angle (degrees) associated with a minutia; the upper left hand subscript denotes the minutiae index, so the total number of minutiae in $T_i{}^j$ is n_{ij}.

- $S(D_i, T_i{}^j)$: The matching score between D_i and $T_i{}^j$.

- $S_{threshold}$: The decision threshold used by the matcher; the matcher accepts the input template if it generates a score higher than this value, otherwise, it rejects the template. Note that the attacker does not know this value.

2.3.3 Attack Steps

The attacker can try breaking multiple accounts (e.g., by attacking all the accounts in the target database sequentially), not just one, to increase her chances of gaining access to the system. Breaking just one account may suffice for Trudy to harm the protected entities. But for the sake of clarity, it is assumed that the attacker tries to break into the account of the ith user:

- Step 1 (Initial guess): Generate a fixed number of synthetic templates. In the current implementation, 100 random minutia templates (${T_i}^1$, ${T_i}^2$, ${T_i}^3$, ... ,${T_i}^{100}$) are created for attacking user i's account.
- Step 2 (Try initial guesses): Attack the i^{th} user account with the templates generated in Step 1. Let the corresponding matching scores be $S(D_i, {T_i}^1)$, $S(D_i, {T_i}^2)$, $S(D_i, {T_i}^3)$, ... , $S(D_i, {T_i}^{100})$.
- Step 3 (Pick the best initial guess): Declare the best guess (${T_i}^{best}$) to be the template resulting in the highest matching score, $S^{best}(D_i)$.
- Step 4 (Modify input template): Modify ${T_i}^{best}$ by (i) perturbing (the location or angle of) an existing minutia, selected randomly, (ii) adding a new minutia, (iii) replacing an existing minutia, selected randomly, and (iv) deleting an existing minutia, selected randomly. If for any one of these attempts, the matching score is larger than the current $S^{best}(D_i)$, declare the modified template as ${T_i}^{best}$, and update $S^{best}(D_i)$ accordingly. Otherwise, do not change ${T_i}^{best}$.
- Step 5 (Stopping criterion): If the current best score $S^{best}(D_i) > S_{threshold}$, i.e.

the attacker gains access, stop the attack (since account is broken); otherwise, go to Step 4. Note that the attacker does not know the value of $S_{threshold}$.

2.3.4 Specifications

It is assumed that the resolution (in dpi) and size (in pixels) of the images generating the original templates are known to the attacker. This is a valid assumption since these values are generally announced by sensor manufacturers. For the current implementation, the MSU-VERIDICOM fingerprint database, comprised of 500 dpi, 300x300 pixel images, is used. The image size is used in generating the locations of synthetic minutiae, whereas the resolution determines the inter-ridge distance (approximately 9 pixels for 500 dpi images) associated with the fingerprints. To ensure that synthesized minutiae are not too close to each other, minutiae are placed in the centers of a 9x9 tessellation grid on the image. This way, the attack system does not create minutiae that are closer than the inter-ridge distance. The angle value θ associated with a minutia is generated randomly as a quantized value (in increments of 22.5^o) in the range [0, 360), with 16 equally spaced intervals.

In Steps 1-3, the aim is to find a good initial guess and concentrate on modifying it. The reason is that, if the initial guess is poor (namely, it has very low similarity with the target template), the attacker may need more iterations to break into the account. These initial templates all have the same number of minutiae (n_{ij} = 25, $\forall i$, $j = 1, 2, 3, ..., 100$). Note that the actual number of minutiae in the target template (n_i) is not known to the attacker, and the value of 25 is selected to be a typical number for sensors used in our experiments. Further, the algorithm modifies the template set so that the number of minutia can increase or decrease freely based on matching scores.

Step 4 is repeated until the account is broken (or a predefined iteration count is reached). It is composed of 4 iterations in each loop of the process:

- An existing minutia in the input template is randomly picked, and its location or angle is modified slightly, each with probability of 0.5. Here, the aim is to change either the location or angle of the selected minutia and see its effect on the matching score. The minutia location in column (c) and row (r) axes is perturbed with a distance equal to the inter-ridge distance (namely, a minutia moves to one of the (at most 8) neighboring cells). The angle is perturbed so that it is increased or decreased to the next angle quantum (hence, it changes ± 22.5 degrees in the current system).

- A new randomly created minutia is added to the current template. It follows the same specifications as the initial template minutiae generation.

- An existing minutia is randomly picked and replaced with a randomly created minutia, which follows the same specifications as the initial template minutiae generation.

- An existing minutia is randomly picked and deleted from the current template.

Note that the ordering of these modifications is based on the amount of change introduced to the existing template: the attacker starts with modifications that are small in magnitude (perturb existing minutia), and continues with modifications of relatively higher magnitude (delete an existing minutia).

After each iteration (submitting the synthetic template and observing its matching score), if the matching score improves, the attacker replaces the current template with the new template; otherwise the attacker keeps the current template as the best template. So, the algorithm *hill-climbs* in the matching score space, always trying to improve the input template.

2.3.5 Information Available to the Attacker

As explained above, the attacker needs to observe the matching scores during the hill-climbing procedure. Note that this assumption is not very restrictive; a majority of the commercial systems reveal the matching scores. But it should be noted that if the templates are encrypted before the matcher accepts them, this attack is not feasible without the knowledge of the correct decrypting key(s).

In situations where the matcher just outputs the decision (accept/reject), but not the matching score, the attacker can only submit randomly generated templates. In this case, the performance of the attack will be determined by the Receiver Operating Characteristics (ROC) operating point of the attacked biometric system. For example, if the system False Accept Rate (FAR) is set to 0.1%, the attacker can break a biometric account by trying 1,000 synthetic templates, on the average. It is shown later in the experimental results section that if the matching scores are available then our attacker can break the accounts with a significantly fewer number of attempts.

2.4 Incorporating Fingerprint Class Information

In this section, we explain the utilization of the fingerprint class information to increase the effectiveness of the basic attack system outlined in Section 2.3. The modules given in this section generate minutiae sets based on the constraints specified by the fingerprint class information.

2.4.1 Utilization of Fingerprint Class Prior Probabilities

Fingerprints are generally classified into 5 major classes: arch (A), tented arch (TA), left loop (LL), right loop (RL) and whorl (W). These are based on points of abrupt changes in the regular ridge flow directions, called singularities (cores and deltas) [34]. Fig. 2.8 shows sample fingerprint images from each of the classes with overlaid core and delta points. Generally, the classes A and TA are treated as a single class (ATA) since it is hard to separate them from each other. Even this augmented ATA class makes up only a small fraction of the fingerprints (approximately 7%).

In reality, the attacker does not know the class of the target fingerprint template. Nevertheless, prior class probabilities are known [34]: P(LL) = 0.338, P(RL) = 0.317, P(W) = 0.279, P(ATA) = 0.066. Therefore, the attacker can use these prior probabilities to guess the class of the target template. The attacker mounts an attack assuming that the class of the target is the guessed class (how this class label is used in the attack procedure is explained in the following sections); if the attack is not successful in a predefined number of attempts (300 per class in the current implementation), the attacker guesses another class and continues the attack.

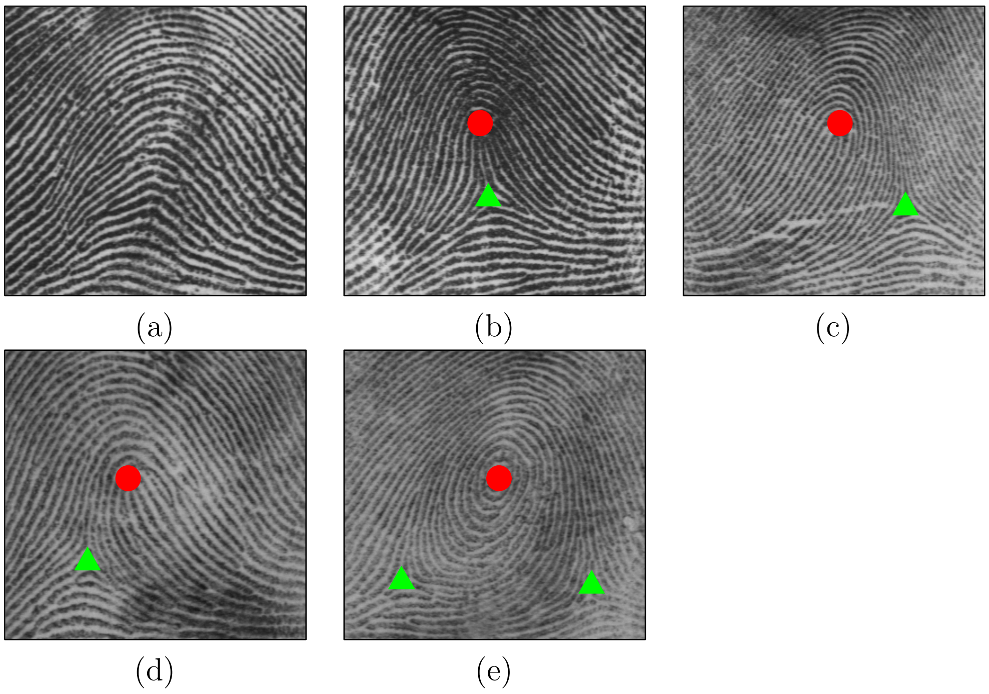


Figure 2.8: Fingerprint classes: (a) arch, (b) tented arch, (c) left loop, (d) right loop, and (e) whorl. Core points are shown with a circle, delta points are shown with a triangle.

2.4.2 Class-Specific Spatial Minutiae Presence Probabilities

The spatial locations of the minutiae generated by the attacker can follow a uniform probability distribution on the 2D image grid. Note that for the 300x300 target images considered in this study, the 2D grid in which the minutiae are generated has the size of 33x33 blocks (considering 9 pixel inter-ridge distance). Uniform distribution dictates that a minutia can occur in any of these blocks with 0.00092 (=1/(33x33)) probability.

The spatial minutiae presence probability is defined as the probability that for a given fingerprint class, a minutia occurs in a specific block (i, j), $i = 1, 2, 3, ..., 33$, $j = 1, 2, 3, ..., 33$. To estimate these probabilities, the NIST 4 database [42], which contains 2,000 image pairs (512x512 in size, 500 dpi) classified by a human expert, is used. This database contains an equal number of images for each of the 5 classes (A, TA, LL, RL, and W, 800 images each). Further, 700 images are *cross-referenced*, i.e., they bear two "true" class labels. This cross-referencing occurs when the expert could not decide on a single class label for an image.

Firstly, images corresponding to 4 classes (TA, LL, RL, and W) are categorized, by also including cross-referenced images. Then, for each of these 4 classes, the minutiae locations are determined (using the algorithm in [23]) and the core locations are found (using the algorithm in [10]). If more than one core point is found for an image (possible for the whorl class), the upper core is selected as the representative one. The minutiae set is translated (horizontally and vertically) so that the core point aligns with the 2D image center, namely at coordinate (256, 256), in order to eliminate the effect of translation in probability estimates. Then, the spatial probability of minutiae ($p_C(r,c)$, where r denotes the row index and c denotes the column index) is estimated for each class C ($C \in \{LL, RL, W, TA\}$) via Parzen window density estimation technique with a bivariate Gaussian window function (with independent components each with variance σ^2) as

$$p_C(r,c) = \frac{1}{N_C}\sum_{i=1}^{N_C} \frac{1}{2\pi\sigma^2} e^{-0.5 * \frac{(r-r_i)^2 + (c-c_i)^2}{\sigma^2}} \tag{2.2}$$

where N_C is the total sample number for the class, and (r_i, c_i) are the row and column indices of the minutiae sample over which the Gaussian window with variance σ^2 is placed to estimate the density. Fig. 2.9 shows this window function.

Fig. 2.10 shows a sample image from the NIST 4 database (belonging to the LL class), along with the core point, 2D image center, and overlaid minutiae. It has 55 minutiae and the core is at coordinate (295, 240).

Figures 2.11, 2.12, 2.13, and 2.14 show the minutiae presence probabilities for the 4 fingerprint classes, estimated using the Parzen window technique when the variance of Gaussian window σ^2 is 3. Note that the so-called *window width* of utilized window function (proportional to σ^2) affects the extracted densities drastically in this technique [16]: small widths lead to noisy (i.e. too dependent on the sample points) estimates, whereas large widths result in very smooth (i.e. with little dependency on the sample points) estimates. We have found that the value $\sigma^2 = 3$ provides a good trade-off between these characteristics and we have selected it for our experiments.

From these figures, it is observed that the minutiae are not distributed uniformly

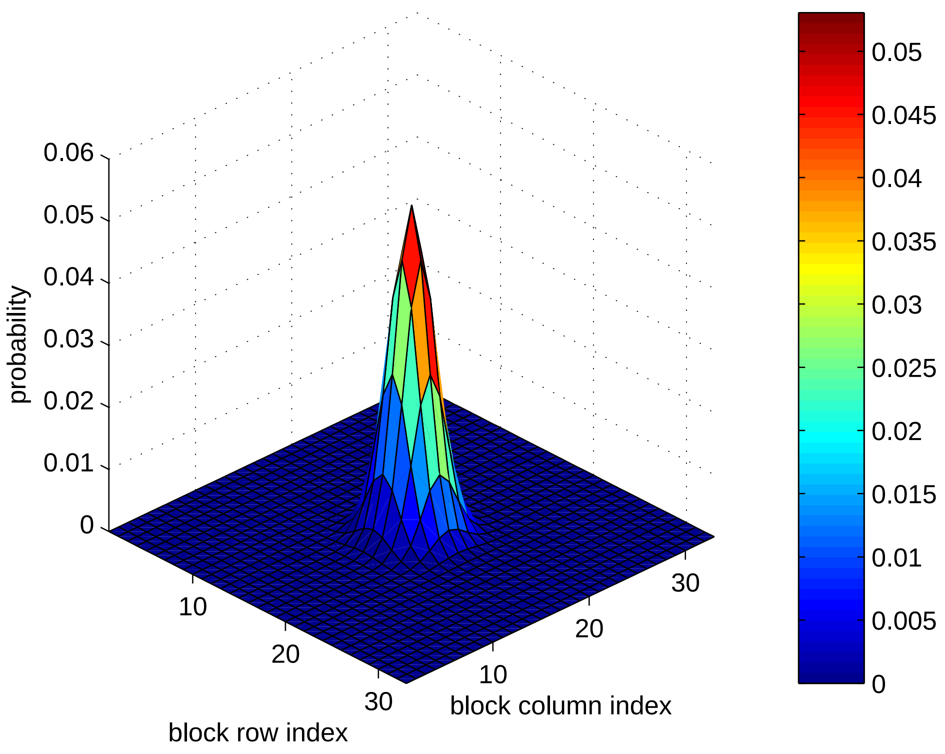


Figure 2.9: Bivariate Gaussian window function with $\sigma^2 = 3$.

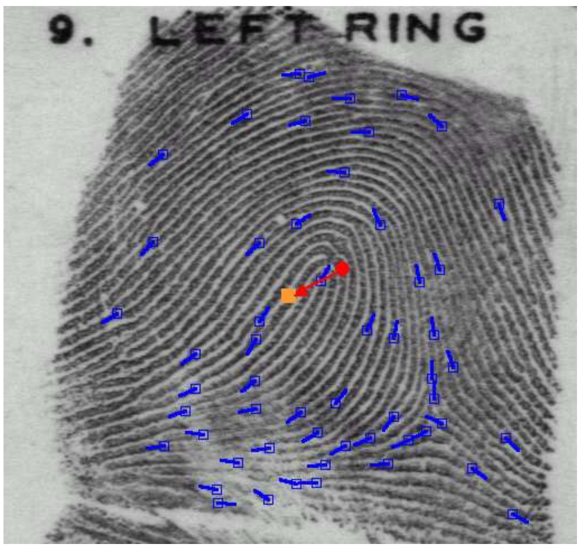


Figure 2.10: Sample NIST 4 image with core, image center, and overlaid minutiae.

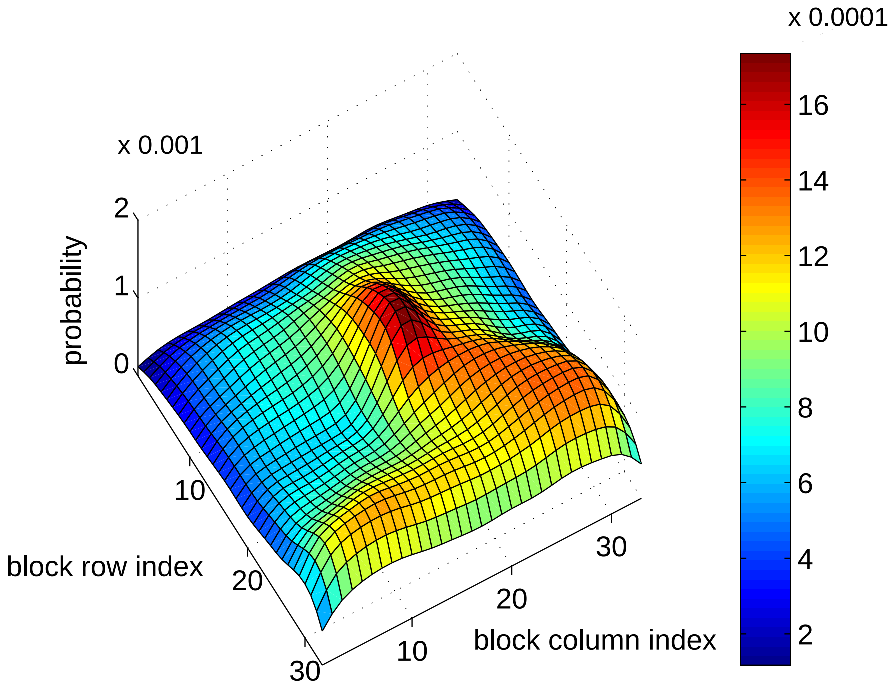


Figure 2.11: Minutiae presence probability distribution for the left loop (LL) class.

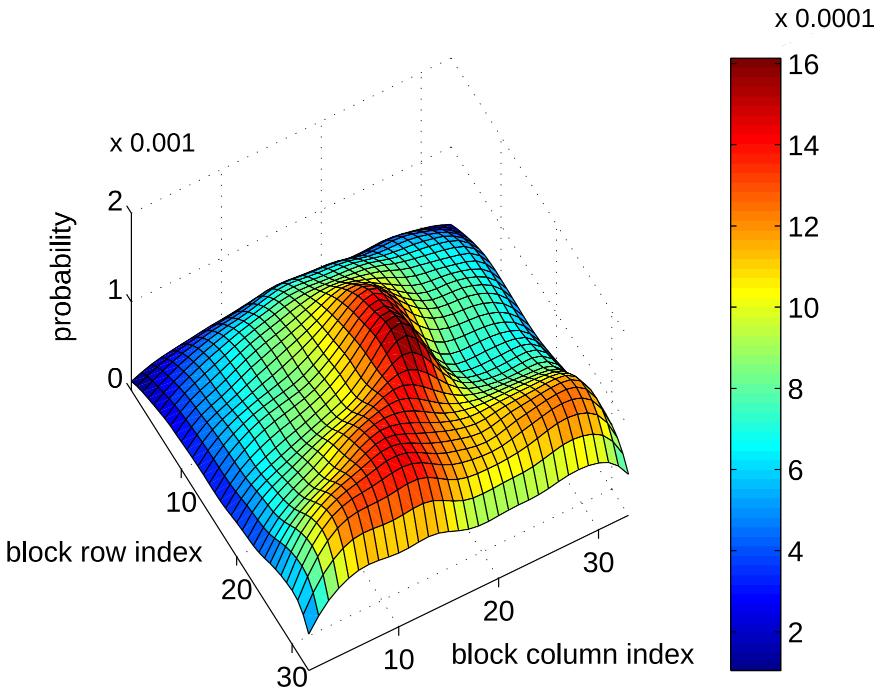


Figure 2.12: Minutiae presence probability distribution for the right loop (RL) class.

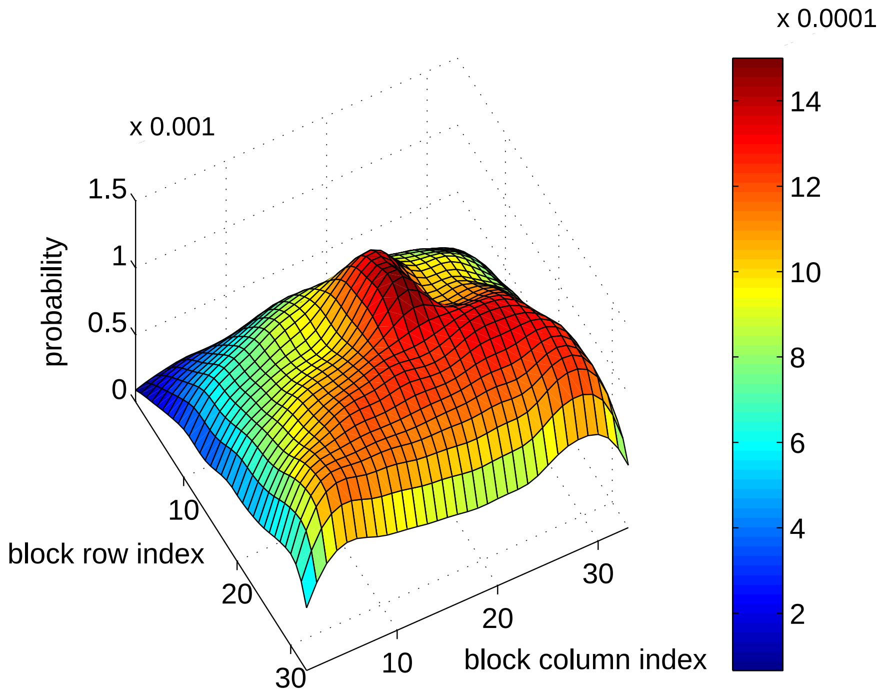


Figure 2.13: Minutiae presence probability distribution for the whorl (W) class.

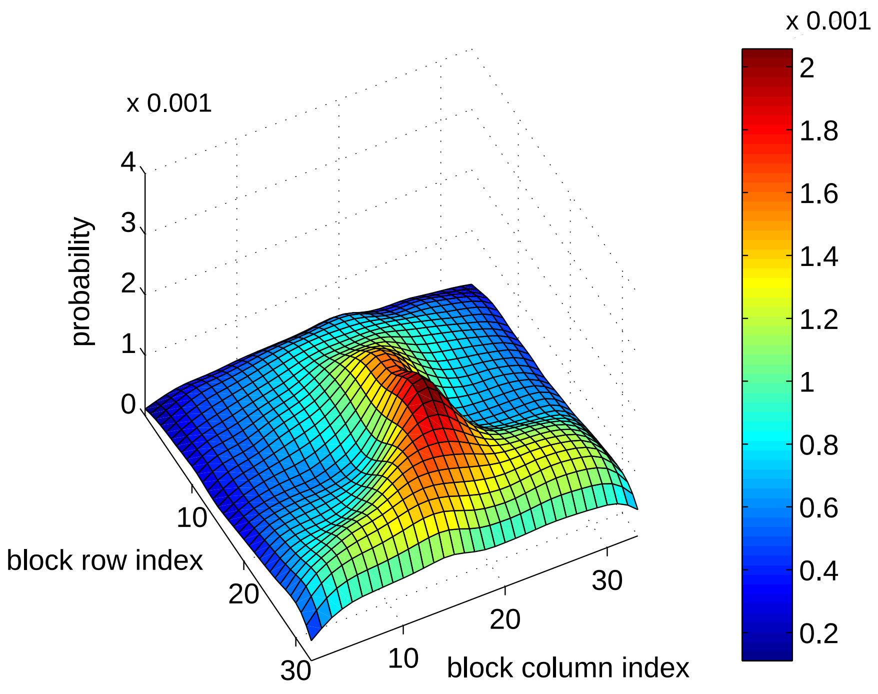


Figure 2.14: Minutiae presence probability distribution for the arch/tented arch (ATA) class.

on the 2D grid; their location depends on the fingerprint class. In all the classes, minutiae are more likely to occur below the core compared to above the core, and the area around the core has a higher probability of observing a minutia. For left loop, ridges enter and leave on the left side of the image and consequently, the minutiae presence probability is higher on the right side. For the right loop, ridges enter and leave on the right side of the image, so the minutiae presence probability is higher on the left side.

Assuming a particular fingerprint class, the attacker randomly generates minutia position by using the respective probability distributions. This increases the effectiveness of the attack since it increases the chance of mimicking the actual target template minutiae.

2.4.3 Class-Specific Orientation Fields

Until this point, the attacker assumed the class of the target template, and generated 2D minutiae locations (c, r) based on respective minutiae presence probability distributions. The remaining feature of the minutiae, angle θ, is found by making use of the ground truth orientation field associated with these classes, as explained below.

Fingerprint classes (LL, RL, W, and ATA) have characteristic orientation fields, hence minutiae directions (dictated by the ridge orientations) are dependent on the 2D location of minutiae for each class. The orientation fields of 4 representative images (one from each class) from the NIST 4 database were obtained (using the method outlined in [23]) and used as the true orientation fields. These 4 images were selected based on the location of their core (it should be near the image center) and the foreground fingerprint area (it should be large to have a complete orientation field). Fig. 2.15 shows the corresponding orientation fields. These orientation fields indicate the orientation ([0, 180)) angle of a minutia at (c, r), given the fingerprint class. The attacker randomly selects one of two directions (in the range [0, 360)) associated with the orientation as the direction for that specific minutia.

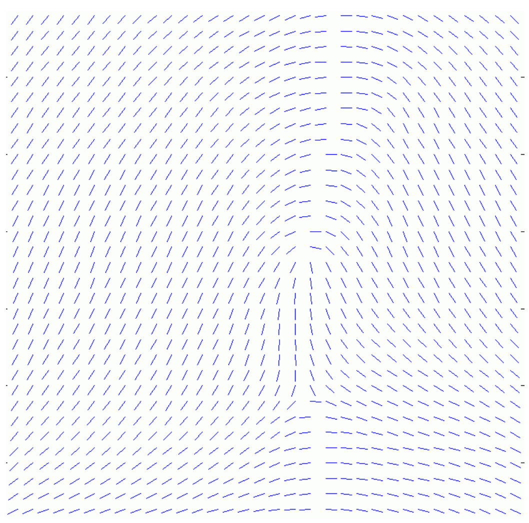
2.5 Experiments and Results

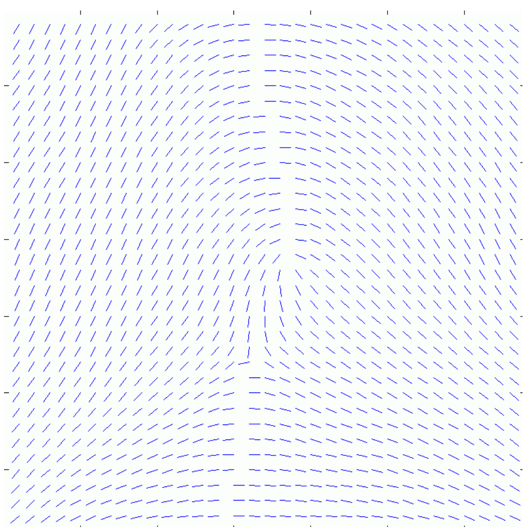
In this section, first the feasilibility of the attack is demonstrated, then a safeguard against the attack system is proposed.

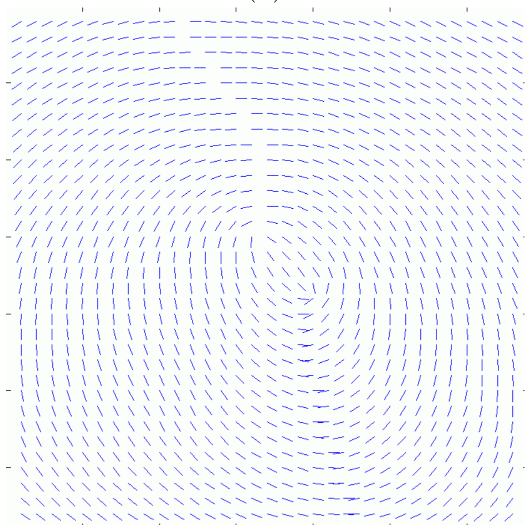
2.5.1 Feasibility of the Attack

The minutiae-based fingerprint matcher attacked in this study is a modified version of the matcher described in [23]. Jain et al. [23] used the ridge sample coordinates, in addition to (c, r, θ) triplets associated with each minutia. To simplify the attack system, their original matcher is modified, using only (c, r, θ) triplets as the fingerprint template.

The right-index finger segment of the MSU-VERIDICOM fingerprint image database (160 users, 4 impressions/finger, obtained with a VERIDICOM solid state sensor, 500 dpi 300x300 images) is used in the experiments. First, for setting a decision threshold for the system, the minutiae for each of these 640 images are extracted using the extractor described in [23], and the matcher is run on these minutia feature


(a)


(b)


(c)

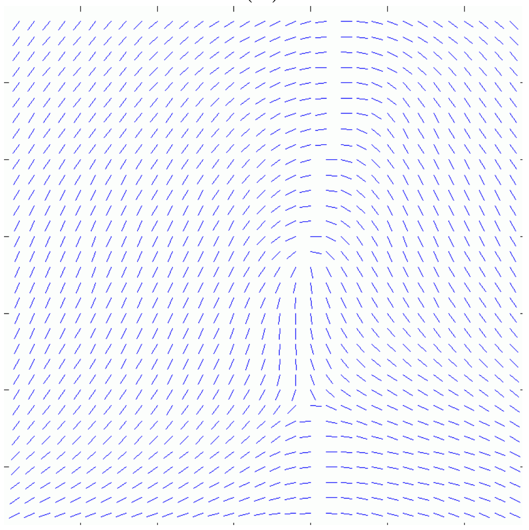

(d)

Figure 2.15: Representative orientation fields: (a) left loop, (b) right loop, (c) whorl, and (d) arch/tented arch. Image size is 300x300, and block size is 9x9.

sets. Hence, a total of 1,600 genuine and 203,520 imposter scores are obtained (scores are normalized to the range [0, 100]). Fig. 2.16 shows the associated ROC curve.

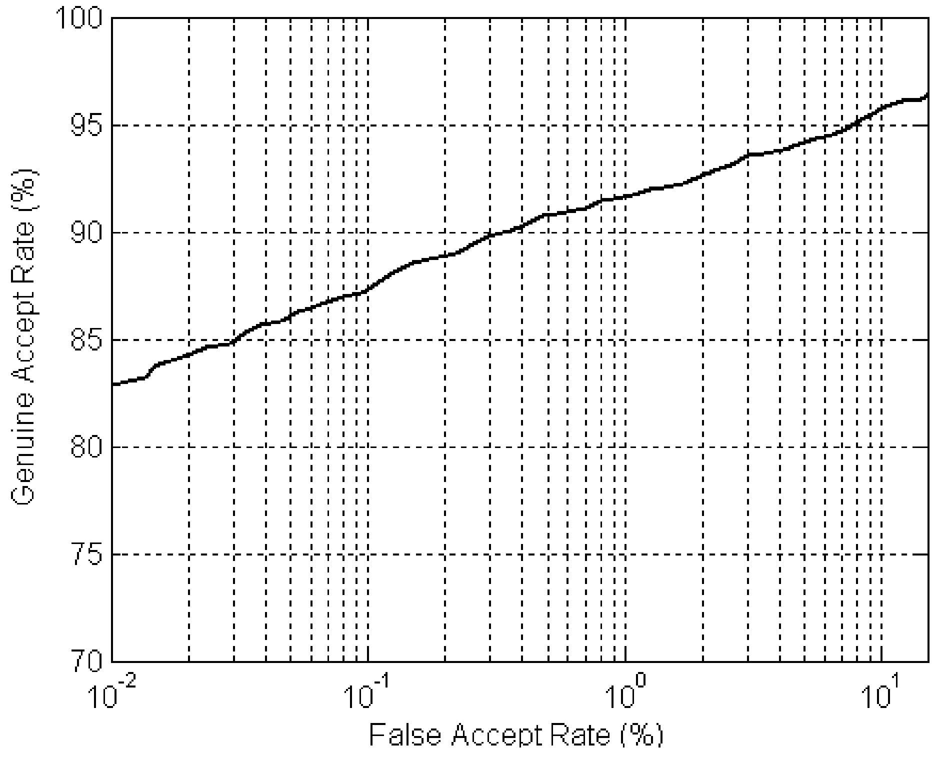


Figure 2.16: ROC curve for the attacked fingerprint matcher.

As a sample operating point for the fingerprint verification system, the setting where FAR = 0.1%, and GAR = 87.6% is considered. The corresponding decision threshold is found to be 12.22. The value of 0.1% FAR is typically used by administrators of biometric systems. Note that the associated threshold value (namely, 12.22) is not known to the attacker. Selecting this specific threshold (12.22 for which FAR=0.1%) implies that, on an average, 1 in 1,000 random imposter attempts will be accepted as a genuine match.

The proposed attack system broke all of the 160 accounts in the database with less than 1,000 attempts for each account. In fact, the minimum, mean, and the maximum number of attack attempts that are required for breaking into all the accounts are found to be 107, 182, and 531, respectively. Fig. 2.17 shows the histogram of the number of required attempts at which the accounts were broken. Note that the attacker used fingerprint class information as explained in Section 2.4 for the experimental results reported here.

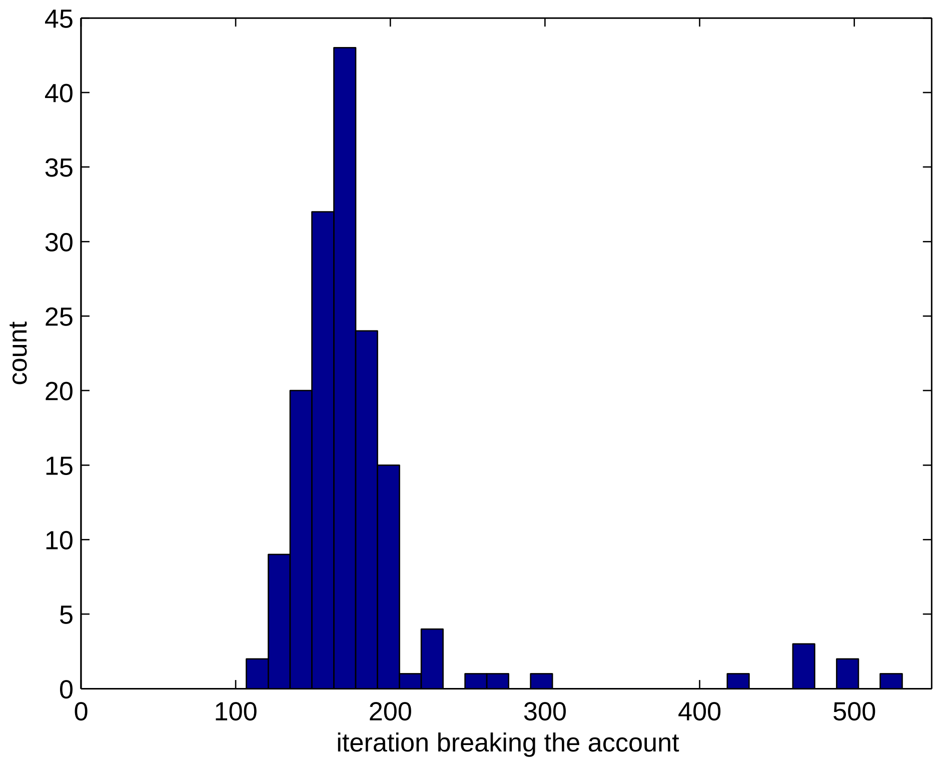


Figure 2.17: Histogram of the number of attempts at which the accounts were broken.

The mean number of required iterations, 182, is surprisingly small, considering that 100 iterations are already required to pick the best initial guess; hence, in approximately 80 additional iterations, the attacker was able to synthesize templates that broke the accounts.

The minimum, mean, and the maximum number of minutiae in the templates that broke the accounts were found as 10, 14.2, and 21, respectively. The minimum, mean, and the maximum number of matching minutiae between the original template and the templates that broke the accounts were 5, 6.8, and 10, respectively. For comparison, the minimum, mean, and the maximum number of matching minutiae for successful genuine matches were 5, 13.5, and 30, respectively.

It is seen that during the hill-climbing procedure, the number of minutiae in the synthetic template is reduced to an average of 14.2 (note that the attacker starts with 25 minutiae in the initial synthesized template). This is probably due to the fact that the attacker finds a salient subset of the original target minutiae. Further, the number of matching minutiae (6.8, on the average) for the synthetic templates may seem small, but note that this was sufficient for positive authentication at the chosen operating point (FAR = 0.1%) of the biometric system.

To demonstrate the effect of random number generation seed used in the algorithm, experiments were repeated for 20 different seeds. The minimum, mean, and maximum values (over these 20 instances) for the mean attempt number were 181, 193, and 210, respectively. Hence, although there is some variation in the number of required attempts due to different seed values, it is not large (standard deviation of the mean number of attempts is 7.8). It is found that the account difficulty (reflected via the number of required attempts at which account is broken) does not depend much on the target template, but it is mainly determined by the system settings (system operating point).

For analyzing the progression of matching scores for a specific account, the account number 11 is selected. This account is broken at iteration number 192. Fig. 2.18 shows the original target image with overlaid minutiae, the synthetic minutiae set that broke the account and original minutiae, and the progression of matching scores. The original template has 16 minutiae, whereas the synthetic template has 10, 5 of which match with those of the target template. The final matching score is 13.3.

For observing the evolution of synthetic templates for the same account, the corresponding minutiae for a few iterations are shown in Fig. 2.19. It is observed that several "incorrect minutiae" are deleted to arrive at a small subset that leads to a high matching score.

2.5.2 Safeguards Against Minutiae Attack

In the previous section, the feasibility of the proposed attack system was demonstrated. Here, we propose to apply a masking operator on the output matching scores in order to alleviate this attack. This masking operator alters the scores randomly without affecting the accept/reject decision, so the matcher does not return the actual matching score between the target template and synthetic template. Instead, the matcher returns a random score that is *smaller* than the preset decision threshold (i.e., $S_{threshold} = 12.22$) for unsuccessful attempts (for successful attempts, there is no need to mask the scores):

$$S(D_i, T_i^j) \leftarrow (S_{Threshold} - \epsilon)RAND, \tag{2.3}$$

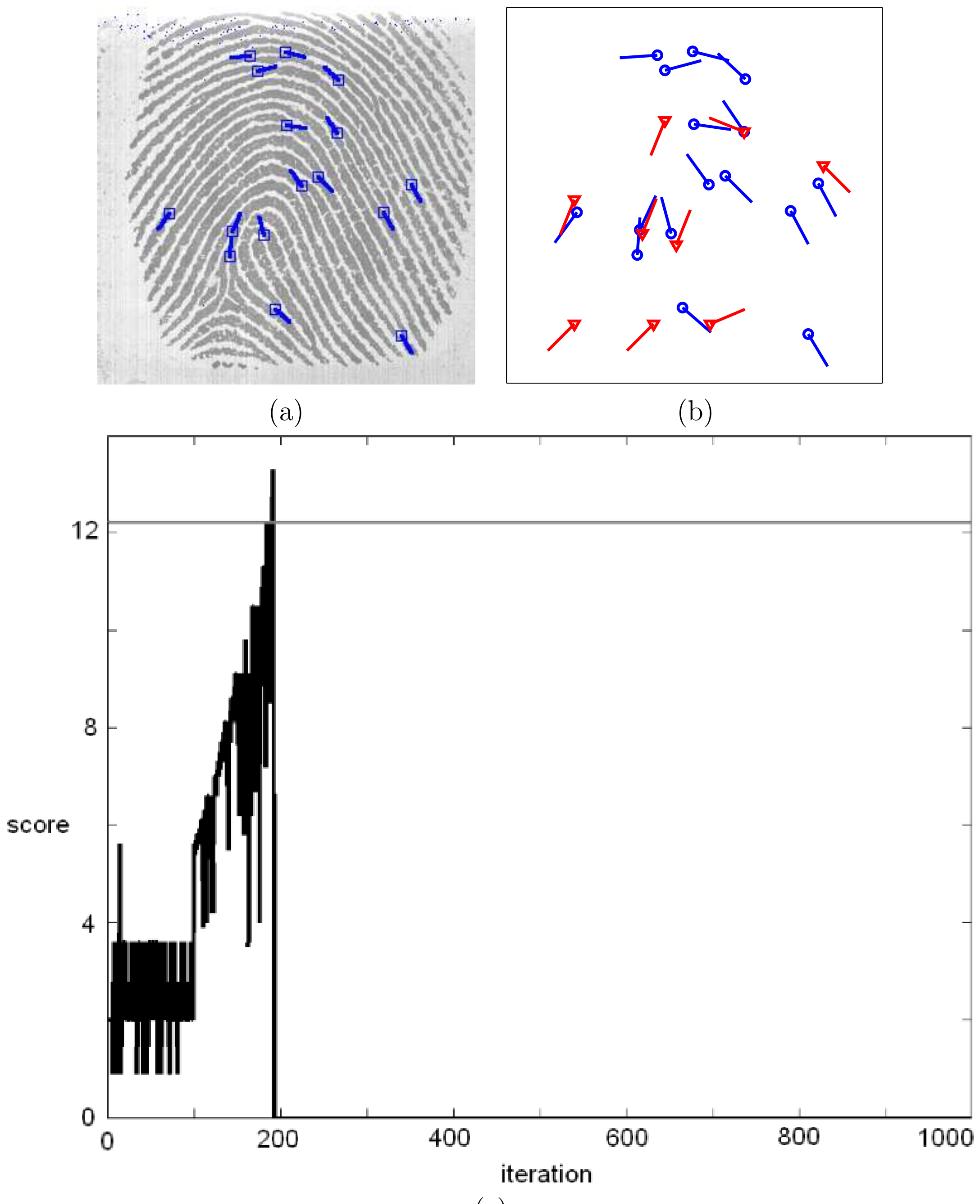


Figure 2.18: A sample account is broken: (a) target template with overlaid minutiae, (b) synthetic minutiae (red) and the original minutiae (blue) when the account is broken, (c) progression of matching scores (decision threshold is shown as the horizontal line).

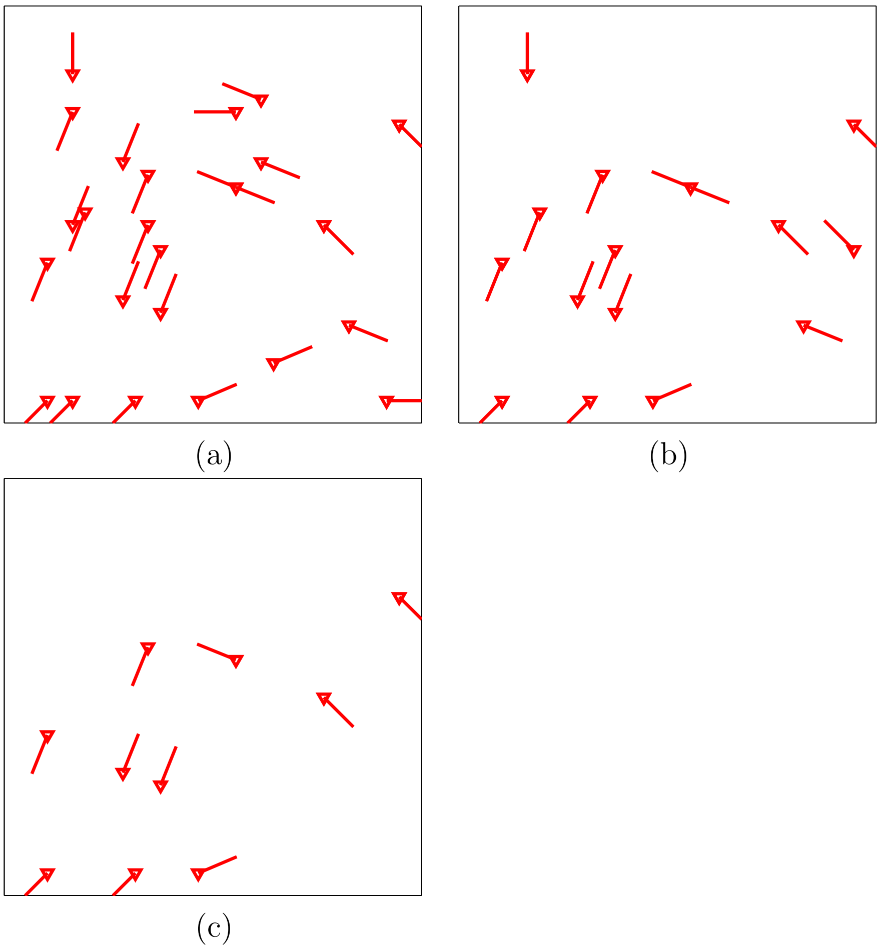


Figure 2.19: Synthetic template evolution for the account in Fig. 2.18: (a) best initial guess (matching score=5.6), (b) at iteration 150 (matching score=8.6), (c) at iteration 175 (matching score=10.5).

where ϵ is a small constant greater than zero ($\epsilon = 0.01$ in the current implementation), and $RAND$ is a random number in the range [0, 1]. Since the accept/reject decision is not affected by this mask, the masked score can still be used by the biometric system to a limited extent. But it eliminates the possibility of matching score fusion for a multimodal system, e.g., the system in [59], where matching scores from different modalities are fused together to arrive at a decision.

Here the information leaked from the matcher (masked score) is not the actual score, so the attacker wanders in the search space without actually improving the template. In fact, when the scores are masked, the attacker did not break any of the 160 accounts before the maximum iteration number (1,200 in the current implementation) was reached. Comparing this with the fact that the attacker broke each of the 160 accounts with 182 attempts on average when the scores were not masked, masking the scores seems an effective way for repelling the attacker.

Another simple, but effective solution is to block matching attempts if there are too many false matches in a given period of time (e.g., it is highly unlikely that a legitimate user can provide more than, for example, 20 false matches per day). Successive attempts may indicate an attack by an imposter, as proposed above, for a specific template. If the attacker has enough time, even this measure may not be very effective. For example, the attacker can accumulate results over multiple days: if 200 iterations are necessary for breaking into an account, the attacker can mount an attack that lasts 10 days (with 20 iterations/day) and still manage to break into the account.

Further, as seen above, the attacker generates synthetic templates that have a relatively small (but still enough for positive authentication) number of matching minutiae. This can be taken into account in the design of the minutiae matcher. For instance, the matcher can be modified so that it rejects templates having number of matching minutiae less than a threshold, at the expense of increasing the False Reject Rate.

2.6 Summary

As the security of biometric systems is gaining widespread attention, numerous attack systems have been proposed by researchers. In this chapter, we proposed a hill-climbing based minutiae attack system that is quite effective for breaking into the accounts of a small fingerprint database comprised of 160 users. It needs just 182 attempts, on average, to break each one of the accounts. Utilization of fingerprint class prior probabilities, spatial minutiae presence probabilities, and class-based orientation fields improves the effectiveness of the attacker. Without these, the attacker needs 271 attempts [67], on average, to break each account. Further, estimated minutiae presence probabilities (with distinct fingerprint class dependencies) can also be used in fingerprint minutiae individuality studies [47].

The proposed score masking procedure decreases the effectiveness of the attacker considerably by eliminating the correlation between the controlled changes the attacker introduces to the synthetic templates and the returned matching scores. With masking, hill-climbing based attack system could not synthesize a valid minutiae template.

Chapter 3

Watermarking for Enhancing Security of Biometric Systems

3.1 Introduction

While biometric techniques have inherent advantages over traditional personal identification techniques, the problem of ensuring the security and integrity of the biometric data is critical. For example, if a person's biometric data (e.g., her fingerprint image) is stolen, it is not possible to replace it, unlike replacing a stolen credit card, ID, or password. Schneier [57] points out that a biometrics-based verification system works properly only if the verifier system can guarantee that the biometric data came from the legitimate person at the time of enrollment. Furthermore, while biometric data provide uniqueness, they do not provide secrecy. For example, a person leaves fingerprints on every surface she touches and face images can be surreptitiously observed anywhere that person looks. Hence, the attacks that can be launched against biometric systems (summarized in Chapter 2) have the possibility of decreasing the credibility of a biometric system.

In order to promote widespread utilization of biometric techniques,an increased security of biometric data, especially fingerprints, is necessary. Encryption, watermarking and steganography are possible techniques to achieve this. Steganography, derived from the Greek language and meaning secret communication, involves hiding critical information in unsuspected carrier data. While cryptography focuses on methods to make encrypted information meaningless to unauthorized parties, steganography is based on concealing the information itself. As a result, steganography-based techniques can be suitable for transferring critical biometric information, such as minutiae data, from a client to a server. Steganographic techniques reduce the chances of biometric data being intercepted by pirates during transmission, hence reducing the chances of illegal modification thereof. Digital watermarking techniques can be used to embed proprietary information, such as a company logo, in the host data to protect the intellectual property rights of that data [19]. They are also used for multimedia data authentication. Encryption can be applied to biometric templates for increasing security; the templates (that can reside in either (i) a central database, (ii) a token such as smart card, (iii) a biometric-enabled device such as a cellular phone with fingerprint sensor) can be encrypted after enrollment. Then, during authentication, these encrypted templates can be decrypted and used for generating the matching result with the biometric data obtained online. As a result, encrypted templates are secured, since they cannot be utilized or modified without decrypting them with the correct key, which is typically secret. However, one problem associated with this system is that encryption does not provide security once the data is decrypted. Namely, if there is a possibility that the decrypted data can be intercepted, encryption does not address the overall security of the biometric data. On the other hand, since watermarking involves embedding information into the host data itself (e.g., no header-type data is involved), it can provide security even after decryption. The watermark, which resides in the biometric data itself and is not related to encryption-decryption operations, provides another line of defense against illegal utilization of the biometric data. For example, it can provide a tracking mechanism for identifying the origin of the biometric data (e.g., FBI). Also, searching for the correct decoded watermark information during authentication can render the modification of the data by a pirate useless, assuming that the watermark embedding-decoding system is secure. Furthermore, encryption can be applied to the watermarked data (but the converse operation, namely, applying watermarking to encrypted data is not logical as encryption destroys the signal characteristics such as redundancy, that are typically used during watermarking), combining the advantages of watermarking and encryption into a single system. In the context of this study, the security of the biometric data can be thought of as the means to eliminate at least some of the attacks cited in Chapter 2.

3.2 Generic Watermarking Systems

Despite the obvious advantages of digital environments for the creation, editing and distribution of multimedia data such as image, video, and audio, there exist important disadvantages: the possibility of unlimited and high-fidelity copying of digital content poses a big threat to media content producers and distributors. Watermarking, which can be defined as embedding information such as origin, destination, and access levels of multimedia data into the multimedia data itself, was proposed as a solution for the protection of intellectual property rights [19].

Handmade paper watermarking that started nearly 700 years ago inItaly was the first application of content protection. These traditional watermarks typically indicated the paper maker, the size of the paper, and the location of the mill that produced the paper. An example of such a watermark, along with the mold used to generate the watermark is given in Fig. 3.1 [21].

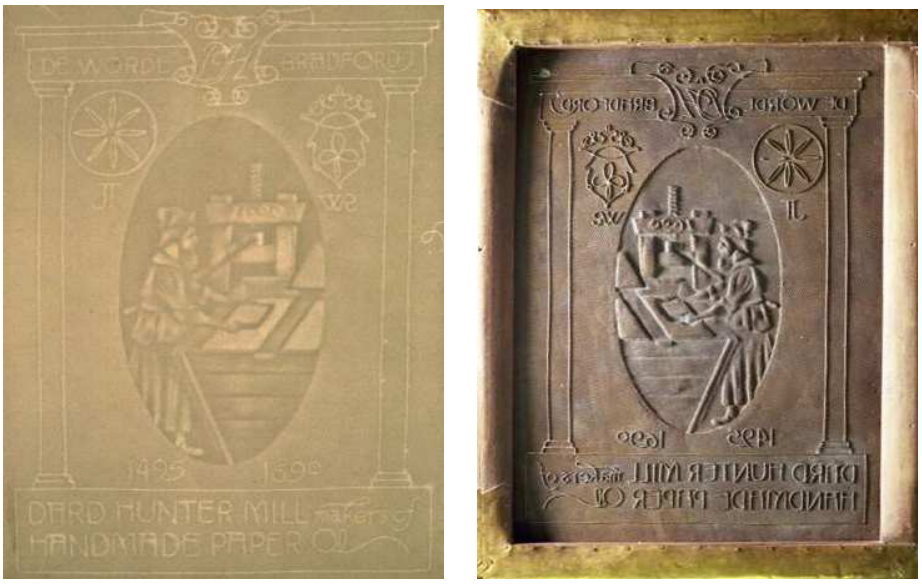


Figure 3.1: Paper watermarking: (a) watermark, (b) mold used to generate the watermark [21].

Even though the main principles date back hundreds of years, digital watermarking has received considerable attention in the last 15 years. Along with the development of watermarking techniques, initial application area of content protection was expanded to include authentication, data-monitoring, and transmission of value-added services.

The flow chart of a generic watermark encoding and decoding system is given in Fig. 3.2. In this system, the watermark signal (W) that is embedded into the host data (X) can be a function of watermark information (I) and a key (K) as in

$$W = f_0(I, K), \tag{3.1}$$

or it may also be related to host data as in

$$W = f_0(I, K, X). \tag{3.2}$$

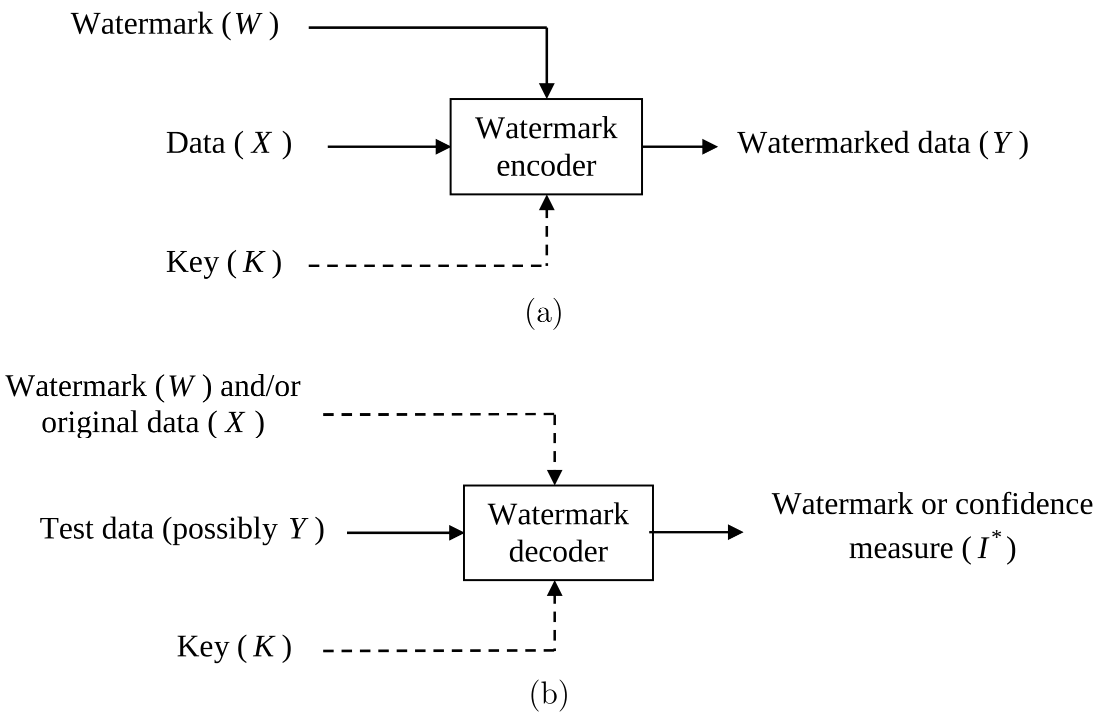


Figure 3.2: Digital watermarking block diagram: (a) watermark encoding, (b) watermark decoding.

The watermark information (I) is the information such as the legitimate owner of the data that needs to be embedded in the host data. The key is optional (hence shown as a dashed line in Fig. 3.2) and it can be utilized to increase the security of the entire system; e.g., it may be used to generate the locations of altered signal components, or the altered values. The watermark is embedded into host data to generate watermarked data Y

$$Y = f_1(X, W). \tag{3.3}$$

In watermark decoding, the embedded watermark information or some confidence measure indicating the probability that a given watermark is present in the test data (the data that is possibly watermarked) is generated using the original data as

$$I^* = g(X, Y, K), \tag{3.4}$$

or without using the original data as

$$I^* = g(Y, K). \tag{3.5}$$

Also, it may be desirable to recover the original, non-watermarked data, X, in some applications, such as reversible image watermarking. In those cases, an estimate $\hat{X}$ of the original data is also generated.

In watermark embedding, it is desired to keep the effects of watermark signal as imperceptible as possible in *invisible* watermarking applications: the end user should not experience a quality degradation in the signal (e.g., video) due to watermarking. For this purpose, some form of masking is generally utilized. For example, the frequency masking properties of the human auditory system (HAS) can be considered in designing audio watermark signals. Similarly, the masking effect of edges can be utilized in image watermarking systems. Conversely, in *visible* watermarking applications, it is not necessary to consider these systems as the actual aim of the application is to robustly mark the data, such as in embedding copyright data in terms of logos for images available over the Internet [65]. An example of visible image watermarking is given in Fig. 3.3 [37].

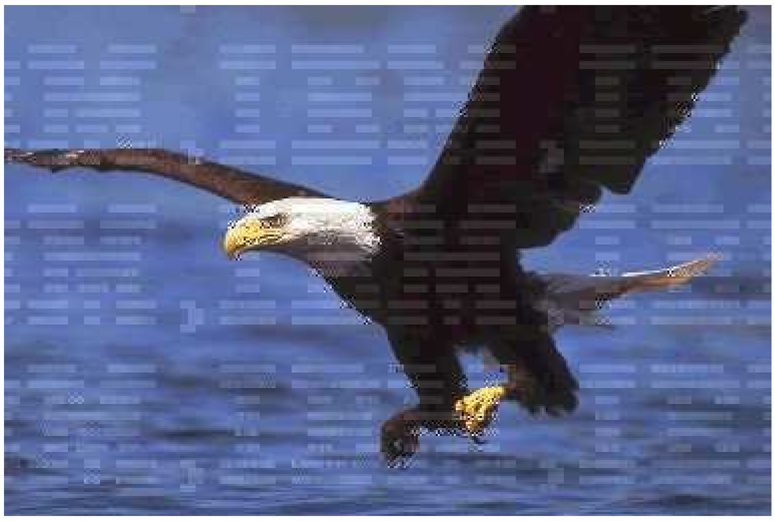


Figure 3.3: Visible image watermark [37].

Although there exist watermarking methods for almost all types of multimedia data, the number of image watermarking methods is much larger than the other types of media. In text document watermarking, generally the appearance of an entity in the document body is modified to carry watermark data. For example, the words in a sentence can be shifted slightly to the left or right, the sentences themselves can be shifted horizontally, or the features of individual characters can be modified (Fig. 3.4). Although text document watermarks can be defeated relatively easily by retyping or Optical Character Recognition (OCR) operations, the ultimate aim of making unauthorized copies of the document more expensive in terms of effort/time/money than obtaining the legal rights from copyright owner can still be achieved [19].

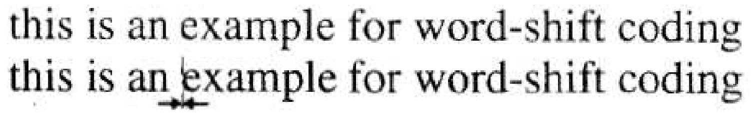

this is an example for word-shift coding
this is an example for word-shift coding

Figure 3.4: Text watermarking via word-shift coding [19].

Some authors claim that image watermarking methods can be applied to video, since a video can be regarded as a sequence of image frames. But the differences that reside in available signal space (much larger in video) and processing requirements (real time processing may be necessary for video) require methods specifically designed for video data [19]. Sample methods modify the motion vectors associated with specific frames or the labeling of frames to embed data. Audio watermarking techniques are generally based on principles taken from spread-spectrum communications. Modifying audio samples with a pseudo-randomly generated noise sequence is a typical example [19, 17].

In image watermarking, the watermark signal is either embedded into the spatial domain representation of the image, or one of many transform domain representations such as DCT, Fourier, and wavelet. It is generally argued that embedding watermarks in transform domains provides better robustness against attacks and leads to less perceptibility of an embedded watermark due to the spread of the watermark signal over many spatial frequencies and better modeling of the human visual system (HVS) when using transform coefficients. An example of watermarking in the spatial domain is given in Fig. 3.5(b) [28]. Amplitude modulation is applied to the blue channel pixels to embed the 32-bit watermark data, represented in decimal form as 1234567890. This is a *robust* watermarking scheme: the watermark data can be retrieved correctly even after the watermarked image is modified. For example, the embedded data 1234567890 is retrieved after the watermarked image is (i) blurred via filtering the image pixels with a 5x5 structuring element (Fig. 3.5(c)), and (ii) compressed (via JPEG algorithm with a quality factor of 75) and decompressed (Fig. 3.5(d)).

A specific class of watermarks, called *fragile* watermarks, are typically used for authenticating multimedia data. Unlike robust watermarks (e.g., the one given in Fig. 3.5), any attack on the image invalidates the fragile watermark present in the image and helps in detecting/identifying any tampering of the image. Hence, a fragile watermarking scheme may need to possess the following features: (i) detecting tampering with high probability, (ii) being perceptually transparent, (iii) not requiring the original image at decoding site, and (iv) locating and characterizing modifications to the image.

Typical fragile image watermarking techniques embed the watermark in the least significant bit planes of an image. To increase their security, several variants such as embedding image hash values have been proposed [29]. An example of fragile image watermarking is given in Fig. 3.6 [36]. The tampering of the watermarked image is

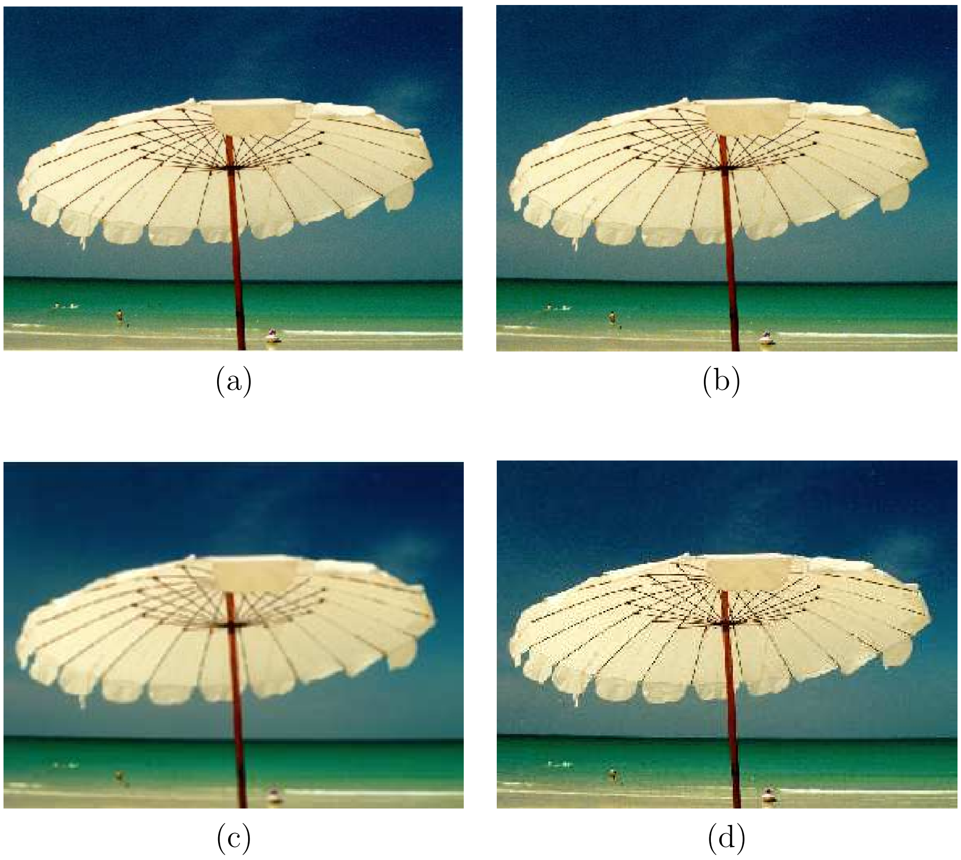


Figure 3.5: Image watermarking: (a) original image (640x480, 24 bpp), (b) watermarked image carrying the data 1234567890, (c) image blurred after watermarking, (d) image JPEG compressed-decompressed after watermarking [28].

identified via the change in the regular structure of the decoded watermark image in Fig. 3.6(d).

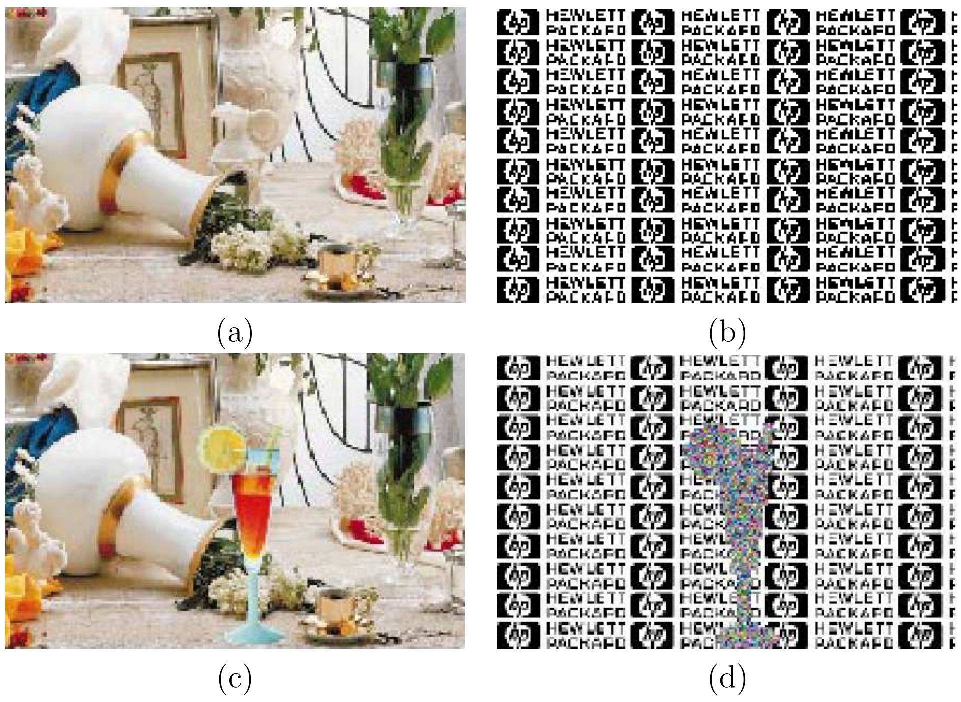


Figure 3.6: Fragile image watermarking: (a) watermarked image, (b) watermark image decoded from the image in (a), (c) altered image (cf. addition of the glass object), (d) watermark image decoded from the altered image in (c) [36].

3.3 Fingerprint Watermarking Systems

There have been only a few published papers on watermarking of fingerprint images. Ratha et al. [51] proposed a data hiding method, which is applicable to fingerprint images compressed with the WSQ (Wavelet Scalar Quantization) wavelet-based scheme. The discrete wavelet transform coefficients are changed during WSQ encoding, by taking into consideration possible image degradation. Fig. 3.7 shows an input fingerprint image and the image obtained after the data embedding-compressing-decompressing cycle. The input image was obtained using an optical sensor. The compression ratio was set to 10.7:1 and the embedded data (randomly generated bits) size was nearly 160-bytes. As seen from these images, the image quality does not suffer significantly due to data embedding, even though the data size is considerable.

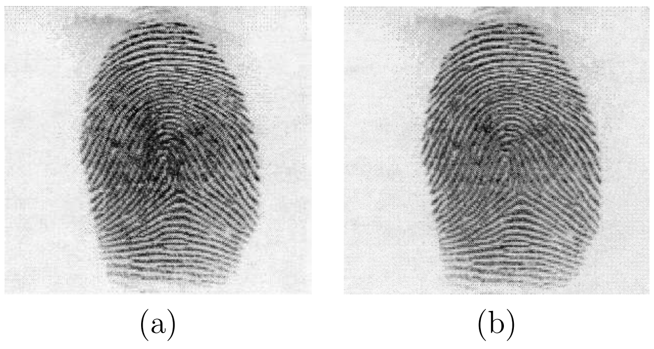


Figure 3.7: Compressed-domain fingerprint watermarking [51]: (a) input fingerprint, (b) data embedded-compressed-decompressed fingerprint.

Pankanti and Yeung [48] proposed a fragile watermarking method for fingerprint image verification. A spatial watermark image is embedded in the spatial domain of a fingerprint image by utilizing a verification key. Their method can localize any region of image that has been tampered after it is watermarked; therefore, it can be used to check integrity of the fingerprints. Fig. 3.8 shows a sample watermark image comprised of a company logo, and the watermarked image. Pankanti and Yeung [48] used a database comprised of 1,000 fingerprints (4 images each for 250 fingers). They calculated the Receiver Operating Characteristics (ROC) curves *before* and *after* the fingerprints were watermarked. These curves are observed to be very close to each other, indicating that proposed technique does not lead to a significant performance loss in fingerprint verification.

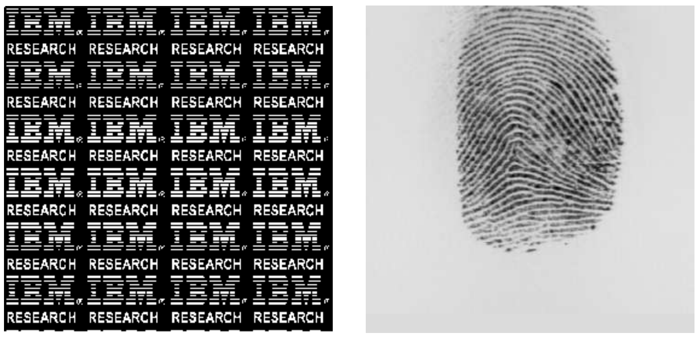


Figure 3.8: Fragile fingerprint watermarking [48]: (a) watermark image, (b) fingerprint image carrying the image in (a).

Gunsel et al. [18] described two spatial domain watermarking methods for fingerprint images (Fig. 3.9). The first method utilizes gradient orientation analysis in watermark embedding, so that the watermarking process alters none of the features extracted using gradient information. The second method preserves the singular points in the fingerprint image, which in turn preserves the classification of the watermarked fingerprint image (e.g., into arch, left loop classes). In both methods, watermark data consisted of 154-bits corresponding to the 7-bit ASCII code of string "Fingerprint_watermark.".

3.4 Architecture of the Proposed System

Two application scenarios are considered in this study. The basic data hiding method is the same in both scenarios, but it differs in the characteristics of the embedded data, the host image carrying that data, and the medium of data transfer. While fingerprint feature vector or face feature vector is used as embedded data, other information such

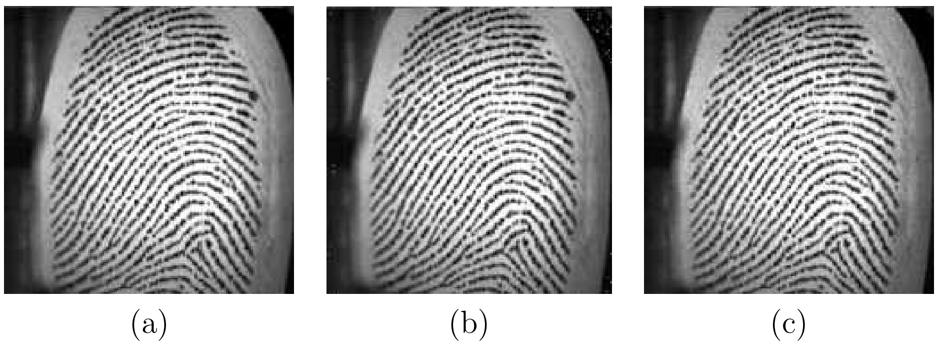


Figure 3.9: Fingerprint watermarking results for [18]: (a) input fingerprint, (b) fingerprint image watermarked using gradient orientation, (c) fingerprint image watermarked using singular points.

as user name (e.g., "John Doe"), user identification number ("12345"), or authorizing institution ("FBI") can also be hidden into the images.

The first scenario involves a steganography-based application (Fig. 3.10): the biometric data (fingerprint minutiae) that need to be transmitted (possibly via a non-secure communication channel) are hidden in a host (also called *cover* or *carrier*) image, whose only function is to carry the data. For example, the fingerprint minutiae may need to be transmitted from a law enforcement agency to a template database, or vice versa. In this scenario, the security of the system is based on the secrecy of the communication. The host image is not related to the hidden data in any way. As a result, the host image can be any image available to the encoder. In our application, we consider three different types of cover images: a synthetic fingerprint image, a face image, and an arbitrary image (Fig. 3.11). The synthetic fingerprint image (360x280) is obtained after a post-processing of the image generated using the algorithm described by Cappelli et al. [5]. Using such a synthetic fingerprint image to carry actual fingerprint minutiae data provides an increased level of security since the person who intercepts the communication channel and obtains the carrier image is likely to treat this synthetic image itself as a real fingerprint image, and not consider that it is in fact carrying the critical data. The face image (384x256) was captured in our laboratory. The "Sailboat" image (512x512) is taken from the USC-SIPI database [70].

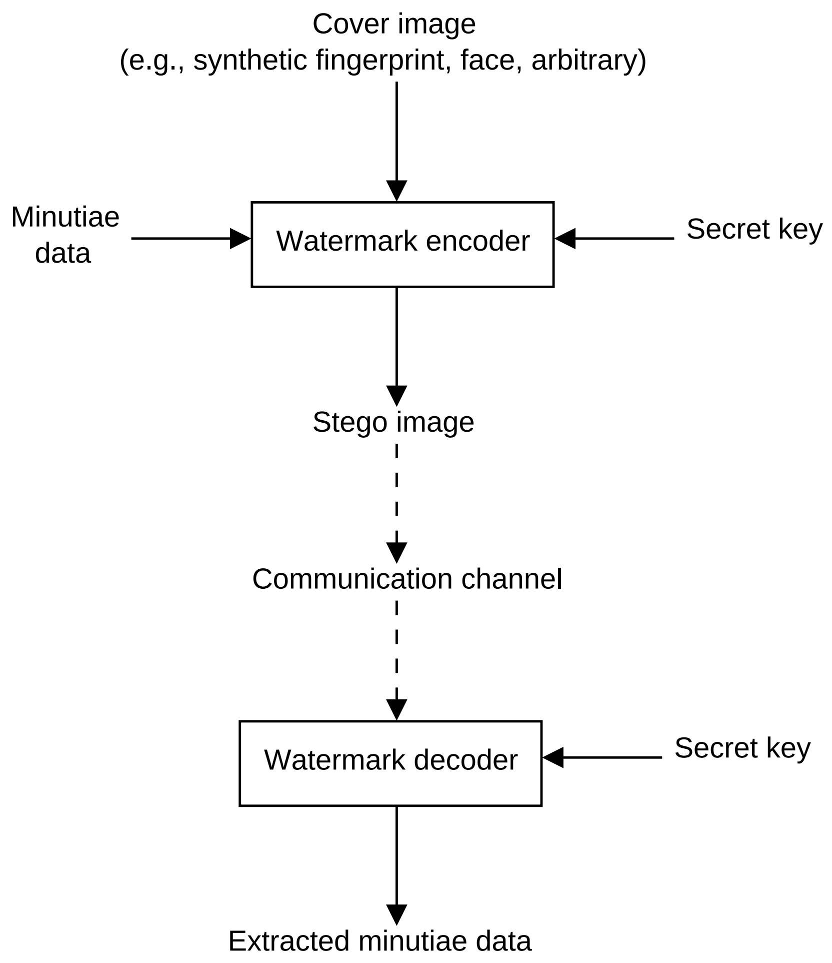


Figure 3.10: Block diagram of application scenario 1.

This application can be used to counter the seventh type of attack (namely, compromising the communication channel between the database and the fingerprint matcher) depicted in Fig. 2.1. An attacker will probably not suspect that a cover

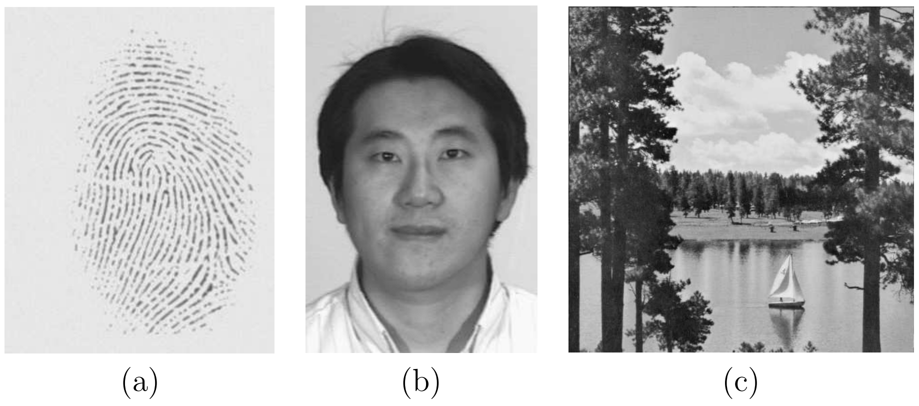


Figure 3.11: Sample cover images: (a) synthetic fingerprint, (b) face, (c) "Sailboat".

image is carrying the minutiae information. Furthermore, the security of the transmission can be further increased by encrypting the host image before transmission. Here, symmetric or asymmetric key encryption [56] can be utilized, depending on the requirements of the application such as key management, coding-decoding time (much higher with asymmetric key cryptography), etc. The position and orientation attributes of fingerprint minutiae constitute the data to be hidden in the host image.

The fingerprint images (300x300) used here were captured by a solid-state sensor manufactured by Veridicom. The minutiae are extracted using the method outlined in [23]. A secret key is utilized for encoding to increase the security of the hidden data. The image with embedded data (called *stego* image) is sent through a channel that may be subject to interceptions. At the decoding site, using the same key that was used by the encoder (which can be delivered to the decoder using a secure channel prior to stego image transfer), the hidden data is recovered from the stego image. The keys can be different for every transmission, or several parameters such as receiver, sender, and fingerprinted subject identities can be used in tuning the key assignment.

The second scenario is based on hiding facial information (e.g., eigen-face coefficients) into fingerprint images (Fig. 3.12). In this scenario, the marked fingerprint image of a person can be stored in a smart card issued to that person. At an access control site, for example, the fingerprint of the person possessing the card will be sensed and it will be compared to the fingerprint stored on the smart card. Along with the fingerprint matching, the proposed scheme will also extract the face information hidden in the fingerprint image. The recovered face will be used as a second source of authentication either automatically or by a human in a supervised biometric application. In this scenario, one biometric (e.g., face) is embedded into another (e.g., fingerprint), in order to increase the security of the latter.

3.4.1 Data Hiding Method

The amplitude modulation-based watermarking method described here is an extension of the blue channel watermarking method of Kutter et al. [28]. The proposed method includes image adaptivity, watermark strength controller, and host image feature analysis along with the basic method in [28]. An earlier version of the method was presented in [18], in which the increase in data decoding accuracy related to these extensions was analyzed.

In the first step, the data to be hidden in the host image is converted to a binary stream. In the first scenario, where fingerprint minutiae data are hidden, every field of individual minutia is converted to a 9-bit binary representation. Such a representation can code integers between [0, 511]. This range is adequate for x-coordinate ([0, N-1]),

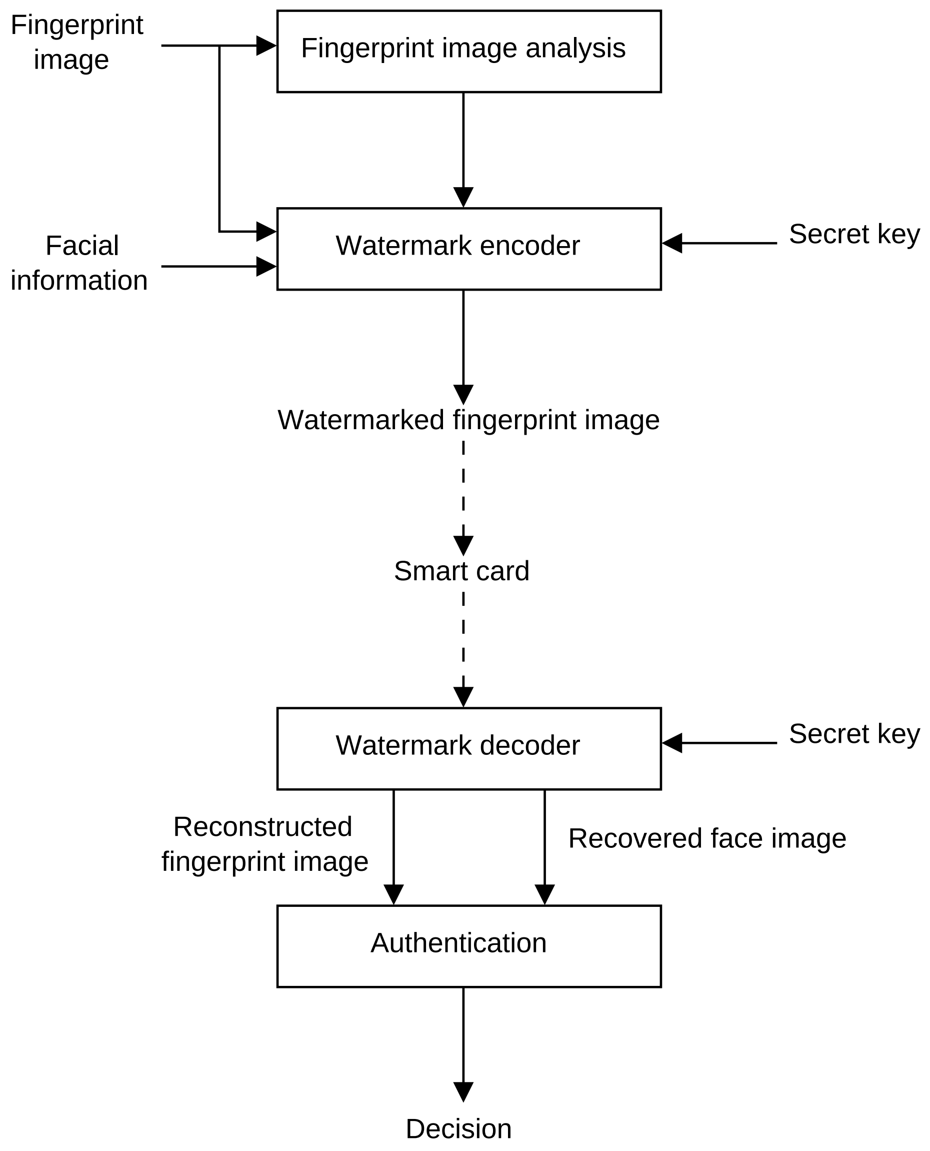


Figure 3.12: Block diagram of application scenario 2.

y-coordinate ([0, M-1]), and orientation ([0, 359]) of a minutia, where N and M are the number of rows and columns in the fingerprint image, respectively. In the second scenario, eigen-face coefficients are converted into a binary stream using 4-bytes per coefficient. A random number generator initialized with the secret key generates locations of the host image pixels to be watermarked. The details of this procedure are as follows: first, a sequence of random numbers between 0 and 1 is generated using uniform distribution. Then, every number with odd indices is linearly mapped to $[0, X-1]$, and every number with even indices is linearly mapped to $[0, Y-1]$, where X and Y are the number of rows and columns of the host image, respectively. Every pair comprised of one number with odd indices and one number with even indices indicates the location of a candidate pixel to be marked. During watermark embedding, a pixel is not changed more than once, as this can lead to incorrect bit decoding. Also, the pixels where $\beta(i,j)$ (marked pixel map, explained below) is zero are not marked. If at any step in embedding, the candidate pixel can not be marked due to one of these reasons, the next pixel location is considered.

The (i,j)th pixel is changed according to the following equation:

$$\begin{aligned} P_{WM}(i,j) = P(i,j) + (2s-1)P_{AV}(i,j)q \\ * (1+\frac{P_{SD}(i,j)}{A})(1+\frac{P_{GM}(i,j)}{B})\beta(i,j), \end{aligned} \quad (3.6)$$

where $P_{WM}(i,j)$ and $P(i,j)$ are values of the watermarked and original pixels at location (i,j), respectively. The value of the watermark bit is denoted as s, and the watermark embedding strength is denoted as q, where $s \in [0,1]$, and $q > 0$. $P_{AV}(i,j)$ and $P_{SD}(i,j)$ denote the average and standard deviation of pixel values in the neighborhood of pixel (i,j), and $P_{GM}(i,j)$ denotes the gradient magnitude. The parameters A and B are weights for the standard deviation and the gradient magnitude, modulating the effect of these two terms: increasing either of them decreases the overall modulation effect on the amount of change in pixel intensity, while decreasing them has the opposite effect. The minimum values for $P_{SD}(i,j)$ and $P_{GM}(i,j)$ are both 0 (for neighborhoods with constant gray level), and the maximum value for $P_{SD}(i,j)$ is around 127 (for a checkerboard pixel pattern composed of just 0 and 255 gray levels). The maximum value for $P_{GM}(i,j)$ is around 1,082 for a maximum magnitude diagonal edge (e.g., intersection of gray levels 0 and 255). The $\beta(i,j)$ term guarantees that the so-called marked pixels, whose alteration may affect the performance of an algorithm using the watermarked image (e.g., fingerprint verification in the case of watermarked fingerprint images) remain unchanged: $\beta(i,j)$ takes the value 0 if the pixel (i,j) is a marked pixel, and 1, otherwise. These three parameters (P_{SD}, P_{GM}, and β) modulate the amount of change in pixel values made due to marking, and they represent a significant modification of the basic marking method given in [28].

In the second scenario, the marked pixels are defined by either minutiae analysis or ridge analysis of the fingerprint image. In our experiments, P_{AV} is calculated in a 5x5 square neighborhood, and P_{SD} is calculated in a 5x5 cross-shaped neighborhood. The gradient magnitude is computed via the 3x3 Sobel operator.

The image adaptivity terms discussed above adjust the magnitude of watermarking by utilizing several properties of the human visual system (HVS). Using $P_{AV}(i,j)$ in modulating the watermark magnitude conforms to the amplitude nonlinearity of the HVS. As Eq. (3.6) shows, the magnitude of the change in the value of pixel (i,j) caused by watermarking is higher when $P_{AV}(i,j)$ is high. The standard deviation and gradient magnitude terms utilize the contrast/texture masking properties of

HVS. These image adaptivity terms increase the magnitude of watermarking in image areas where such an increase does not become very visible to a human observer.

Every watermark bit with value s in Eq. (3.6) is embedded at multiple locations in the host image. This redundancy increases the decoding accuracy of the embedded information (hence the system is *not* fragile). The amount of this redundancy is limited by image capacity (size) and visibility of the changes in pixel values. Furthermore, the β mask preserves critical features of the host fingerprint image. In addition to the binary minutiae data, two reference bits, 0 and 1, are also embedded in the host image. These reference bits help in calculating an adaptive threshold in determining the minutiae bit values during decoding.

Decoding starts with finding the data embedding locations in the watermarked image, via the secret key used during the watermark encoding stage. Note that the original, non-watermarked image is not used in decoding, just the watermarked image. For every bit embedding location (i, j), its value during decoding is estimated as the linear combination of pixel values in a 5x5 cross-shaped neighborhood of the watermarked pixels as shown in Eq. (3.7).

$$\hat{P}(i,j) = \frac{1}{8}(\sum_{k=-2}^{2} P_{WM}(i+k,j) + \sum_{k=-2}^{2} P_{WM}(i,j+k) - 2P_{WM}(i,j)). \qquad (3.7)$$

The difference between the estimated and the watermarked pixel values is calculated as

$$\delta = P_{WM}(i,j) - \hat{P}(i,j). \tag{3.8}$$

These differences are averaged over all the embedding locations associated with the same bit, to yield $\bar{\delta}$. For finding an adaptive threshold, these averages are calculated separately for the reference bits, 0 and 1, as $\bar{\delta}_{R0}$ and $\bar{\delta}_{R1}$, respectively. Finally, the watermark bit value $\hat{s}$ is estimated as

$$\hat{s} = \begin{cases} 1 & \text{if } \bar{\delta} > \frac{\bar{\delta}_{R0} + \bar{\delta}_{R1}}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

Eq. (3.9) essentially indicates that if $\bar{\delta}$ is closer to $\bar{\delta}_{R1}$, that bit is declared as '1'; if it is closer to $\bar{\delta}_{R0}$, that bit is declared as '0'.

The watermark decoding process can produce erroneous bits since decoding is based on an estimation procedure, which may fail to find the exact original pixel values. This may lead to switched bits in decoding. However, in the context of this work, it is imperative that every one of the embedded bits is decoded correctly (i.e., 0% error rate). Since critical information is embedded, even one bit change can decrease the usability of the data (e.g., minutiae data change, eigen-face coefficient change due to switched bits). In order to increase the decoding accuracy, the encoder uses a controller block. This block adjusts the strength of watermarking, q, on a pixel-by-pixel basis, if there is a possibility of incorrect bit decoding. Effectively, given the parameters such as A, B, and q, the encoder checks whether the decoding will be correct or not. In the former case, the controller moves on to analyze the next bit embedding location; in the latter case, q is increased to the point where the bit can be correctly decoded.

From the decoded watermark bits, the payload data hidden in the host image (minutiae data or eigen-face coefficients) are extracted. Using the recovered eigen-face coefficients and the eigen-faces stored in the watermark decoding site, the hidden face image is reconstructed. In the second application scenario, an estimate of the original host fingerprint image is also found via replacing the watermarked pixel values with the $\hat{P}(i,j)$ calculated in Eq. (3.7).

3.5 Experiments and Results

In this section, experimental results will be presented for the two application scenarios explained in Section 3.4. Factors such as decoding accuracy and matching performance will be highlighted. For the first scenario, nearly 17% of the stego image pixels are changed during minutiae data hiding for all the three cover images shown in Fig. 3.11. The key used in generating the locations of the pixels to be watermarked is selected as the integer 1,000. However, the exact value of the key does not affect the performance of the method. This key is used as the seed for the C++ random number generator. The generated random numbers are used as explained in Section 3.4.1. Other random number generators can also be used without affecting the performance of the proposed method. Remaining watermarking parameters are set to: $q = 0.1$, $A = 100$, $B = 1,000$. A higher q value increases the visibility of the hidden data. Increasing A or B decreases the effect of standard deviation and gradient magnitude in modulating watermark embedding strength, respectively. The amount of hidden data here is approximately 85 bytes. The extracted minutiae data from all the three cover images is found to be exactly the same as the hidden data. Furthermore, the performance of the proposed algorithm was determined as follows: 15 images (5 synthetic fingerprint, 5 face, and 5 arbitrary) were watermarked with 5 different sets of minutiae data, and by using 5 different keys. As a result, 375 different watermarked images were produced. Characteristics and sources of the host images and watermarking parameters were the same as described previously. Individual minutiae data sets contained between 23 to 28 points, with an average value of 25. From all of these 375 watermarked images, the algorithm was able to extract the embedded minutiae information with 100% accuracy.

For the second application scenario, the fingerprint image (300x300 pixels) shown in Fig. 3.13(a) is watermarked using the input face image (150x130) shown in Fig. 3.13(b). The watermark information occupies 56 bytes, corresponding to the 14 eigen-face coefficients (4 bytes per coefficient). These 14 eigen-face coefficients generate the 150x130 watermark face image of Fig. 3.13(c) [7]. Note that 14 eigen-face coefficients are sufficient for a reasonable fidelity reconstruction of the input face. A small face image database, consisting of 40 images (four images for each of the 10 subjects), was used to generate the eigen-faces and coefficients.

Figures 3.13 (d)-(e) correspond to minutiae-based data hiding. The input image in Fig. 3.13(a) is watermarked without changing the pixels shown in black (16% of the total image pixels) in Fig. 3.13(d). This minutiae-based feature image, which represents the $\beta(i,j)$ term in Eq. 3.6, is obtained by drawing 23x23 square blocks

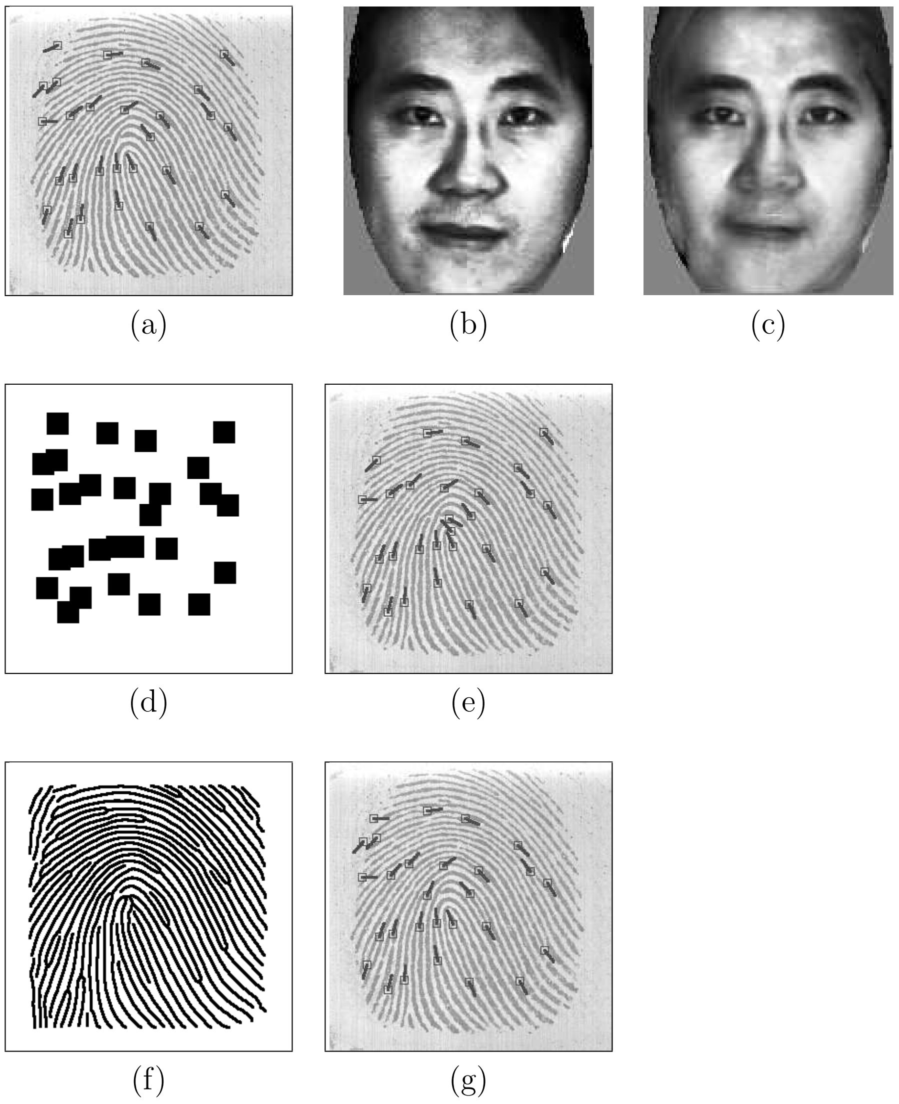


Figure 3.13: Facial information embedding and decoding: (a) input fingerprint image with overlaid minutiae, (b) input face image, (c) watermark face image, (d) fingerprint feature image based on the minutiae, (e) reconstructed fingerprint image with overlaid minutiae, where watermarking did not change the pixels shown in black in (d), (f) fingerprint feature image based on the ridges, (g) reconstructed fingerprint image with overlaid minutiae, where watermarking did not change the pixels shown in black in (f).

around every minutia of the input fingerprint image. Fig. 3.13(e) shows the image reconstructed during watermark decoding. Nearly 15% of the total number of image pixels are modified during watermark encoding. This marking ratio is determined experimentally, by requiring 100% correct decoding of the embedded data. Figures 3.13 (f)-(g) correspond to ridge-based data hiding: The input image in Fig. 3.13(a) is watermarked without changing the pixels shown in black (31% of the total number of image pixels) in Fig. 3.13(f). This ridge-based feature image is obtained from the thinned ridge image of the input fingerprint via dilation with a 3x3 square structuring element comprised of all 9 pixels. Fig. 3.13(g) shows the image reconstructed during watermark decoding. Nearly 15% of all image pixels are modified during watermark encoding. This embedding ratio is the same as the one used for minutiae-based embedding; fixing this parameter allows comparing the two methods based on their $\beta(i,j)$ mask characteristics alone, and not based on modified pixel ratio. Effectively, the images in Figures 3.13(d) and 3.13(f) denote the binary $\beta(i,j)$ maps.

In both of these cases, the key used in generating the locations of the pixels to be watermarked is selected as the integer 1,000. However, as mentioned earlier, the exact value of the key does not affect the performance of the method. Other watermarking parameters are set to the same values used previously, namely: $q = 0.1$, $A = 100$, $B = 1,000$. The watermark data are decoded correctly in the decoding phase in both of the cases; the recovered faces are exactly the same as the watermark face image in Fig. 3.13(c).

In order to assess the effect of watermarking on fingerprint verification accuracy, ROC curves for the original images and images that are recovered after watermark decoding are computed. A total of 640 fingerprint images are used in the experiments. These images come from 160 users, with 4 impressions each of the right index finger captured using a Veridicom sensor. Three ROC curves given in Fig. 3.14 correspond to fingerprint verification (i) without data hiding, (ii) with minutiae-based data hiding, and (iii) with ridge-based data hiding.

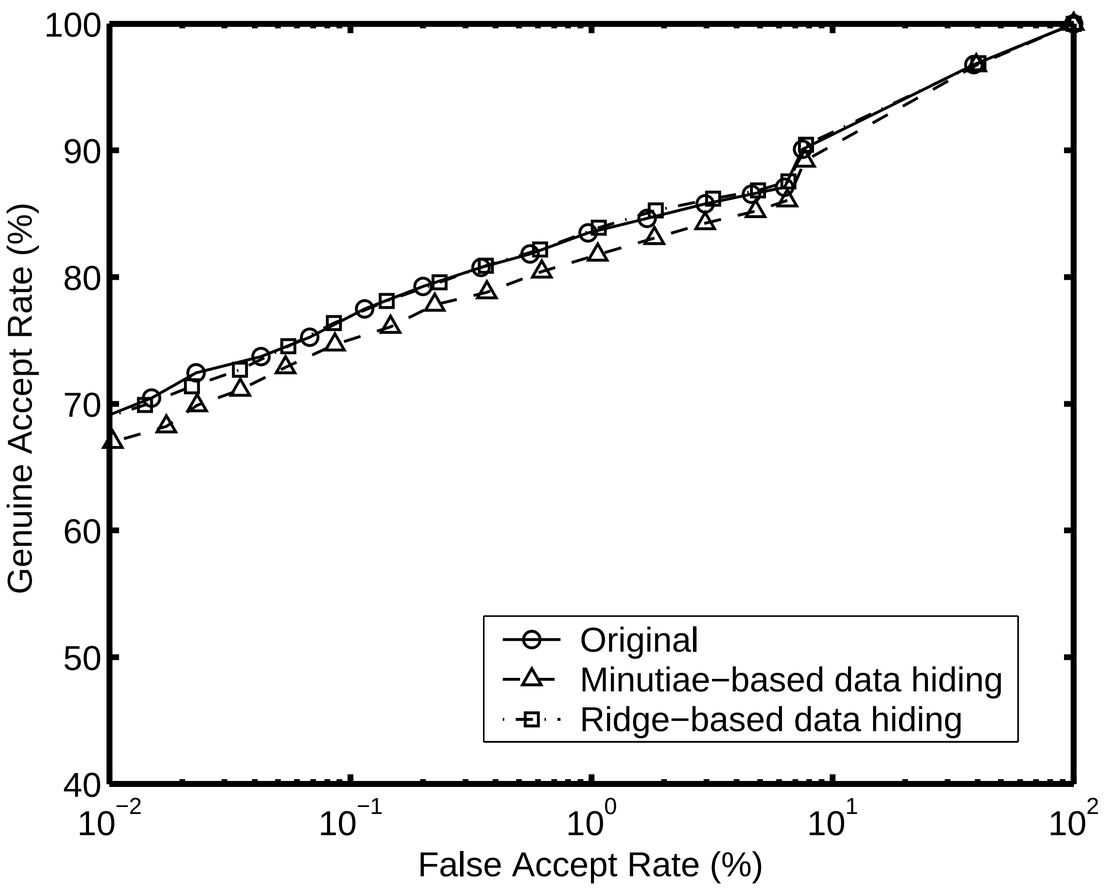


Figure 3.14: ROC curves corresponding to original and eigen-face carrying fingerprint images.

The proximity of the three curves in Fig. 3.14 indicates that both the minutiae-based and the ridge-based watermarking methods do not introduce any significant degradation in fingerprint verification accuracy, though it is observed that ridge-based watermarking leads to less degradation. Furthermore, in both of these cases, the embedded information (i.e., 14 eigen-face coefficients) was decoded with 100% accuracy from all of the 640 watermarked images.

3.6 Summary

The ability of biometrics-based personal identification techniques to differentiate between an authorized person and an impostor who fraudulently acquires the access privilege of an authorized person is one of the main reasons for their popularity compared to traditional identification techniques. However, the security and integrity of the biometric data itself raise important issues, which can be ameliorated using encryption, watermarking, or steganography.

In this chapter, two applications of watermarking to secure biometrics data were presented. In addition to watermarking, encryption can also be used in order to further increase the security of biometrics data. Our first application was related to increasing the security of biometric data exchange, which was based on steganography. In the second application, facial information was embedded within fingerprint images. In this application, the data was hidden in such a way that the features that were used in fingerprint matching were not significantly changed during encoding and decoding. As a consequence, the verification accuracy based on decoded watermarked images was very similar to that with original images. For applications where the watermarked images need to be processed by humans (e.g., manual fingerprint matching), the proposed method utilizes several properties of the human visual system to keep the visibility of the changes made to the host images low. Further, the proposed system can be coupled with a fragile watermarking scheme to detect illegitimate modification of the watermarked templates.

Chapter 4

Biometrics-Based Encryption

4.1 Introduction

The use of digital techniques in the creation, editing and distribution of multimedia data (e.g., image, video, and audio) offers various opportunities to a pirate user, such as high-fidelity and rapid duplication: the generated copies are exactly the same as the original data, and copying is very fast. Contrariwise, analog techniques (e.g., printing and scanning an image) are relatively time consuming and they lead to quality degradation (e.g., due to printer noise). As a result of another advantage of digital techniques, the widespread usage of Internet provides additional channels for a pirate to quickly and easily distribute copyrighted digital content to a large number of users without the fear of being tracked. Hence, multimedia data owners and their legal distributors are raising concerns about the loss of considerable amounts of revenues and its adverse effects on the creation of novel material and its timely distribution. As a result, the protection of multimedia content is now receiving a substantial amount of attention from the academia, multimedia industry, and regulatory government agencies [69].

4.2 Techniques to Protect Multimedia Data

Two of the most commonly used methods for the protection of intellectual property rights (IPR) of multimedia data are digital watermarking and cryptography:

4.2.1 Digital Watermarking

Digital watermarking (see Chapter 3 for details) consists of embedding some information about the data (e.g., ownership, legitimate user identity, access rights) into the multimedia data itself, typically by the copyright owner or the data distributor. Currently, there is no watermarking technique that is robust to all possible attacks that can be mounted against it by the pirates (e.g., filtering, cropping, format change that results in the erasure of, duplication of, or ambiguity about embedded watermarks). As a result, this solution cannot eliminate the cited problems of piracy completely.

4.2.2 Cryptography

In traditional cryptographic systems, one or more keys are used to convert the plain text (i.e. data to be encrypted: audio files) to cipher text (i.e. encrypted data: encrypted audio files). The encrypting key(s) maps the plain text to essentially a sequence of pseudo random bits (modern crypto algorithms are designed with this criteria), that can only be mapped back to the plain text using the appropriate decrypting key(s). Without the knowledge of the correct decrypting keys, the conversion of cipher text to the plain text is infeasible considering time and cost limitations [56, 64]. Hence, the cipher text is secured: even if an attacker obtains the cipher text, she cannot extract useful information (i.e. plain text) from it. Here, the plain text can be any data that needs to be stored or transmitted securely: financial transactions, e-mail communication, health records, fingerprint images, secret cryptographic keys, etc.

Fig. 4.1 shows block diagrams of symmetric and asymmetric key cryptographic systems, in the realm of two entities that want to communicate securely (denoted as Alice and Bob). In the symmetric system, the decrypting key is the same as the encrypting key (namely, Alice and Bob share the key K_{AB}). Whereas in the asymmetric system, the decrypting key is not the same as the encrypting key, and it is only known to the recipient of the message: Alice can access Bob's public key (encrypting key) K_B^+, but only Bob knows his private key (decrypting key) K_B^-.

Current cryptographic algorithms (e.g., symmetric key systems Advanced Encryption Standard (AES) [44], Data Encryption Standard (DES) [64], or asymmetric key system RSA [64]) have high theoretical and proven security. That is, there are no publicly known *feasible* procedures to invert the associated cipher text back to the plain text, given the computational resources (processor speed, processor quantity, and storage capacity) available to attackers today. As a result, encryption of multimedia data (by the copyright owner or the data distributor) can be utilized to eliminate the problems of unauthorized copying and distribution: the data will be useless without the knowledge of the correct encrypting and decrypting keys. But this solution

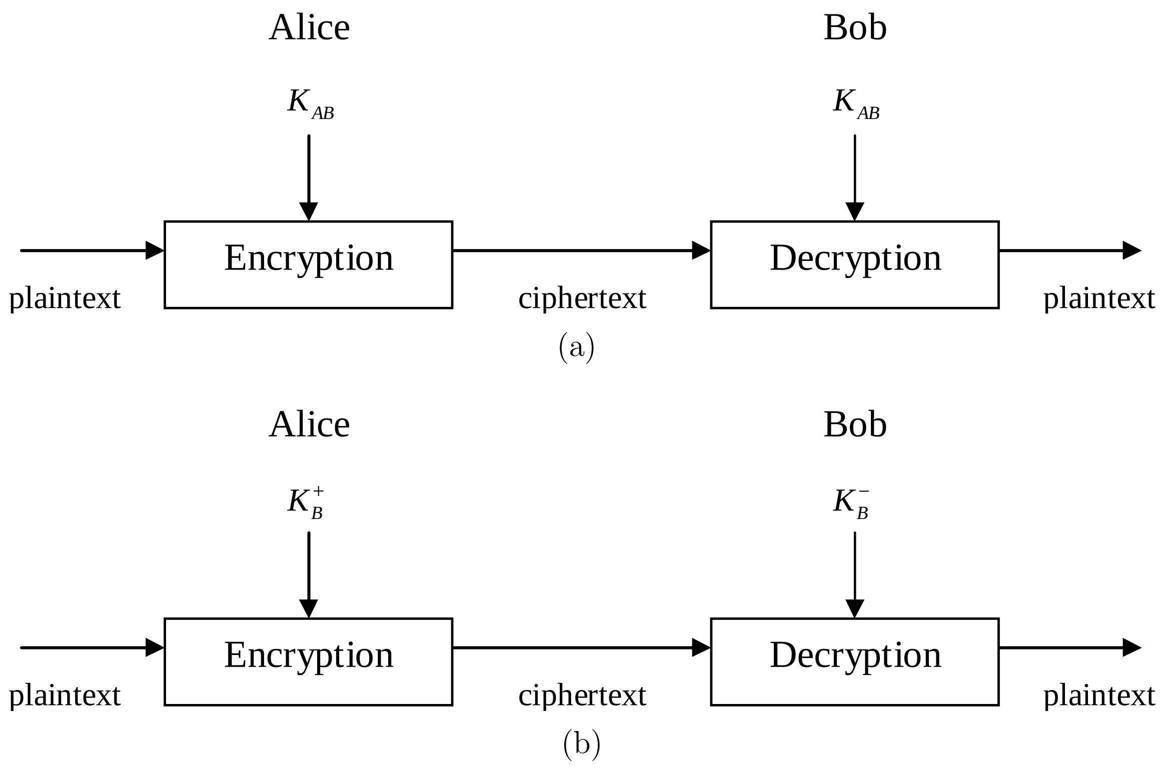


Figure 4.1: Traditional cryptography: (a) symmetric key system, (b) asymmetric key system.

also has problems, as explained below.

Limitations of Cryptography for Multimedia Data Protection

Suppose Alice has an encrypted multimedia file, and assume that a pirate web site or a pirate user, Bob, is distributing this file. In order for Alice to use the file, she must also have the correct key(s) to decrypt the data. Alice can obtain the key(s) via legal means, e.g., receiving them after registering herself at the legitimate web site associated with the content and supplying her payment information (e.g., credit card number). This way the content provider has the information about the user (Alice) who is about to view/play/listen to (henceforth, this utilization of multimedia data will be referred collectively as *playing*) the protected content, and it has the means to charge Alice for this privilege. However, Alice can also obtain the key via pirated means (e.g., Bob sends Alice the correct key, in addition to the encrypted file), which eliminates the security provided by the encryption instantly. This illegal sharing of keys (i.e. key management problem) is a big drawback of any content protection scheme based on traditional cryptography.

How to Improve Traditional Cryptographic Algorithms?

An additional source of information that can be introduced to the encryption process is related to the attributes of the physical system (hardware or software) utilized by the consumers of multimedia data. For example, the hard disk (HD) serial number, the operating system number, computer IP address, etc. can be used as keys (or generators for keys) in the encryption process. In this scenario, the decoder checks these entities in a host computer and, if they are not the ones used during encryption, the data cannot be decoded correctly (hence, if Alice is not using a computer that has exactly the same credentials as those of Bob's, she will not be able to play the data). Here, it is assumed that the hardware identifiers cannot be tampered with, for example, not only can Bob not easily send his hard drive to Alice, but also Alice cannot tamper with her own hard drive serial number to imitate Bob's hardware credentials. But a legitimate user may want to play the multimedia file in multiple systems, such as a notebook, desktop computer, or PDA. Using hardware identifiers in the encryption/decryption processes eliminates such a possibility.

Another possible solution to illegal sharing of keys is to use biometric characteristics of the users. Assuming that the biometric system is secure (information about the feasibility of possible attacks can be found in Chapter 2), introducing the biometric data of the user into the encryption/decryption processes (as keys or key generators) can increase the security of the digital content and decrease the feasibility of its illegal utilization. This would be equivalent to substituting the keys (e.g., K_{AB} in Fig. 4.1) with biometric data (e.g., fingerprint features). For example, in the scenario mentioned earlier where Bob sent Alice a pirated file, now Alice would need to present Bob's finger (if fingerprint was used as the biometric identifier) to correctly decode the pirated data. The feasibility (with respect to time, cost, and required knowledge) of Alice replicating Bob's fingerprint features at her site is considerably less than obtaining traditional encryption keys from Bob. Hence, this solution has the potential to alleviate the piracy of copyrighted multimedia data, if yet another problem (variability of biometric data as explained below) can be solved.

Biometric Variability & Cryptography

There is a big challenge to be overcome in using biometric signals as keys in encryption/decryption processes: biometric signals are *not* invariant over time. That is, even legitimate users of the system may not be able to decrypt the files that were encrypted with a previous acquisition of their biometric characteristics. For example, in the case of fingerprints, multiple impressions of a finger change because of improper placement of the finger on the sensor, sensor noise, dry or dirty fingers and cuts and bruises on them (Fig. 4.2).

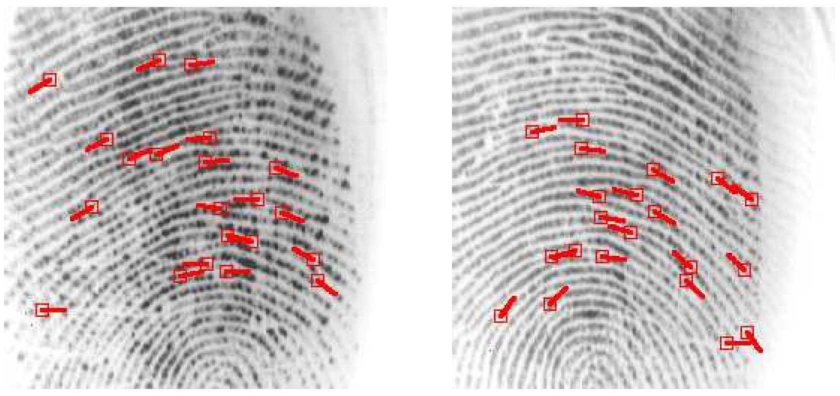


Figure 4.2: Intra-class variability of biometric signal: two different images of the same finger, with overlaid minutiae.

Intra-class variations in the biometric signals lead to non-invariance of features. For example, Fig. 4.2 shows the extracted minutiae features overlaid on the fingerprint images. As a result, biometric data cannot be used directly to define a key in a digital signature system. Note that, although these changes in biometric data are "small", the cryptographic system becomes useless if the intra-class variability results in even a 1-bit change in the generated key. Traditional cryptographic systems work only if the key used during encryption is identical to the key used in decryption. Note that, in spite of these intra-class changes, the biometric matcher will normally generate a "higher" similarity score between two impressions of Bob's fingerprints, compared to the case when the input pair consists of one fingerprint from Bob and one from Alice.

4.3 Proposed System

We propose a multimedia content protection scheme based on biometric matching, with the aim of eliminating the feasibility of the illegal key sharing problem outlined above. Further, to achieve robustness against biometric variance, we use a novel biometric data transfer protocol, as explained below.

Assume that there exist two communicating entities; the server S and the user U. The user U wants to receive the file V that resides at S. During data transfer, both asymmetric and symmetric key encryption schemes are used (see Section 4.2 above). In Fig. 4.3, K_S^+ and K_S^- denote the public and private keys of S, respectively. $K_S^+(.)$ and $K_S^-(.)$ denote the application of these keys in an asymmetric key system. In the symmetric key system, $E(X, k_1, k_2, \ldots, k_n)$ denotes encrypting the file X first with key k_1, then encrypting the resulting file with k_2, and so on. Similarly, $D(Y, k_n, k_{n-1}, \ldots, k_1)$ denotes decrypting the file Y, first with key k_n, then decrypting the resulting file with key k_{n-1}, and so on.

The data initially available at S and U are shown in respective columns inside dashed boxes in Fig. 4.3. I_U is the identity of the user U, such as the user name, P_U is the password selected by the user to be used in encryption/decryption steps, and

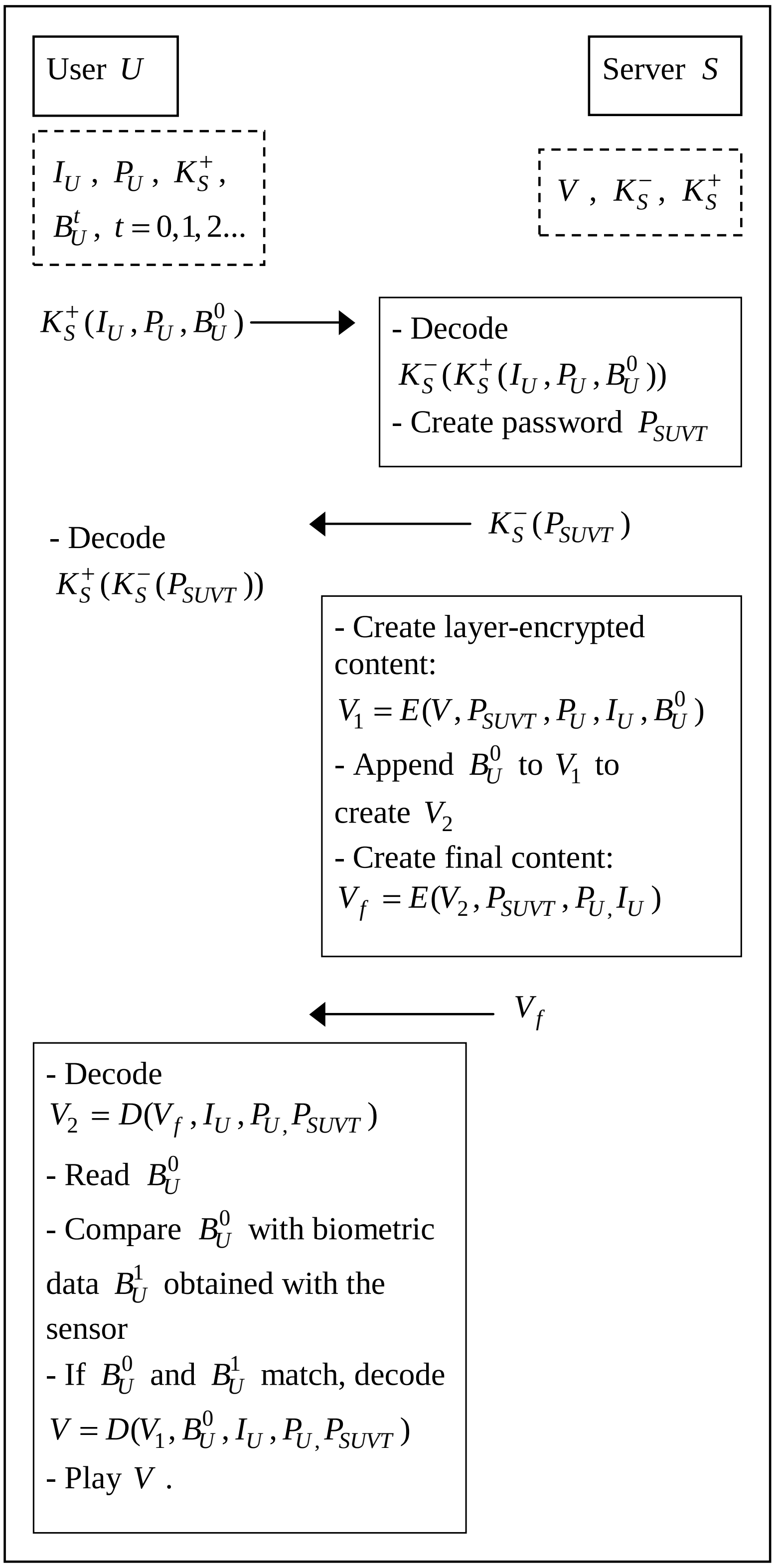

User U
Server S
I_U, P_U, K_S^+, B_U^t, $t = 0,1,2...$
V, K_S^-, K_S^+
$K_S^+(I_U, P_U, B_U^0)$
- Decode $K_S^-(K_S^+(I_U, P_U, B_U^0))$
- Create password P_{SUVT}
$K_S^-(P_{SUVT})$
- Decode $K_S^+(K_S^-(P_{SUVT}))$
- Create layer-encrypted content: $V_1 = E(V, P_{SUVT}, P_U, I_U, B_U^0)$
- Append B_U^0 to V_1 to create V_2
- Create final content: $V_f = E(V_2, P_{SUVT}, P_U, I_U)$
V_f
- Decode $V_2 = D(V_f, I_U, P_U, P_{SUVT})$
- Read B_U^0
- Compare B_U^0 with biometric data B_U^1 obtained with the sensor
- If B_U^0 and B_U^1 match, decode $V = D(V_1, B_U^0, I_U, P_U, P_{SUVT})$
- Play V.

Figure 4.3: Data transfer structure.

B_U^t $(t = 0, 1, 2, \ldots)$ denotes the biometric data (either as raw data, or feature vectors extracted from the fingerprint images as used in this study) of the user U, obtained at time t. Note that the biometric data is not invariant with respect to time (i.e. generally, $B_U^t \neq B_U^{t'}$, if $t \neq t'$) due to the reasons cited before.

Now, we describe the biometric-based encryption and decryption processes in detail. First, using the public key of the server, the user encrypts I_U, P_U and B_U^0, and sends this encrypted data to the server. The server decodes these three pieces of information by using its private key K_S^-. Due to this asymmetric key scheme (e.g., RSA [56]), only the server S, and not an intruder, can decode this information. After checking the validity of the user credentials and related issues such as payment status (e.g., charging a credit card), the server creates a password P_{SUVT} (which is a function of the server S, user U, content V and a time stamp T, indicating the transaction date and time) and sends it to the user after encrypting it with K_S^-. The user decodes this data by using the public key K_S^+.

The server creates the encrypted content V_1 by encrypting (symmetrically, using e.g., DES [56]) V in a layered manner with the keys generated from P_{SUVT}, P_U, I_U, and B_U^0. After appending the biometric data (B_U^0) to V_1 (necessary for eliminating the problem of biometric variance), another set of layered encryption is carried out as shown in Fig. 4.3 to arrive at V_f. The server sends this file to the user. When the user wants to play the file, the keys used in encryption are used in reverse order to find V_2. Since this data (V_2) contains B_U^0, the biometric data obtained online from the sensor, B_U^1, can be matched with B_U^0. If there is a positive match, i.e., B_U^0 and B_U^1 are likely to be from the same finger (or face, voice, iris, etc.), a final set of layered decryption is carried out to arrive at the actual multimedia content V. This will enable the media player to play the content V for user U. Note that if the user U wants to play the multimedia data at a different time, say, t_2, then B_U^0 will be matched with B_U^{t2} and so on.

In the above secure data transfer scheme, we assume a "closed application", where the decrypted file is not stored at the user's computer but decrypted just before it is played. The biometric sensor, matcher, decryption module, media player and playing medium (e.g., monitor, speaker, etc.) are assumed to be connected together securely, where no tampering (that could allow access to the decrypted file) is possible.

4.4 Computational Requirements

When encryption and decryption are utilized in any system, the computational requirements become an important issue, since these additional operations (that increase the security) may render the overall system impractical. The computational requirements of an asymmetric key system are several orders of magnitude larger than that of a symmetric key system. Hence, we use the former just for processing relatively small amounts of data, such as user identity, password, etc., and the latter for encrypting the multimedia data itself (which can be huge in size; for example a typical 3 min. music encoded in MP3 format can occupy 5 MB).

In the next section, we provide the computing times measured for typical encryption and decryption processes via the Advanced Encryption Standard (AES) [44], which is a successor for the traditional Data Encryption Standard (DES) [43]. AES provides stronger security, not only due to increased key size (128 bits for AES, and 56 bits for DES) but also due to the design of the encryption algorithm itself. As a result, it is replacing DES and its variants (e.g., 3-DES) in both government and commercial applications. We have also implemented the system using DES during the initial phases of this research [66].

As an alternative to generic systems such as AES and DES, cryptosystem architectures that are specifically designed for multimedia data (e.g., [71]) can be used for reducing the time complexity and increasing the applicability of the system, especially for real-time applications.

The utilization of biometric matching in encryption leads to two additional considerations. Due to intra-class variations and inter-class similarities in biometric identifiers, every biometric system leads to some false rejects (conveyed via FRR or False Reject Rate) and some false accepts (conveyed via FAR or False Accept Rate). Below, we provide the results from recent government and academic evaluations of fingerprint verification systems:

- FVC (Fingerprint Verification Competition) 2002 [32]: Publicly available (but small-sized) databases are used. The results can be good indicators for commercial system performances. The best system had an FRR of 0.28% at the FAR of 0.1%.

- FVC (Fingerprint Verification Competition) 2004 [33]: The best system had an FRR of 4.7% at the FAR of 0.1%. Note that the utilized databases were more complex (e.g., due to larger, exaggerated finger distortions) than those of FVC

2002, hence performance of state-of-the art in fingerprint matching may seem to be decreasing (cf. FRR's of 4.7% vs 0.28%). But this database difference is the key factor in the observed performance deviation.

- NIST FpVTE (Fingerprint Vendor Technology Evaluation) 2003 [46]: Government databases (large size) are used. The results can be good indicators for government application system performances. The best system had an FRR of 0.4% at the FAR of 0.01%.

- NIST SDK (Software Development Kit) Test 2005 [45]: Again, large-sized government databases are used. The best system had an FRR of 0.99% at the FAR of 0.01%. The system evaluation architecture (vendor supplied hardware for FpVTE vs. common government hardware for SDK Test) and utilized databases affect the performance deviation.

As an example, we will consider the FVC 2004 evaluation. For the cited FAR value of 0.1%, FRR of 4.7% would imply that a genuine user will not be accepted by the fingerprint matcher (approximately once in 20 tries) and therefore will not be able to play the multimedia content that she has legitimately acquired (a nuisance for the genuine user). While user habituation will decrease this error significantly, to eliminate this problem and reduce the FRR, the sensor may capture the same biometric more than once to increase the probability of a match. For example, if two impressions are captured, the FRR may reduce to 0.0017 ($=1 - [(1 - 0.047) + 0.047 * (1 - 0.047)]$).

Also, multiple biometric modalities (such as fingerprint, iris, etc.) can be used in encryption and decryption processes, and matching of any single biometric modality can suffice for initiating the decryption of the encrypted file.

The FAR value of 0.1% indicates that a user (e.g., Alice) will be able to play the multimedia file encrypted with *another* user's (e.g., Bob's) fingerprint, with a probability of 0.1% (resulting in pirate playing of multimedia content with 1 in 1,000 probability). This false accept phenomenon is present in any system including biometric components, and as such, it is a drawback. Nevertheless, its effect can be decreased by tuning the biometric matcher's decision threshold so as to decrease FAR (but at the expense of increasing the associated FRR).

Another issue in using biometric data is the time needed for verifying a user. The FVC 2004 study [33] reported that the verification time for the best fingerprint matcher was 1.48 seconds (for a 1.41 GHz processor). This suggests that fingerprint matching is viable for use in encryption/decryption processes to secure multimedia data as outlined in Fig. 4.3.

4.5 Experiments and Results

Here, we provide encryption and decryption times for the application of AES symmetric cipher on multimedia files. The standard key length in AES is 128-bits. Hence, the user ID (I_U), the user selected password (P_U), and server generated password (P_{SUVT}) are used directly as AES keys, where these values are 16-character strings composed of 8-bit ASCII code. The biometric data (B_U^t, $t = 0, 1, 2, \ldots$) are generally larger in size; for example, a typical fingerprint image may generate a feature vector (composed of minutiae location and orientation data) that is more than 600-bits [23]. Similarly, iris images generate a feature vector (IrisCode) with a 2048-bit length. These feature vectors can be converted to 128-bit keys via one-way hash functions, and then utilized as AES keys. In this study, we used the MD5 [56] hash function on the fingerprint feature vectors to arrive at 128-bit hash values and used them as biometric-based encryption and decryption keys.

On a 3.2 Ghz Pentium 4 processor machine, the encryption (totally 7 epochs) and decryption (totally 7 epochs) of a 5 MB file took 1.8 seconds each. This time is acceptable since the decryption is only carried out once before playing the multimedia file. Furthermore, utilization of special hardware chips can reduce these times substantially [56].

4.6 Summary

In this chapter, a multimedia content protection scheme was proposed. It is based on layered encryption/decryption involving biometric authentication. Utilization of fingerprints as keys in encryption/decryption procedures eliminates the feasibility of illegal key sharing, which hampers the content protection schemes based solely on traditional keys.

The computation times required for the necessary encryption and decryption processes are provided for AES symmetric-key system. These times show the applicability of the method. Utilization of widely available encryption/decryption systems (e.g., AES and previously DES) increases the applicability even further. Custom hardware chips will reduce these times in future applications.

Chapter 5

Biometric Cryptosystems

5.1 Introduction

As seen in Section 4.2.2, current cryptographic systems (e.g., DES, AES, RSA) have high theoretical (related to the complexity of the underlying building blocks in their construction) and proven (related to the infeasibility of the attacks mounted against them) security for *containing* the plain text data. Nevertheless, we showed that illegal key sharing (key management problem) is one of their major drawbacks: regardless of the security of the algorithms, if the keys that need to be known *only* to the legitimate parties in the communication are shared freely, it is trivial to convert cipher text back to plain text. As a solution to this problem, we have proposed to use biometric identifiers as keys in traditional cryptographic systems described in Chapter 4.

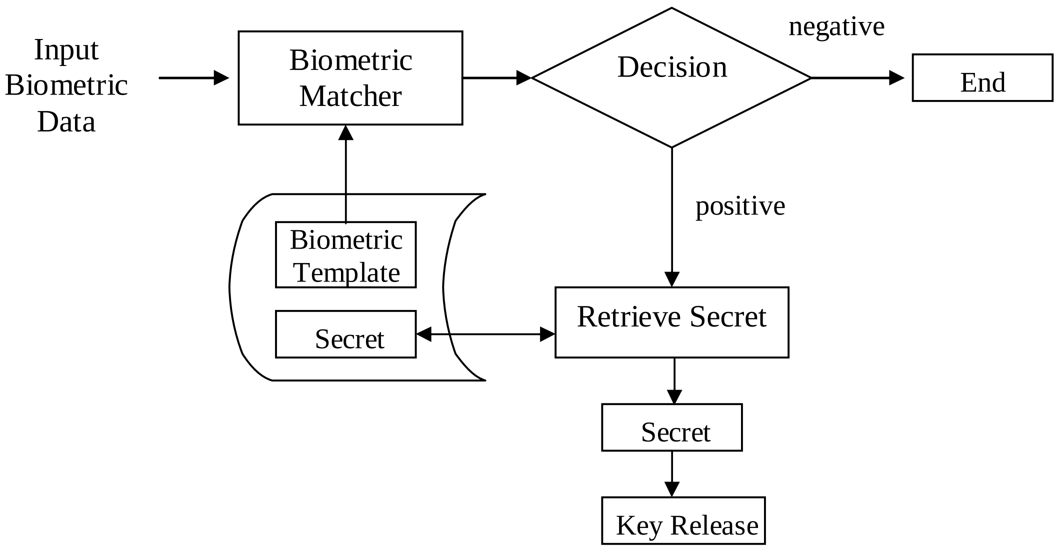
Another limitation of the cryptographic systems is that they require the keys to be very long and random for high security. For example, AES requires at least 128-bits (which corresponds to a 19 character key from a 7-bit ASCII code). This makes it impossible for users to remember the keys. As a result, the cryptographic keys are stored within a physical medium (e.g., in a computer or on a smart card) and released based on some alternative authentication mechanism. If this mechanism succeeds, the released key can be used in encryption/decryption procedures.

The most popular authentication mechanism used for this purpose is based on passwords, which are again cryptographic key-like strings but they are simple enough for users to remember (hence it is not necessary for users to store this information within a physical medium). As an example, the string "CoTtOn1970+" can be selected as a password by someone who was born in 1970 and who has a cat named *Cotton*: she can remember it easily *and* she hopes that it can not be guessed by attackers. Hence, the plain text (e.g., e-mail records, financial records) protected by a cryptographic algorithm is only as secure as the password (the weakest link) that releases the correct decrypting keys. Simple passwords (e.g., "cotton") compromise security: they can be guessed, either by using social engineering methods (observing the names of pets, relatives, favorite movies ...), or by brute force search. In fact, even though the theoretical password space can be quite large (for 8 character passwords from 7-bit ASCII code, there are $128^8 \approx 7.2 \cdot 10^{16}$ different passwords), in an experiment involving 13,797 Unix passwords, Klein [27] was able to crack approximately 25% of the passwords using a dictionary including just 62,727 words. As a natural remedy to this problem, complex passwords are, however, difficult to remember and expensive to maintain (e.g., due to calls to helpdesks to reset a password if the user forgets it). Furthermore, passwords are unable to provide non-repudiation: a subject may deny releasing the key using password authentication, claiming that her password was stolen and that a thief released the key.

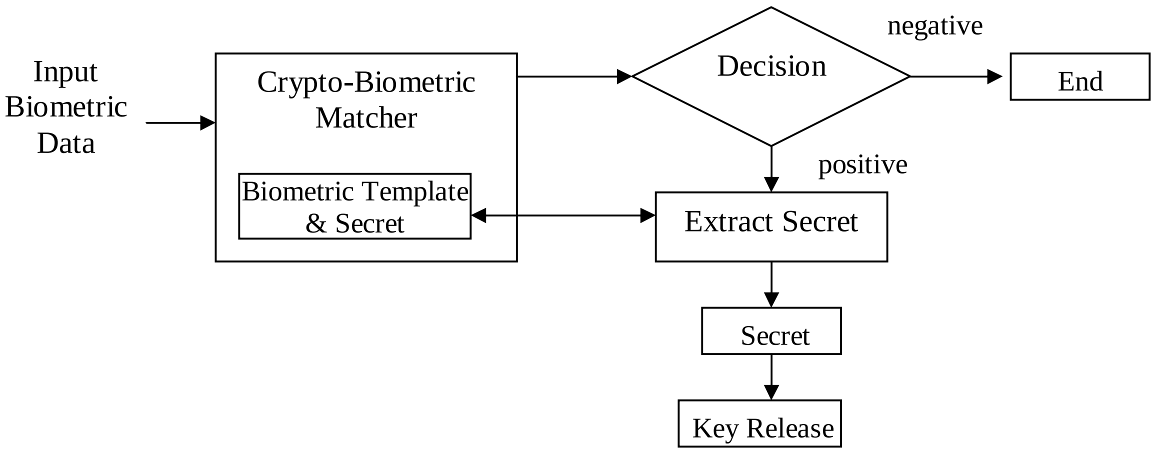
Many of the above limitations of password-based authentication can be eliminated by incorporating biometric authentication into the cryptographic system. This can be done in one of the following two modes (Fig. 5.1): (i) in biometrics-based key release, the biometric matching is decoupled from the cryptographic part. Biometric matching operates on the traditional biometric templates: if they match, the cryptographic key is released from its secure location, e.g., a smart card or a server. Here, biometrics effectively acts as a wrapper mechanism for the cryptographic domain, (ii) in biometrics-based key generation, biometrics and cryptography are merged together at a much deeper level. Biometric matching can effectively take place *within* the cryptographic domain, hence there is no separate matching operation that can be attacked. In other words, biometric matching does not function as a wrapper around the stored secret key; instead, positive biometric matching extracts the secret key from the conglomerate key/biometric template data.

Note that the biometrics-based solution to the illegal key sharing problem we introduced in Chapter 4 uses biometric identifiers *as* cryptographic keys. However, the two alternatives to password-based authentication that we introduce here (i.e. biometrics-based key generation and biometrics-based key release) aim at *containing* the cryptographic keys: once the keys are released as shown in Fig. 5.1, they can be used directly in asymmetric or symmetric schemes of Fig. 4.1.

In this chapter, we propose a key generation algorithm operating on fingerprint features (Section 5.4). First, however, we review the algorithms (including the algorithm of Juels and Sudan [25], which forms the basis of our system) proposed in the



(a)



(b)

Figure 5.1: Two modes of combining biometrics with cryptography: (a) key release, (b) key generation.

literature associated with both of the biometric cryptosystem modes and highlight their relative strengths and weaknesses.

5.2 Background

Soutar et al. [62, 61, 63] proposed a "key binding" algorithm for an optical correlation-based fingerprint matching system. This algorithm binds a cryptographic key with the user's fingerprint image at the time of enrollment. The key is then retrieved only upon successful authentication. By using several training fingerprint images of a finger (typically 5), the algorithm first creates a correlation filter function $H(u)$, which has both the magnitude ($|H(u)|$), and phase ($e^{i\varphi(H(u))}$) components. The design criteria for this function include both distortion tolerance (in order to minimize FRR) and discriminability (in order to minimize FAR). The algorithm also outputs $c_0(x)$, which is obtained by convolution/correlation of the training fingerprint images with $H(u)$. Then, the complex conjugate of the phase component of H(u), $e^{-i\varphi(H(u))}$, is multiplied with a randomly generated phase-only array of same size, resulting in $H_{stored}(u)$, and the magnitude of $H(u)$ is discarded. This process eliminates the possibility of reverse engineering a user's fingerprint image from $H(u)$. A given or randomly generated N-bit (typically 128-bit) cryptographic key, k_0, is then linked with binarized correlation output c_0 by using an error-correcting code (in order to tolerate some expected variation in the biometric signal during authentication), resulting in a lookup table, LT. This cryptographic key is also used as an encryption key to encrypt S bits of $H_{stored}(u)$ and the resultant encrypted message is hashed (using a standard hashing function such as SHA-1 or Triple-DES [56]) to form an identification code id_0. Finally, $H_{stored}(u)$, LT, and id_0 are stored in the database as the biometric template for the user (called Bioscrypt by the authors).

During authentication, the user inputs one or more (typically 5) fingerprint images of her finger. The $H_{stored}(u)$ for this user is retrieved from her stored Bioscrypt, and combined with the input fingerprint images to produce a correlation output $c_1(x)$. A cryptographic key retrieval algorithm then uses the LT of the user (stored in her Bioscrypt) to extract a key k_1 from the correlation output $c_1(x)$. The retrieved key, k_1, is used to create a new identification code id_1 in exactly the same way as was done during enrollment. If $id_1 = id_0$, then k_1 is released into the system, else an "authentication failed" message is returned. Thus, the system never releases any wrong key into the system if the biometric authentication fails. The main criticism of Soutar et al.'s work in the literature [12, 25] is that the method does not carry rigorous security guarantees. The authors do not explain how much entropy is lost at each stage of their algorithm. Further, the resulting FAR and FRR values associated with key release are unknown. The authors also assume that the input and database fingerprint images are perfectly aligned. Even with a very constrained image acquisition system, it is unrealistic to acquire fingerprint images without any misalignment.

Davida et al. [12, 13] propose an algorithm based on iris biometric. They consider binary representation of iris texture, called IrisCode [11], which is 2,048 bits in length. The biometric matcher computes the Hamming distance between the input and database template representations and compares it with a threshold to determine whether the two biometric samples are from the same person or not. The authors assume that the IrisCodes from different images of the same iris can have up to 10% of the 2,048 bits (i.e. 204 bits) different from the same iris's template IrisCode. The authors also assume that the IrisCodes of different irises differ in nearly 45% of the 2,048 bits (i.e. 922 bits).

During enrollment, multiple scans of a person's iris are collected and K-bit ($K = 2,048$) IrisCodes are generated for each scan. These IrisCodes are then combined (through a majority decoder) to arrive at a canonical IrisCode, T, of the same length. An $[N, K, D]$ (where N: codeword size, and D: the minimum distance between codewords) bounded distance decoding error correcting code [30] is then constructed by adding C check bits to the K-bit IrisCode (C is determined such that 10% of K bits can be corrected) resulting in an N-bit codeword, denoted by $T||C$. The codeword, $T||C$, is hashed and digitally signed, denoted by $Sig(Hash(T||C))$, and together with the check bits C, stored as the database template for the user. At the time of authentication, multiple samples of the iris of a person are collected and T' is estimated. The check bits C from the database template are used to perform error correction on the codeword $T'||C$ and the corrected IrisCode T'' is produced. Then $T''||C$ is hashed and signed (just like during enrollment), resulting in $Sig(Hash(T''||C))$. If $Sig(Hash(T''||C))$ is exactly the same as the $Sig(Hash(T||C))$ stored in the database template, authentication succeeds. Davida et al. [12, 13] argue that the database template of a user itself can be used as a cryptographic key (note that this key would always be the same for the same biometric identifier in contrast to cryptographic key binding algorithms such as the Soutar et al.'s algorithm [62, 61, 63] that can bind any random/given key with biometric data). If a chosen biometric does not provide the desired entropy (i.e. cryptographic strength), the authors propose that a password, PIN, or multiple biometrics can be added to the system to increase the entropy. Davida et al.'s [12, 13] algorithm is fast and provably secure. However, they propose to store error-correcting bits C in the database, which may lead to some leakage of information (that can be exploited by an attacker) about the user's biometric data. Further, the error tolerance of their scheme is rather small. The authors' assumption that only 10% bits of IrisCode change among different presentations of a person's iris is too restrictive. In fact, up to 30% bits of IrisCode could be different between different presentations of the same iris [11]. The authors assume that by acquiring a large number of samples of the iris, the errors in the IrisCode could be significantly minimized. Finally, the authors assumed that the input and database template IrisCodes are completely aligned. Although constrained iris image acquisition systems can limit the misalignment among different acquisitions of the same iris, some degree of misalignment is natural.

Monrose et al. [41] proposed a method to make passwords more secure by combining keystroke biometrics with passwords. Their technique was inspired by password "salting", where a user's password (pwd) is salted by prepending it with an s-bit random number (the "salt"), resulting in a hardened password ($hpwd$). During enrollment, the following information is stored in the user's database template: (i) a randomly chosen k-bit (typically, $k = 160$) number r, (ii) an "instruction table" encrypted with pwd (the instruction table is created as follows: first, the user's keystroke features (typically 15 in number) are thresholded to generate a binary feature descriptor, then the binary feature descriptor and r are used to create the instruction table using Shamir's secret sharing scheme [56] (the instruction table essentially contains instructions on how to generate *hpwd* from the feature descriptor, r, and *pwd*)), and (iii) a "history file" encrypted with *hpwd*. At the time of authentication, the algorithm uses r and the instruction table from the user's template and the authentication password pwd' and keystroke features acquired during the authentication to compute $hpwd'$. The $hpwd'$ is then used to decrypt the encrypted history file. If the decryption is successful, the authentication is considered successful, and the r and history file of the user are modified in the template; if the authentication is unsuccessful, another instance of $hpwd'$ is generated from the instruction table in a similar way but with some error correction. If the authentication does not succeed within a fixed number of error correction iterations, the authentication finally fails. The authors claim that the hardened password itself can be used as an encryption key. A weakness of this work is that it only adds about 15-bits of entropy to the passwords, thus making them only marginally more secure. However, in their subsequent work [40, 39, 38], Monrose et al. made minor modifications to their original scheme, applied it to voice biometrics (which is more distinctive than keystroke biometrics), and were eventually able to generate cryptographic keys of up to 60-bits, which although much higher than the 15-bits achieved in their earlier work, is still quite low (e.g., compared to 128-bits of AES) for most security applications. One of the strengths of Monrose et al.'s work is that they have demonstrated the practicality of their algorithm through experiments. The authors also implemented their scheme on a resource-constrained device (Compaq's off-the-shelf IPAQ Personal Digital Assistant) [38]. Different applications can use different cryptographic keys for a person (or the same application can change the cryptographic key upon re-enrollment) by using different content (i.e. a different typed or uttered password).

Tuyls et al. [31] assume that a noise-free template X of a biometric identifier is available at the enrollment time and use this to enroll a secret S to generate a helper data W. They assume that each dimension (of a multidimensional template) is quantized at q resolution levels. In each dimension, the process of obtaining W is akin to finding residuals that must be added to X to fit to an odd or even grid quantum depending upon whether the corresponding S bit is 0 or 1. At decryption time, the biometric template Y (a noisy version of X) is used to decrypt W to obtain a decrypted message S' which is approximately the same as S. In each dimension, the process of decryption guesses whether a particular bit of secret S is 0 or 1 depending upon whether the sum of Y and W resides in an even or odd quantum of the corresponding dimension. It is hoped that the relatively few errors in S' can be corrected using error-correction techniques, hence the resultant S can be used as a key in traditional cryptosystems. Their technique assumes that the biometric representations are completely aligned and that noise in each dimension is relatively small compared to the quantization Q. Due to variability in biometric identifiers, different W may be generated for the same message S. The authors prove that very little information is revealed from W by appropriately tuning the quantization scheme with respect to the measurement noise.

In their "fuzzy commitment" scheme [26], Juels and Wattenberg generalized and significantly improved Davida et al.'s methods [12, 13] to tolerate more intra-class variation in the biometric characteristics and to provide stronger security. In the fuzzy commitment scheme, the user at the enrollment time selects a secret message C. Let d denote the difference vector between the user biometric key X and C. The encrypted message (which is considered as a fuzzy commitment F: user commits to secret C) then consists of d and $y = hash(C)$, where hash is a one-way hash function such as SHA-1 [56]. At the decrypting end, with biometric representation Y, $(Y + d)$ is used to decode the nearest codeword C'. Again, with the help of error correcting techniques, it is hoped that the error in C' can be corrected to obtain the original message C. A simple numerical example for this system is as follows: Assume the secret message C is selected from the universal code space of $\{0000000000, 0000011111, 1111100000, 1111111111\}$, e.g., $C = (0000011111)$. Also, assume that the error correction (f) is based on majority decoding on 5-bit blocks (e.g., 01010 decodes to 00000, and 11001 decodes to 11111). If the user biometric key (e.g., fingerprint template) is $X = (0101010101)$, it follows that $d = X - C = (0101001010)$. The fuzzy committment is then $F = (hash(0000011111), 0101001010)$. During decrypting, the user can provide the fingerprint template $Y = (1101011101)$. Note that X and Y differ in two bits. Finally, $hash(f(Y - d)) = hash(f(1000010111)) = hash(0000011111)$, which is equivalent to the hash value in commitment F, hence verification succeeds. The authors acknowledge that one of the major shortcomings of the fuzzy commitment scheme is that it requires the biometric representations X and Y to be ordered so that their correspondence is obvious.

Juels and Sudan's fuzzy vault scheme [25] is an improvement upon previous work by Juels and Wattenberg [26]. Assume Alice is a legitimate user of this scheme, and Bob is an attacker. In [25] Alice can place a secret κ (e.g., secret encryption key) in a vault and lock (secure) it using an unordered set A. Here, an unordered set means that the relative positions of set elements do not change the characteristics of the set: e.g., the set $\{-2, -1, 3\}$ conveys the same information as $\{3, -1, -2\}$. Bob, using an unordered set B, can unlock the vault (access κ) only if B overlaps with A to a large extent. The procedure for constructing the fuzzy vault is as follows: First, Alice selects a polynomial p of variable x that encodes κ (e.g., by fixing the coefficients of p according to κ). She computes the polynomial projections, $p(A)$, for the elements of A. She adds some randomly generated chaff points that do not lie on p, to arrive at the final point set R. When Bob tries to learn κ (i.e., find p), he uses his own unordered set B. If B overlaps with A substantially, he will be able to locate many points in R that lie on p. Using error-correction coding (e.g., Reed-Solomon [30]), it is assumed that he can reconstruct p (and hence κ). A simple numerical example for this process is as follows: Assume Alice selects the polynomial $p(x) = x^2 - 3x + 1$, where the coefficients (1, -3, 1) encode her secret κ. If her unordered set is $A = \{-1, -2, 3, 2\}$, she will obtain the polynomial projections as $\{(A, p(A))\} = \{(-1, 5), (-2, 11), (3, 1), (2, -1)\}$. To this set, assume Alice adds two chaff points $C = \{(0, 2), (1, 0)\}$ that do not lie on p, to find the final point set $R = \{(-1, 5), (-2, 11), (3, 1), (2, -1), (0, 2), (1, 0)\}$. Now, if Bob can separate at least 3 points from R that lie on p, he can reconstruct p, hence decode the secret represented as the polynomial coefficients (1, -3, 1). Otherwise, he will end up with an incorrect p, and he will not be able to access the secret κ.

The security of the scheme is based on the infeasibility of the polynomial reconstruction problem (i.e., if Bob does not locate many points that lie on p, he can not feasibly find the parameters of p, hence he cannot access κ). The scheme can tolerate some differences between the entities (unordered sets A and B) that lock and unlock the vault, so Juels and Sudan named their scheme *fuzzy* vault. This fuzziness can come from the variability of biometric data: even though the same biometric entity (e.g., right index finger) is analyzed during different acquisitions, the extracted biometric data will vary due to acquisition characteristics (e.g., placement of the finger on the sensor), sensor noise, robustness of the feature extractor, etc. The fuzzy vault scheme is expected to tolerate these intra-class variations. On the other hand, in traditional cryptography, if the keys are not exactly the same, the decryption operation will fail.

Note that since the fuzzy vault can work with unordered sets (common in biometric templates, including fingerprint minutiae data), it is a promising candidate for biometric cryptosystems. Having said this, the fuzzy vault scheme requires *pre-aligned* biometric templates. Namely, the biometric data at the time of enrollment (locking the vault) must be properly aligned with the biometric data at the time of verification (unlocking the vault): following the example given above, Bob is able to unlock the vault if he selects the set $B = \{-1, -2, 2, 0\}$, since he manages to identify three points $\{$(-1,5),(-2,11),(2,-1)$\}$ that lie on Alice's polynomial p. When we extend this example to biometric templates, if the entity (e.g., minutiae location), say, $\{-5\}$ is in fact *biometrically* equivalent to element $\{-1\}$ (e.g., due to just 4 pixel shift of the coordinate frame), $\{-5\}$ should be used as $\{-1\}$ during vault unlocking, hence the need for alignment arises.

Alignment is an essential requirement due to different types of distortion that can occur during biometric data acquisition. Further, the number of feasible operating points (where the vault operates with *negligible* complexity, e.g., measured via the number of required access attempts to reveal the secret for a genuine user and with *considerable* complexity for an imposter user) for the fuzzy vault is limited.

Dodis et al. [15] proposed theoretical foundations for generating keys from the "key material" that is not exactly reproducible (e.g., passphrases, answers to questionnaires, biometric data that changes between enrollment and verification). Similar to the notion of helper data of Tuyls et al. [31], Dodis et al. [15] define *fuzzy extractors* (FE) that create variable R from the key material w, plus generate public (helper) data P. FE again generates R from w', if w' is "close" to w, given the variable P. For three distance metrics (Hamming distance, set difference and edit distance), Dodis et al. calculate the information revealed by P, and elaborate on the existence of possible algoritms for FE construction. They also propose a modification of the Juels and Sudan's fuzzy vault scheme [25]: instead of adding chaff points to the projections of the polynomial p, Dodis et al. [15] propose to use a polynomial p' (of degree higher than p) which overlaps with p only for the points from the genuine set A. The security of the scheme is based on the degree of this new polynomial p', which replaces the final point set R of Juels and Sudan's scheme [25].

Clancy et al. [6] propose a "fingerprint vault" based on the scheme of Juels and Sudan [25]. At the enrollment time, multiple (typically 5) fingerprints of users are acquired. The fingerprint representation (minutiae positions) is extracted from each fingerprint. Correspondence between feature points (minutiae) extracted from the multiple prints is established using a bounded nearest-neighbor algorithm. That is, when different prints of a finger are overlaid on top of each other, the minutiae in one print that are within a close spatial proximity of minutiae in the other print are considered as the same ("corresponding"). Thus, corresponding minutia form clusters and these are used to estimate the variance of minutia location (d). The minutia for which the correspondence is found in at least a predetermined (typically 2) set of prints constitutes the effective feature representation (i.e. locking set, G). Given the fingerprint impression size and d, Clancy et al. add the maximal number of random (chaff) points ($N = 313$ in their implementation) to the feature representation that are at least a distance d ($d = 11$ in their implementation) away from all the other feature points. As in Juels and Sudan's work [25], the union of G and N constitutes the abscissa of the encoded message; the ordinates are determined by the polynomial embedding of the secret to be shared. At the decrypting end, given a user fingerprint (which is assumed to be pre-aligned with the enrollment fingerprint), the features are extracted. The features are used to find the corresponding points within the encoded message using the bounded nearest-neighbor algorithm based on the abscissa alone. The corresponding ordinates with the encoded message are fed into Reed-Solomon error correcting codes to recover the encoded polynomial. The authors simulated this error-correction step without actually implementing it. The strength of this work is that it concretely describes the fuzzy vault implementation in the fingerprint domain and concludes that it is possible to achieve an FAR of $2^{-69} \approx 1.7 \cdot 10^{-21}$ (e.g., 69-bit security), at an FRR of about $20 - 30\%$ for a fingerprint database comprised of 23 distinct fingers (5 impressions each).

Yang and Verbauwhede [73] attempted to eliminate the pre-alignment requirement of Clancy et al.'s [6] algorithm. First, the enrollment fingerprint is analyzed to find a “reference minutia”: it is defined as a minutia (i) that is present in all of the multiple (e.g., 3) fingerprint impressions, and (ii) whose local structure (based on its distance and orientation with respect to its two nearest-neighbor minutiae) does not change. The fingerprint that will lock the vault is registered with respect to this reference minutiae (using the reference minutia as the origin of a new coordinate frame), and the vault is generated similar to Clancy et al.’s [6] algorithm, by using minutiae coordinates. For unlocking the vault, the same steps are applied to query fingerprint images to find another “reference minutia”. Now, if the two reference minutiae (the one found during locking the vault, and the one found during unlocking the vault) are the same, the origins of the coordinate frames used during locking and unlocking would be the same, and hence, the alignment could be established. Yang and Verbauwhede [73] assumed this to be the case. Note that this assumption is not guaranteed to be true, but the authors did not provide details about this issue. In an experiment involving a very small database (comprised of 10 distinct fingers, with 10 impressions each), they reported a FRR of 17% with no false accepts.

In Table 5.1, we provide a comparison of these algorithms (Soutar et al. [62, 61, 63], Davida et al. [12, 13], Monrose et al. [41, 40, 39, 38], Tuyls et al. [31], Juels and Sudan [25], Dodis et al. [15], Clancy et al. [6], and Yang and Verbauwhede [73]). The third column indicates the key release (R) or key generation (G) classification. In the remaining columns, the qualifications *high*, *medium*, and *low* are denoted by H, M, and L, respectively. Practicality deals with the complexity of the algorithm. The sensitivity of a scheme was assessed based on our qualitative perception of whether the algorithm can tolerate realistic variations in the biometric signal such as noise, variable-length representation and unaligned representation. Rigorous security analysis for the first two algorithms in Table 5.1 has not been provided (denoted by N/A).

Note that the scheme that we propose later in this chapter provides a concrete fingerprint-based implementation of the Dodis et al. [15] algorithm (hence enjoying the associated characteristics such as high practicality and security) and furthermore eliminates the sensitivity to variations cited above.

Algorithm	**Biometric**	**Mode**	**Privacy Protection**	**Practicality**	**Sensitivity**	**Security**
Soutar et al. [62, 61, 63]	Fingerprint	R	H	M	H	N/A
Davida et al. [12, 13]	Iris	G	H	H	L	N/A
Monrose et al. [41, 40, 39, 38]	Keystroke, voice	G	H	H	H	M
Tuyls et al. [31]	No evaluation	G	H	L	L	H
Juels and Sudan [25]	No evaluation	G	H	H	L	H
Dodis et al. [15]	No evaluation	G	H	H	L	H
Clancy et al. [6]	Fingerprint	G	H	H	M	H
Yang and Verbauwhede [73]	Fingerprint	G	H	H	M	H

Table 5.1: Comparison of various biometrics-based key generation and key release algorithms.

5.3 Methods to Improve Biometric Cryptosystems

As seen above, the main problem of biometric cryptosystems is dealing with biometric variability. How can the system perform successfully even when the underlying entity (i.e. biometric identifier) differs in the *measurement* space (e.g., shift in minutiae coordinates), but is equivalent in the *biometric* space (e.g., the minutiae are identical even though there is a shift in their coordinates)? Note that the inherent construction of biometric cryptosystems relies on processing data in *their* own space (e.g., point lists as in Juels and Sudan's algorithm [25]), and *not* in the biometric space, complicating the problem further.

This biometric variability can arise from (i) missing/extraneous features, (ii) unordered features, (iii) misalignment of features, (iv) measurement noise, and (v) distortion in the sensed biometric. In this thesis, we aim to develop an algorithm that is immune to these variability sources.

5.4 Architecture of the Proposed System

In this section we present our implementation of the fuzzy vault, operating on fingerprint minutiae features. Note that first, we assume that fingerprints used in the construct are pre-aligned, and then develop an automatic alignment scheme in Section 5.7. The minutiae features are shown in Fig. 5.2 (the images are from the IBM-GTDB [68] database). The minutiae are found using the algorithm outlined in [23]. Note that the variability in the number and position of minutiae (cf. Section 5.3) is evident.

Fig. 5.3 shows the block diagram of the proposed fingerprint fuzzy vault system. Note that the other realizations of fingerprint vaults (by Clancy et al. [6], and Yang and Verbauwhede [73]) simulated the error-correction step of fuzzy vault construction,

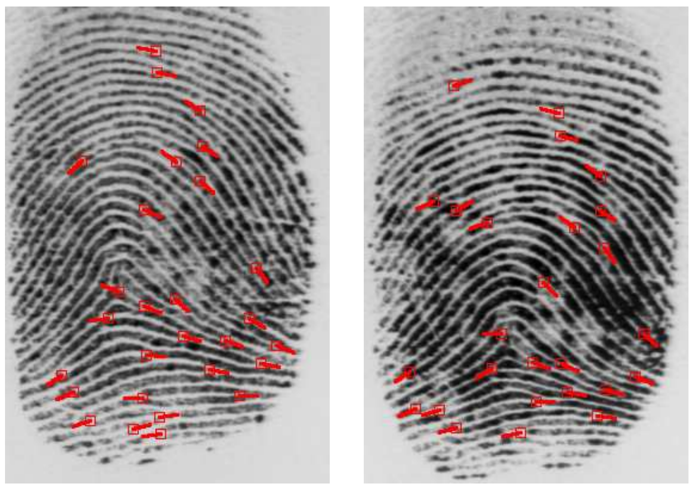


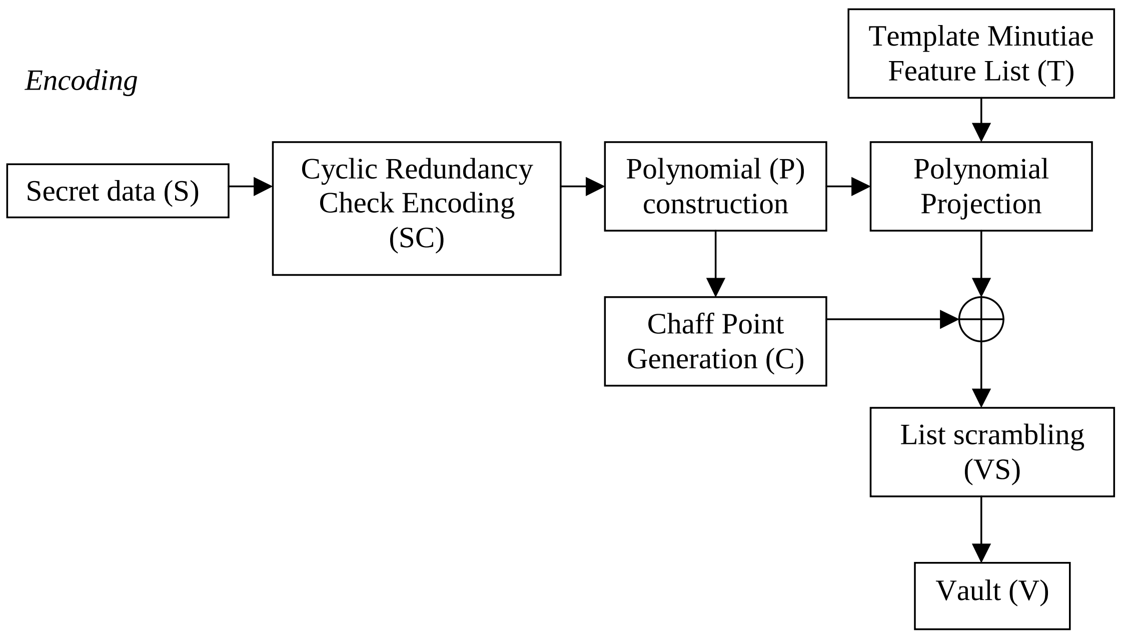
Figure 5.2: Fingerprint images with overlaid minutiae: (a) template image (28 minutiae), (b) query image (26 minutiae).

whereas our system includes actual locking/unlocking of the secret to be contained, without resorting to any simulations.

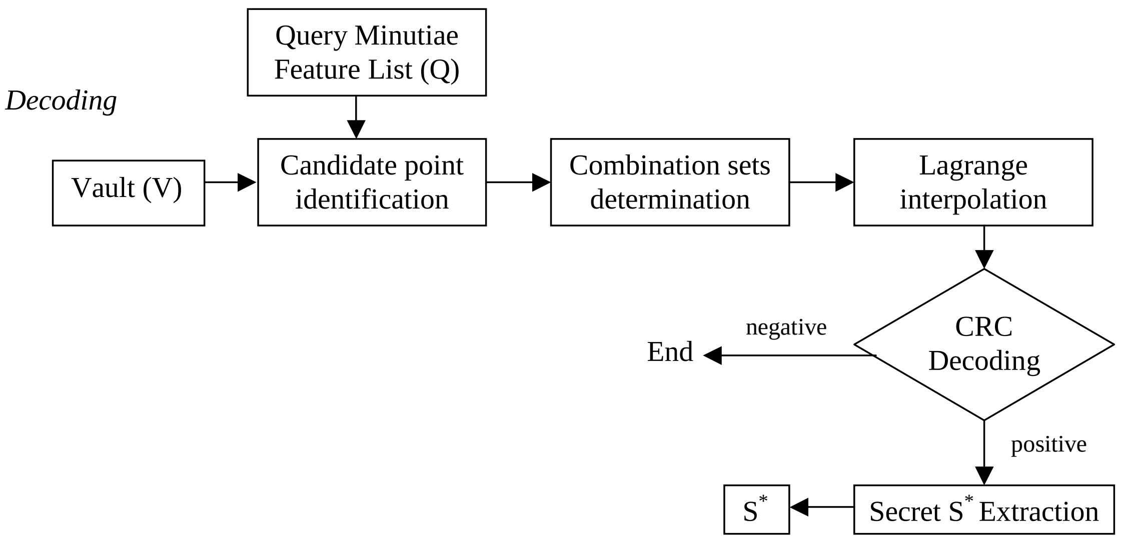
Fig. 5.4 shows the variables used in the system pictorially: the polynomial in Fig. 5.4(a) encodes the secret. It is evaluated at both genuine (black) and chaff (red) points in Fig. 5.4(b). Finally, the vault is the union of genuine and chaff points with no discriminating information (conveyed via color) attached to them.

5.4.1 Encoding

Secret S is any data that needs to be protected, but the size of S that can be feasibly protected is limited by the capacity of the entity used for locking and unlocking the vault. Currently, we use the x and y coordinates of minutiae points for locking/unlocking the vault (similar to the algorithms of Clancy et al. [6], and Yang and Verbauwhede [73]). We first align the minutiae, namely, compensate for the transla-


Encoding
Template Minutiae Feature List (T)
Secret data (S)
Cyclic Redundancy Check Encoding (SC)
Polynomial (P) construction
Polynomial Projection
Chaff Point Generation (C)
List scrambling (VS)
Vault (V)

(a)



(b)

Figure 5.3: Flowchart of the proposed fuzzy fingerprint vault: (a) vault encoding, (b) vault decoding.

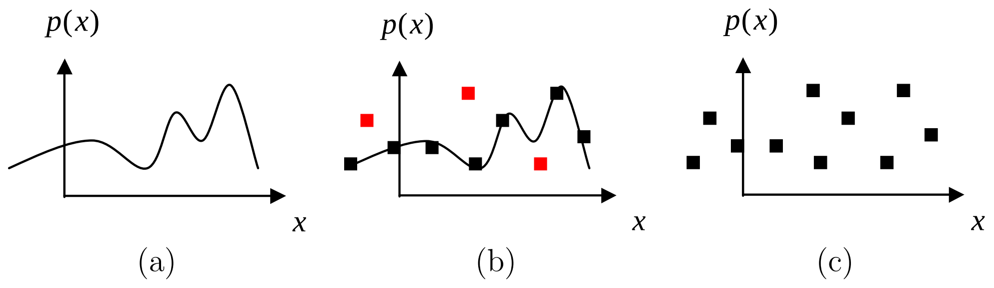


Figure 5.4: Pictorial representation of system variables: (a) polynomial, (b) evaluation of the polynomial (black: genuine points, red: chaff points), (c) final vault list.

tion and rotation between template and query minutiae data. The encoding operation secures S with fingerprint minutiae data: if a query minutiae set that is similar to the template minutiae set is presented during decoding, it indicates the presence of an authorized person, and S can be reconstructed accurately. Note that the vault operation is decoupled from any backend application (e.g., encryption/decryption using S): vault is only responsible for securing S with fingerprint data. The fingerprint template plays the role of a key. Note that this is not the key in traditional cryptosystems (e.g., AES) per se: rather, it has the role of a key for a new cryptographic construct, namely the fuzzy vault. In the current implementation, S is generated as a 128-bit random bit stream. This can simulate securing AES symmetric encryption keys.

Our current decoding implementation does not include any error-correction scheme, as proposed by Juels and Sudan [25], since realizing the necessary polynomial reconstruction via error-correction has not been demonstrated in the literature. Instead, our algorithm decodes many candidate secrets (as explained below). To identify which one of these candidates is the actual secret, we need to put some structure into the initial secret S. By checking the validity of this structure during decoding, the algorithm can identify whether a given candidate secret is correct or not.

Cyclic Redundancy Check (CRC) is a generalization of simple parity bit checking. It is commonly used in communication channel applications for error detection, where the errors are introduced due to channel noise. In our case, using incorrect minutiae points during decoding will cause an incorrect polynomial reconstruction, resulting in errors. In our current implementation, we generate 16-bit CRC data from the initial secret S. Hence, the chance of a random error being undetected (i.e., failing to identify an incorrect decoding) is 2^{-16}. The 16-bit primitive polynomial we use for CRC generation is called "CRC-16" and is used for EBCDIC messages by IBM [49]:

$$g_{CRC}(a) = a^{16} + a^{15} + a^{2} + 1. \tag{5.1}$$

Appending the CRC bits to the original secret S (128-bits), we construct 144-bit data SC. From this point on, all operations take place in Galois fields with cardinality 65,536, namely $GF(2^{16})$: we concatenate x and y coordinates of a minutia (8-bits each) as $[x|y]$ to arrive at the 16-bit locking/unlocking data unit u. To account for slight variations in minutiae data (due to nonlinear distortions), raw minutiae data are first quantized. Namely, each minutia is translated to originate from a square tessellation of the 2D image plane. For example, if the block size used in the tessellation is 7, any minutiae that has an x coordinate in the range [1, 7] is assumed to originate from the x coordinate 4. This allows for ± 3 pixel variations in the x and y coordinates of template and query minutiae. Fig. 5.5 shows this square tessellation:

black points indicate the centers of square blocks, which determine possible values for minutiae coordinates. Note that the tessellation covers the whole range of coordinates possible with 8-bit representation (i.e. $(x, y) \in \{(0,0), (0,1), (0,2), \ldots, (255,255)\}$).

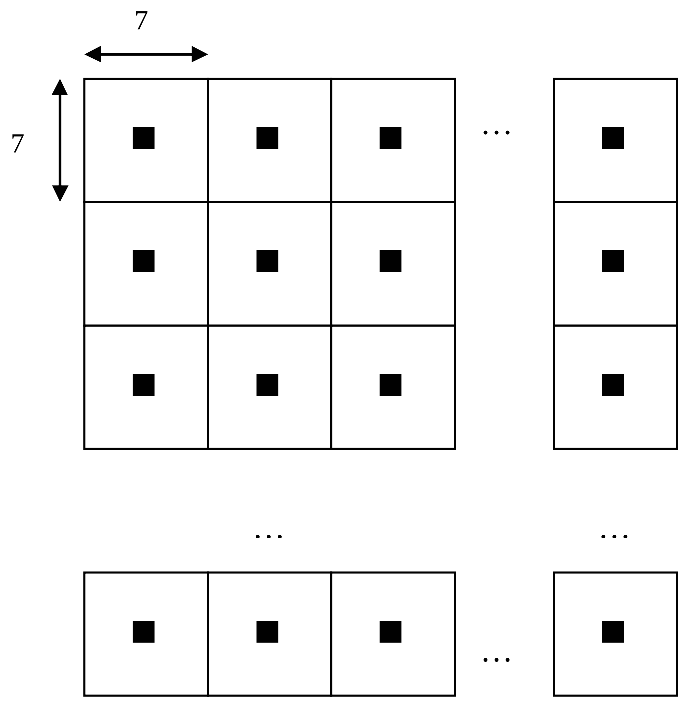


Figure 5.5: Square tessellation used in minutiae location quantization.

SC is used to find the coefficients of the polynomial p: 144-bit SC can be represented as a polynomial with 9 (144/16) coefficients, with degree $D = 8$:

$$p(u) = c_8 u^8 + c_7 u^7 + \ldots + c_1 u + c_0. \tag{5.2}$$

Simply, SC is divided into non-overlapping 16-bit segments, and each segment is declared as a specific coefficient, c_i, $i = 0, 1, 2, \ldots, 8$. Note that this mapping method (from SC to c_i) should be known during decoding, where the inverse operation takes place: decoded coefficients (c_i^*) are mapped back to decoded secret SC^*.

Now, two sets composed of point pairs need to be generated. The first one, called genuine set G, is found by evaluating $p(u)$ on the template minutiae features (T). Assuming that we have N template minutiae (if we have more than N minutia, we choose the first N sorted according to ascending u values), $u_1, u_2, \ldots, u_N$, we construct the set G

$$G = \left\{ \begin{array}{c} (u_1, p(u_1)) \\ (u_2, p(u_2)) \\ \vdots \\ (u_N, p(u_N)) \end{array} \right\}. \tag{5.3}$$

Note that the template minutiae, $u_1, u_2, \ldots, u_N$, are selected to be unique, namely, $u_i \neq u_k$, if $i \neq k$, where $i = 1, 2, \ldots, N$, $k = 1, 2, \ldots, N$.

The second set, called the chaff set C, ensures the security of the system. Assuming we need to add M chaff points, we first generate M unique random points, $c_1, c_2, \ldots, c_M$ in the field $GF(2^{16})$, with the constraint that they do not overlap with $u_1, u_2, \ldots, u_N$:

$$c_j \neq u_i, \quad j = 1, 2, \ldots, M, \quad i = 1, 2, \ldots, N \tag{5.4}$$

Then, we generate another set of M random points, $d_1, d_2, \ldots, d_M$, with the constraint that the pairs (c_j, d_j), $j = 1, 2, \ldots, M$ do not fall onto the polynomial $p(u)$. The chaff set C is then

$$C = \left\{ \begin{array}{c} (c_1, d_1) \\ (c_2, d_2) \\ \vdots \\ (c_M, d_M) \end{array} \right\} \tag{5.5}$$

where

$$d_j \neq p(c_j), \quad j = 1, 2, \ldots, M. \tag{5.6}$$

The union of these two sets, $G \cup C$, is finally passed through a list scrambler that randomizes the list, with the aim of removing any stray information that can be used to separate chaff points from genuine points. This results in the vault set VS

$$VS = \left\{ \begin{array}{c} (v_1, w_1) \\ (v_2, w_2) \\ \vdots \\ (v_{N+M}, w_{N+M}) \end{array} \right\}. \tag{5.7}$$

Note that increasing M increases the security of the system (namely, it decreases FAR (False Accept Rate) but it also has the potential to increase FRR (False Reject Rate)), and N is limited by the maximum number of feature points that can be extracted from the fingerprint. Along with VS, the polynomial degree D forms the final vault, V. Note that V can be transferred over insecure communication channels (e.g., Internet), stored in a local computer or a smart card, etc.

5.4.2 Decoding

Here, a user tries to unlock the vault V using the query minutiae features. Assuming that we have N (note that this number is the same as the number of genuine template minutiae in order to balance the complexity, e.g., measured via the number of required access attempts to reveal the secret for genuine and imposter users) query minutiae (Q), $u_1^*, u_2^*, \ldots, u_N^*$, the points to be used in polynomial reconstruction are found by comparing u_i^*, $i = 1, 2, \ldots, N$ with the abscissa values of the vault V, namely v_l, $\;l = 1, 2, \ldots, (N+M)$: if any u_i^*, $i = 1, 2, \ldots, N$ is equal to v_l, $\;l = 1, 2, \ldots, (N+M)$, the corresponding vault point (v_l, w_l) is added to the list of points to be used during decoding. Assume that this list has K points, where $K \leq N$.

Now, for decoding a degree D polynomial, $(D+1)$ unique projections are necessary. We find all possible combinations of $(D+1)$ points, among the list with size K. Hence, we end up with $\begin{pmatrix} K \\ D+1 \end{pmatrix}$ combinations. For each of these combinations, we construct the Lagrange interpolating polynomial. For a specific combination set given as

$$L = \left\{ \begin{array}{c} (v_1, w_1) \\ (v_2, w_2) \\ \vdots \\ (v_{D+1}, w_{D+1}) \end{array} \right\}, \tag{5.8}$$

the corresponding polynomial is

$$p^*(u) = \frac{(u-v_2)(u-v_3)\dots(u-v_{D+1})}{(v_1-v_2)(v_1-v_3)\dots(v_1-v_{D+1})}w_1 + \frac{(u-v_1)(u-v_3)\dots(u-v_{D+1})}{(v_2-v_1)(v_2-v_3)\dots(v_2-v_{D+1})}w_2 + \dots$$
$$\dots + \frac{(u-v_1)(u-v_2)\dots(u-v_D)}{(v_{D+1}-v_1)(v_{D+1}-v_2)\dots(v_{D+1}-v_D)}w_{D+1}. \quad (5.9)$$

This calculation is carried out in GF(2^{16}), and yields

$$p^*(u) = c_8^* u^8 + c_7^* u^7 + \dots + c_1^* u + c_0^*. \quad (5.10)$$

The coefficients are mapped back to the decoded secret SC^*. For checking whether there are errors in this secret, we divide the polynomial corresponding to SC^* with the CRC primitive polynomial, $g_{CRC}(a) = a^{16} + a^{15} + a^2 + 1$. Due to the definition of CRC, if the remainder is not zero, we are certain that there are errors. If the remainder is zero, with very high probability, there are no errors. For the latter case, SC^* is segmented into two parts: the first 128-bits denote S^* while the remaining 16-bits are CRC data. Finally, the system outputs S^*. If the query minutiae list Q overlaps with template minutiae list T in at least $(D+1)$ points for some combinations, the correct secret will be decoded, namely, $S^* = S$ will be obtained. This denotes the desired outcome when query and template fingerprints are from the same finger.

Note that CRC is an error detection method, and it does not leak any information that can be utilized by an imposter attacker (e.g., Bob). He cannot learn which one of the polynomial projections is wrong; hence, he cannot separate genuine points from chaff points.

5.5 Experiments with Pre-aligned Templates

We used the IBM-GTDB [68] fingerprint database (consisting of 229 mated image pairs with 500 dpi resolution and approximately of size 300x400) for obtaining the results presented in this section. The minutiae coordinates are linearly mapped to 8-bit range (e.g., the values [0, 255]) for both row and column dimensions before using them in locking/unlocking the vaults. In this database, a fingerprint expert manually marked the minutiae in every image. Further, the expert also established the correspondence between minutiae in mating fingerprint images (two fingerprint images per finger). Using this database has many advantages since (i) the adverse effects of using an automatic (and possibly imperfect) minutiae extractor are eliminated, and (ii) minutiae correspondence has been established. Note that we specifically chose to use such a database because it allows us to establish the upper bound on the performance of the fuzzy fingerprint vault. In our implementation, the pre-alignment of template and query minutiae sets is based on the correspondence marked by the expert (for an analysis of this requirement, see Section 5.6). The translation and rotation parameters that minimize the location error (in the least squares sense) between the corresponding minutiae are found, and the query minutiae sets are aligned with template minutiae sets.

We randomly generated a 128-bit secret S, and after appending the 16 CRC bits, the resulting 144-bits were converted to the polynomial $p(u)$ as

$$p(u) = 60467u^8 + 63094u^7 + \ldots + 52482u + 11995. \tag{5.11}$$

As explained in Section 5.4.1, 144-bit data was divided into 16-bit chunks, and each one of these chunks determined one of the 9 coefficients of $p(u)$. A total of 229 pairs of fingerprint images (of 229 users) from the IBM-GTDB database were used for locking and unlocking the vault with the following parameters: the number of template and query minutiae $N = 22$ (selected so that the number of candidate points will be enough for reconstruction), the number of chaff points $M = 200$ (the effect of this number on the security of the method against attackers is given below), and the block size used for quantization of minutiae x and y coordinates is 7 pixels (determined empirically; a larger value decreases the capacity of the vault, whereas a smaller value does not eliminate the minutiae variability).

By evaluating 9-element combinations of the candidate points, 195 out of 229 query fingerprints were able to successfully unlock the vault (which was locked by its mated fingerprint) and recover the secret S. The remaining 34 query fingerprints selected fewer than the required number of genuine points (i.e., less than 9) during decoding, hence they were unable to unlock the vault. As a result, the False Reject Rate (FRR) of the proposed system is 0.15 (=34/229), for the cited system parameters (so, the Genuine Accept Rate is 0.85, i.e., 85%).

It was observed that, during decoding, CRC performed with 100% accuracy: it signaled an error *if and only if* there was an error in the decoded polynomial.

For evaluating the corresponding False Accept Rate (FAR), we tried to unlock the vault with fingerprint templates that were *not* the mates of the templates that locked the vault. Hence, we have 228 imposter unlocking attempts for a distinct finger, and totally 52,212 (228x229) attempts for 229 users. Note that the unlocking templates were first aligned with the locking templates (to avoid giving an unfair disadvantage to imposter attempts), using the centers of mass of minutiae coordinates: the minutiae of the unlocking template were translated in x and y dimensions until their center of mass coincided with the center of mass of locking templates. None of the 52,212 attempts could unlock the vault. Hence, for this small database, the observed FAR is 0%. Alternatively, we can say that, since not even one false accept is observed within 52,212 attempts, $FAR < 1/52212 = 0.000019$).

5.6 Automatic Alignment within Fuzzy Vault

Automatic alignment of biometric templates is one of the main challenges in biometric cryptosystems. The transformed version of the template does not always allow alignment: they either do not carry the necessary information, or the transform hides such information. Note that during vault construction, the template minutiae (whose coordinates are identified via variable u) are mapped to the genuine set

$$G = \left\{ \begin{array}{c} (u_1, p(u_1)) \\ (u_2, p(u_2)) \\ \vdots \\ (u_N, p(u_N)) \end{array} \right\}, \tag{5.12}$$

where $N = 22$ in the current system. After adding chaff points (e.g., M=200), and list scrambling, the vault set is found as

$$VS = \left\{ \begin{array}{c} (v_1, w_1) \\ (v_2, w_2) \\ \vdots \\ (v_{218}, w_{218}) \end{array} \right\}. \tag{5.13}$$

Now, during decoding, there is no information to separate genuine minutiae from chaff minutiae. That is, the associated labeling, for example,

$$VS_{Labels} = \left\{ \begin{array}{c} genuine \\ chaff \\ genuine \\ chaff \\ chaff \\ \vdots \\ genuine \\ chaff \end{array} \right\} \tag{5.14}$$

is *not* known. Since the alignment procedure needs to work with this data (where the template minutiae are not available to align with the query minutiae), automatic alignment techniques (e.g., the algorithm in [23]) relying on both query (which is available) and template minutiae (which is not available) are not applicable.

As an alternative, auxiliary (helper) data can be used to assist in alignment, but such data can not reveal too much (i.e. enough to reconstruct the critical fingerprint template or the fingerprint image itself) information about the templates. This would lead to privacy problems that are intended to be addressed by the utilization of biometric cryptosystems to begin with. Faced with these two (generally contradicting) limitations (namely, (i) helper data should contain enough information to allow alignment, and (ii) helper data should not leak critical information about the templates), we develop a novel fingerprint alignment scheme that uses helper data. The proposed orientation-field based helper data does not leak minutiae or texture-related information, making it a feasible candidate for fingerprint-based biometric cryptosystems.

5.7 Orientation Field Flow Curves (OFFC) based Helper Data

The Orientation Field Flow Curves (OFFC) are sets of piecewise linear segments that represent the underlying flow of fingerprint ridges [9]. They are robust to noise arising from minutiae, islands, smudges, and cuts. These curves are obtained as follows. Firstly, the orientation field that shows the dominant orientation (o) in each 8x8 fingerprint block is found. A flow curve with a starting point s_O is then found iteratively as

$$s_j = s_{j-1} + d_j * l_j * o_{s_{j-1}} \tag{5.15}$$

where j denotes the index of points on the curve, $d_j \in \{-1, 1\}$ is the flow direction between s_j and s_{j-1}, l_j is the length of line segment (e.g., 5 pixels) between these two points, and $o_{s_{j-1}}$ is the orientation value at location s_{j-1}. By tracing points in the direction of respective orientation values, the full flow curve is obtained. Repeating this procedure for multiple starting points s_0, the set of flow curves is generated. The maximum curvature points on these curves and the corresponding curvature values constitute our helper data. Fig. 5.6 shows the block diagram of helper data extraction, while Fig. 5.7 shows the input and output of the process, along with the intermediate steps. The final helper data are shown as blue points in Fig. 5.7.

The fingerprint images are from the DB2 of FVC 2002 study [32], with size 560x296 pixels, and 569 DPI resolution. Note that for automatic alignment-based vault experiments, we will be using this database with fully automatic processing (cf. ground truth minutiae information utilized before), including minutiae extraction, alignment, and vault decoding.

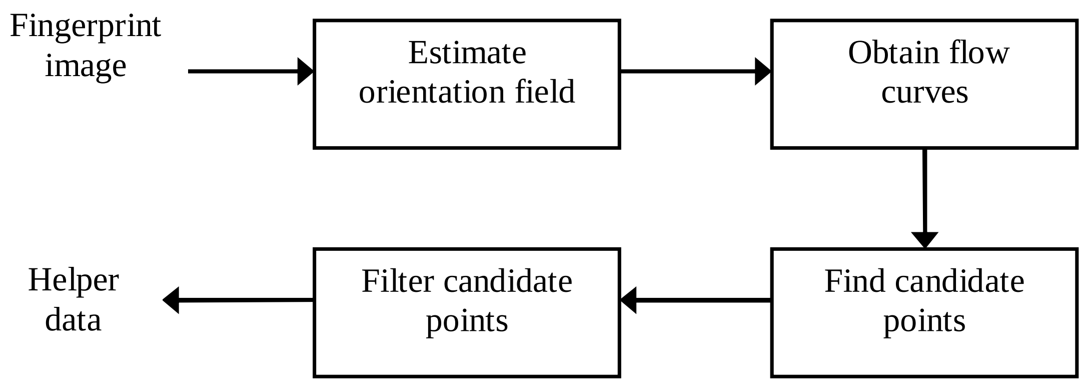


Figure 5.6: Orientation Field Flow Curves (OFFC) based helper data extraction flowchart.

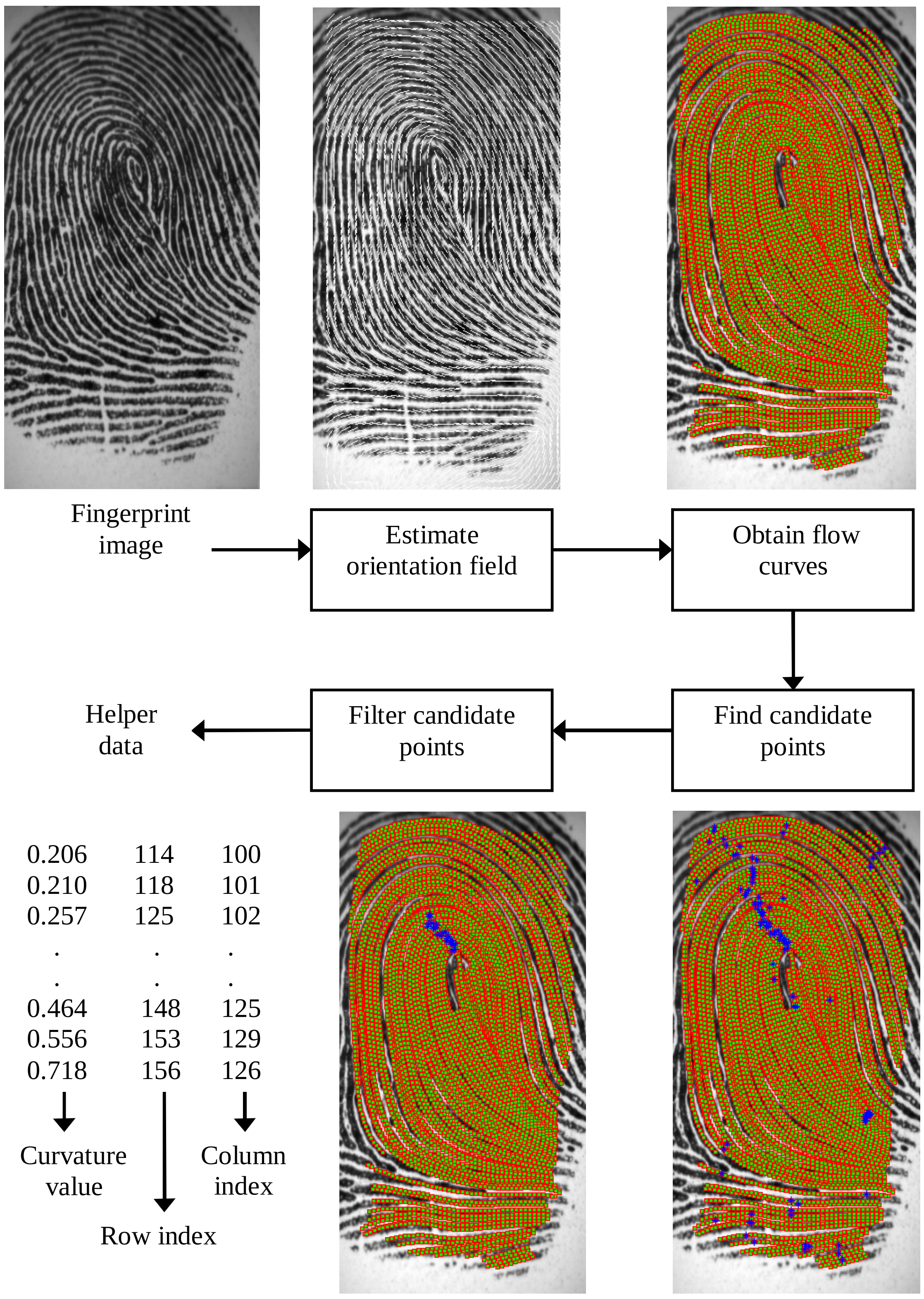


Figure 5.7: Path from fingerprint image to helper data.

5.7.1 Curvature Estimation for OFFC

The set of points on the piece-wise linear OFF curves are used to find the maximum curvature points locations and the actual curvature values. Namely, given a curve $OFFC_1$, (i) the curvature value is calculated for every point on the curve, (ii) the point with the maximum curvature is identified, and (iii) its location and curvature value is added to the corresponding candidate helper data CH (Fig. 5.7).

The underlying curvature estimation is based on finding the angles subtended by multiple (5) neighbors of a point p, and a linearly weighted cosine of these angles (Fig. 5.8):

Find_curvature($OFFC_1$*)*

Input: $OFFC_1$

Output: Curvatures for all points $p \in OFFC_1$

Step 1: For a point p, calculate the angles subtended by its 1-neighbors, 2-neighbors, 3-neighbors, 4-neighbors, and 5-neighbors (a_1, a_2, a_3, a_4, a_5)

Step 2: Calculate the sub-curvature as

$$CA_n = \sqrt{\frac{1 + cos(a_n)}{2}}, \quad n \in \{1, 2, 3, 4, 5\} \tag{5.16}$$

Step 3: Calculate curvature at p as

$$C = \frac{[CA_1 \;\; CA_2 \;\; CA_3 \;\; CA_4 \;\; CA_5].[w_1 \;\; w_2 \;\; w_3 \;\; w_4 \;\; w_5]^t}{(w_1 + w_2 + w_3 + w_4 + w_5)} \tag{5.17}$$

.

The weight vector used is

$$[w_1 \;\; w_2 \;\; w_3 \;\; w_4 \;\; w_5]^t = [1.0 \;\; 1.5 \;\; 2.0 \;\; 2.5 \;\; 3.0]^t \tag{5.18}$$

.

As a result, more weight is given to points that are far from point p in the curvature calculation. Note that $0 \leq C \leq 1$ for all points in the OFF curve. The point with the maximum curvature is added to candidate helper data CH for all OFF curves.

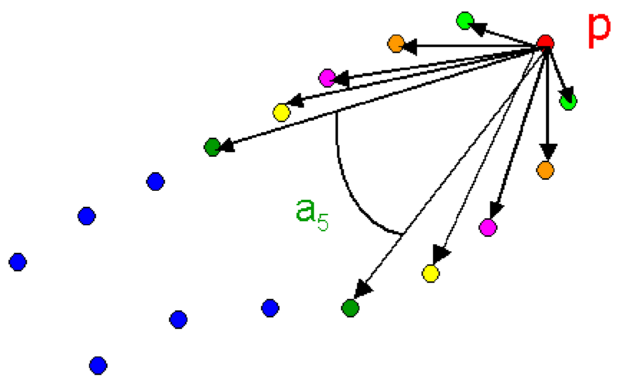


Figure 5.8: Neighbors of point p.

5.7.2 Helper Data Filtering

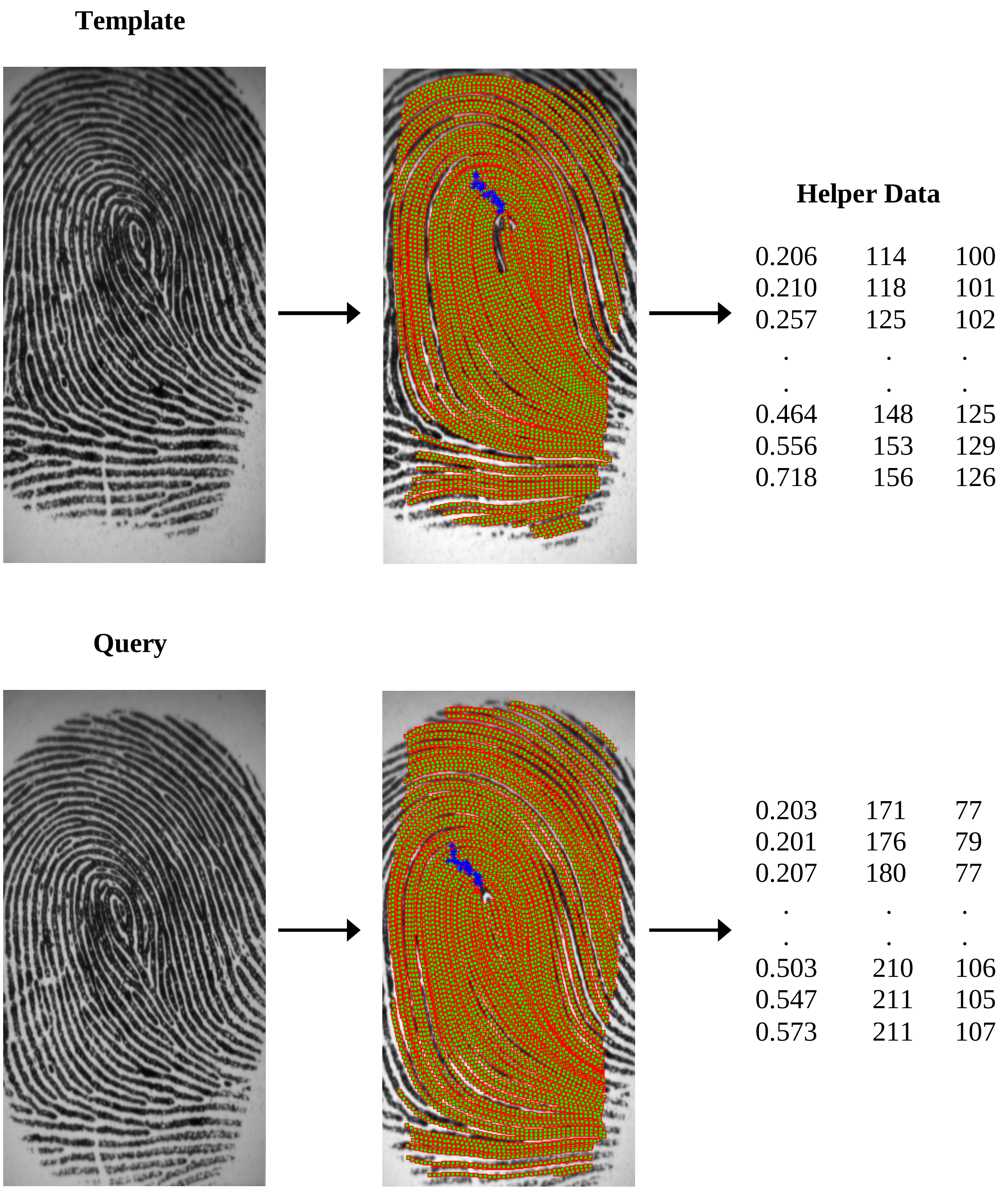
The candidate helper data CH that is output by the curvature estimation routine given above can contain outlier points (due to inexact localization of maximum curvature point). Furthermore, points with very low (corresponding to points on curves that are nearly flat) and very high curvature (corresponding to points that are too spiky) should be filtered. This filtering operation results in the final helper data, H (Fig. 5.7) that is either stored with the vault (H_E) or utilized during vault decoding (H_Q):

```
*Filter_candidates(CH)*
Input: $CH$

Output: $H$

for (point $p$, $p \in CH$)
    if($C \leq 0.2$ or $C \geq 0.8$ or ($p$ is further from 15 pixels from its neighbors))
        do not use $p$
    elseif
        keep $p$
    endif
endfor
```

Fig. 5.9 shows the helper data extraction for sample template and query fingerprints.



0.206	114	100
0.210	118	101
0.257	125	102
.	.	.
.	.	.
0.464	148	125
0.556	153	129
0.718	156	126

0.203	171	77
0.201	176	79
0.207	180	77
.	.	.
.	.	.
0.503	210	106
0.547	211	105
0.573	211	107

Figure 5.9: Extracting helper data from enrollment template (during vault encoding) and from query (during vault decoding).

5.8 Iterative Closest Point (ICP) based Alignment

The helper data that are extracted as explained above are used in the fuzzy fingerprint vault system as follows:

(i) Vault encoding: the helper data, H_E , associated with the enrollment fingerprint E (that is used for vault encoding) is saved along with the vault, V.

(ii) Vault decoding: the helper data, H_Q , extracted from the query fingerprint and H_E are used as inputs to the Iterative Closest Point (ICP) based alignment routine summarized below. As output, $T_{Q/E}$, which is a representation of the query minutiae template T_Q aligned with respect to the enrollment template T_E , is generated and used for vault decoding.

The feature-weighted ICP algorithm [58] employed in this thesis is a modification of the generic ICP method of Besl and McKay [3] (the details are given in Section 5.8.1 below):

ICP_align(H_E, H_Q, T_Q, α*)*

Input: H_E, H_Q, T_Q, α

Output: $T_{Q/E}$

Step 1: Estimate the initial transformation between H_Q and H_E: The center of mass of maximum curvature points in H_Q is translated so that it coincides with the center of mass of points in H_E. Translate T_Q accordingly.

Step 2: Iterate until convergence (or a maximum number of iterations (MAX_ITER) is reached)

- Compute the set of correspondences between points in H_Q and H_E:
  Find the distance between a point in H_E (ie. $p_E^i = (r_E^i, c_E^i, C_E^i)$, denoting the row index, column index and the curvature value for a maximum curvature point, respectively) and a point in H_Q (ie. $p_Q^i = (r_Q^i, c_Q^i, C_Q^i)$) as

$$dist(p_E^i, p_Q^i) = \sqrt{(r_E^i - r_Q^i)^2 + (c_E^i - c_Q^i)^2} + \alpha * abs(C_E^i - C_Q^i) \qquad (5.19)$$

  where the weight parameter α modulates the contribution of curvature based distance (second term) with respect to the Euclidean distance (first term).

- Compute the transformation that minimizes the mean square error between paired points. Apply the transformation to H_Q and T_Q. Use T_Q during vault decoding.

We have used the mean distance between corresponding points ($MEAN\_DIST$) as the metric for convergence: if $MEAN\_DIST$ does not change between successive iterations, or if the number of iterations exceeds $MAX\_ITER = 200$, the algorithm terminates.

5.8.1 Minimizing the Objective Function in ICP

Here, we summarize the original formulation of Besl and McKay [3] for finding the transformation to minimize the objective function.

The unit quaternion is a 4-tuple

$$\vec{q}_R = [q_0 \;\; q_1 \;\; q_2 \;\; q_3]^t \tag{5.20}$$

, where $q_0 \geq 0$ and $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. The rotation matrix $\mathbf{R}$ corresponding to $\vec{q}_R$ is

$$\mathbf{R} = \left| \begin{array}{ccc} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q0q3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{array} \right| \tag{5.21}$$

.

Denoting the translation vector $\vec{q}_T$ as

$$\vec{q}_T = [q_4 \;\; q_5 \;\; q_6]^t \tag{5.22}$$

, the final transformation (composed of rotation and translation) vector can be written as $\vec{q} = [\vec{q}_R | \vec{q}_T]^t$. Let $P = \{\vec{p}_i\}$ and $X = \{\vec{x}_i\}$ denote the point sets to be aligned. Note that for clarity it is assumed that P and X are ordered such that each point p_i corresponds to x_i with the identical index, and the set cardinality is the same, namely, $N_x = N_p$. The objective function to be minimized is

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p} \|\vec{x}_i - R(\vec{q}_R)\vec{p}_i - \vec{q}_T\|^2 \tag{5.23}$$

.

The centers of mass for point and model sets are given by

$$\vec{\mu}_p = \frac{1}{N_p}\sum_{i=1}^{N_p} \vec{p}_i \tag{5.24}$$

and

$$\vec{\mu}_x = \frac{1}{N_x}\sum_{i=1}^{N_x} \vec{x}_i \tag{5.25}$$

, respectively. Then, the cross-covariance matrix Σ_{px} is

$$\Sigma_{px} = \frac{1}{N_p}\sum_{i=1}^{N_p} [(\vec{p}_i - \vec{\mu}_p)(\vec{x}_i - \vec{\mu}_x)^t] \tag{5.26}$$

.

The cyclic components of the anti-symmetric matrix $A = (\Sigma_{px} - \Sigma_{px}^t)$ forms the column vector $\Delta = [A_{23} \;\; A_{31} \;\; A_{12}]^t$, which in turn generates the symmetric 4x4 matrix $Q(\Sigma_{px})$ as

$$\mathbf{R} = \begin{vmatrix} tr(\Sigma_{px}) & \Delta^t \\ \Delta & \Sigma_{px} + \Sigma_{px}^t - tr(\Sigma_{px})\mathbf{I_3} \end{vmatrix} \tag{5.27}$$

, where $\mathbf{I_3}$ is the identity matrix. The unit eigenvector corresponding to the maximum eigenvalue of the matrix $Q(\Sigma_{px})$, namely

$$\vec{q}_R = [q_0 \;\; q_1 \;\; q_2 \;\; q_3]^t \tag{5.28}$$

determines the optimal rotation matrix **R** using Eq. 5.21. Finally, the optimal translation vector is found by

$$\vec{q}_T = \vec{\mu}_x - R(\vec{q}_R)\vec{\mu}_p \tag{5.29}$$

which completes the calculation of total transformation composed of rotation and translation.

5.9 Multiple ICP Applications and Helper Data Clustering

As explained above, the parameter α in Eq. 5.19 modulates the contribution of curvature-based distance with respect to the Euclidean distance in the ICP algorithm: higher α values increase the effect of curvature (hence, more appropriate when the Euclidean distances are unreliable, e.g., due to noisy helper data points). Accordingly, no single α value is found to be optimal for all the fingerprints. Hence, our algorithm uses multiple (3 in the current implementation) α values ($\alpha \in 100, 150, 400$) successively, if the vault can not be opened with the chosen α value.

We often observe that the helper data has two strong clusters (one above the core, one below the core). This is especially true for the whorl class of fingerprints. Applying ICP to the conglomerate data is not logical when one of clusters is missing from either enrollment or template helper data. This clustering is done using the Euclidean distances between neighboring points in the helper data set: if this distance distance between two points is larger than 30 pixels, a new cluster is formed. So, now the ICP algorithm is applied to pairs of individual clusters in helper data H_Q and H_E.

This modified ICP-based decoding routine is given below, where S is the secret key, extracted from vault V:

```
*Vault_decode($H_E$, $H_Q$, $T_Q$, $V$)*
Input: $H_E$, $H_Q$, $T_Q$, $V$

Output: $S$ or $NULL$

for ($\alpha \in \{100, 150, 400\}$)
    for (cluster $CL_E$ in $H_E$)
        for (cluster $CL_Q$ in $H_Q$)
            Call ICP: *ICP_align($CL_E$, $CL_Q$, $T_Q$, $\alpha$)*
            if (vault opens)
                output $S$, exit
            elseif
                continue
            endif
        endfor
    endfor
endfor
```

Fig. 5.10 shows this alignment process pictorially for the previously presented template and query fingerprints in Fig. 5.9.

5.10 Experiments and Results: Automatically Aligned Templates

In this section, we present the vault performance when the aforementioned alignment scheme is employed. Note that previously presented results (pertaining to pre-aligned templates when the minutiae correspondances are known) can be thought of as part of an ideal (perfect) but unrealistic environment. As a result, the realistic scenario employed henceforth (namely, automatic alignment of templates using the helper data) is more applicable, but its performance is bounded from above by that of the ideal scenario.

We have used DB2 database of FVC 2002 study [32], with image sizes of 560x296 pixels, and 569 DPI resolution (sample images are shown above). It contains 8 impressions for each of 100 distinct fingers. For vault processing, a block size of 11 pixels (to accommodate imperfect -automatic- alignment and the increase in fingerprint resolution), and 24 genuine points (to account for imperfect -automatic- minutiae extractor) dispersed among 200 chaff points are used. All other parameters are the same as given

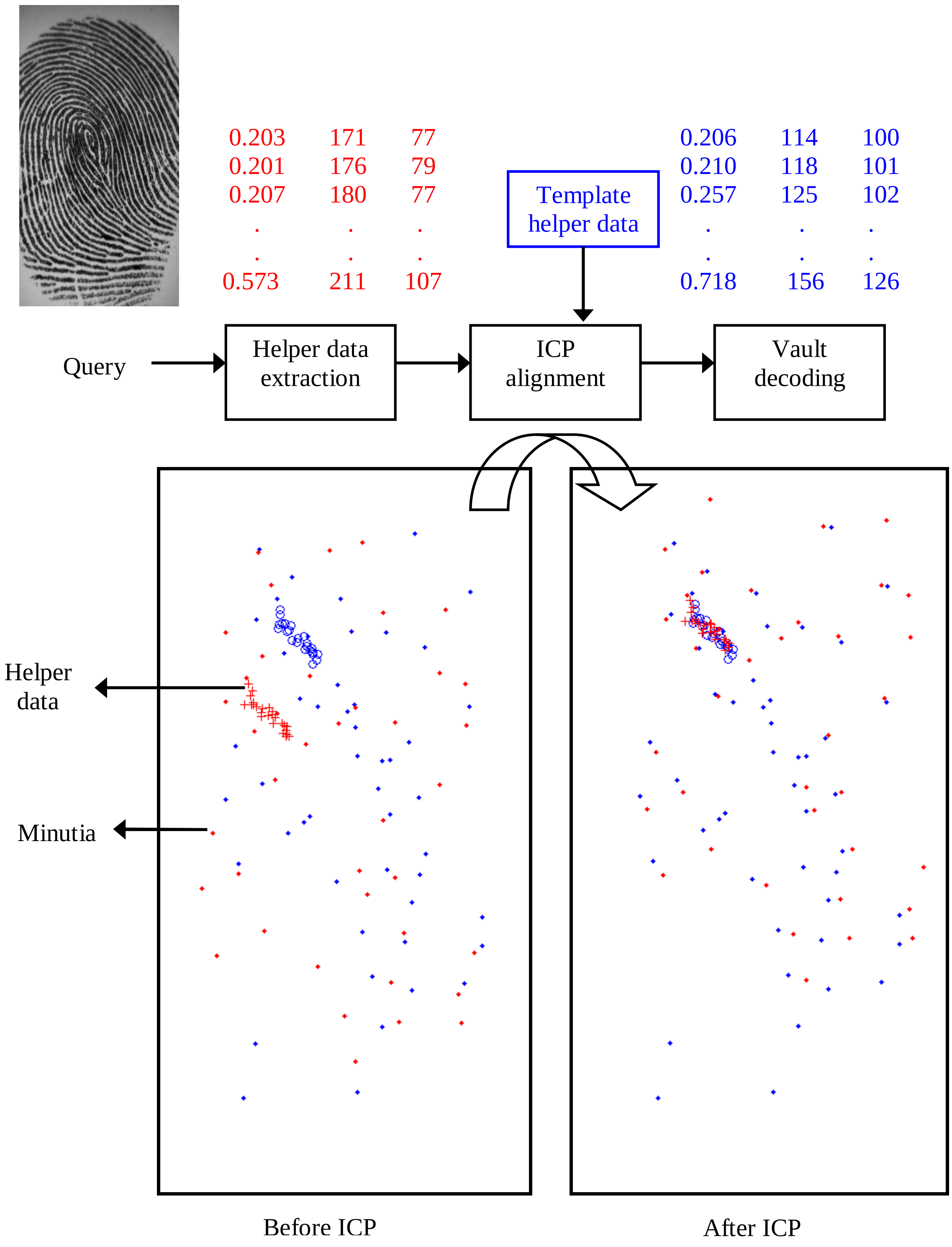


Figure 5.10: Aligning query using helper data.

before.

When 2 impressions per finger (impression number 1 for locking the vault, and impression number 2 for unlocking the vault) are used, with the above proposed automatic alignment scheme, the Genuine Accept Rate (GAR) is found to be 72.6% with no false accepts, e.g, FAR = 0%. Note that 16% of the fingers was rejected since they did not have enough (24) minutiae for locking or unlocking the vault. Like before, genuine access attempts occurred when the fingerprint impressions locking and unlocking the vault were from the same finger; false access attempts occurred when they were from different fingers. Using two impressions (impression numbers 2 and 7 in DB2) for decoding the vault improves the recognition performance; the GAR is increased to 84.4%, with no false accepts. The false rejects were due to (i) errors in helper data (7 fingers), (ii) poor quality images where reliable helper data could not be extracted (4 fingers), and (iii) number of common minutiae between locking and unlocking prints is less than the required number (2 fingers). Note that majority of these errors can be eliminated with increased user cooperation and habituation. We must also point out that the failure to enroll rate for fingerprint biometric can be as high as 4% [34]: namely, nearly 4% of the population may not generate acceptable fingerprint images (either during vault encoding or decoding) with current sensors. Hence, the figures that we cite above implicitly assume that acceptable fingerprints can be acquired (ie., for approximately 96% of the population).

The proposed helper data does not leak *any* information about the minutiae information used for locking and unlocking the vault. The helper data is extracted from the global flow of orientation field. On the other hand, the minutiae are local the discontinuities in the structure of ridges, and there is no way to learn minutiae features from the public helper data (cf. Fig. 5.10 shows the helper data overlaid with the critical minutiae information).

5.11 Summary

After exploring the characteristics of the cryptographic construct called fuzzy vault, we presented the results of its actual implementation using fingerprint minutiae data, without resorting to simulating the error-correction step. The vault performs as expected (namely, the genuine users can unlock the vault with acceptable accuracy, and the complexity of attacks that can be launched by imposter users is high). It is shown that 128-bit AES keys can be feasibly secured using the proposed architecture, and the helper data based alignment scheme performs reasonably well, considering the small amount of data used for alignment and the criteria for not leaking critical information outside the vault.

The proposed helper data, based on orientation field of fingerprints, is robust to several noise sources that can affect image quality (e.g., scars, bruises). Further, utilizing maximum curvature information (invariant to translation and rotation of fingerprints) of the curves originating from orientation field inside a variant of Iterative Closest Point (ICP) based alignment routine achieves reasonable alignment accuracy.

Chapter 6

Conclusions and Future Work

6.1 Research Contributions

Security analysis of biometric systems is an important yet challenging problem, due to the complexity of the algorithms in such systems (e.g., feature extractor and matcher). As an increasing number of these systems are being deployed in both commercial and government applications, a thorough analysis of this challenge becomes necessary. In this thesis, we address four specific instances of this problem:

Firstly, a hill-climbing based attack system against fingerprint matchers was developed. Utilizing the matching score to gradually improve synthetic templates, it was able to effectively bypass the security afforded by a minutiae-based fingerprint verification system. Utilization of fingerprint class prior probabilities, spatial minutiae presence probabilities, and class-based orientation fields contributed to the effectiveness of the attacker by decreasing the number of required access attempts to reach templates that mimic the actual target templates. We proposed a score masking procedure to decrease the feasibility of this attack: by eliminating the correlation between the controlled changes the attacker introduces to the synthetic templates and the returned matching scores, it is shown that a valid minutiae template can not be synthesized.

Secondly, we proposed watermarking techniques to increase the security of biometric templates: the associated application scenarios included: (i) transferring fingerprint templates over an insecure communication channel and (ii) increasing the security of image-based templates (e.g., fingerprint, face) by actually embedding one biometric into another. The proposed spatial-domain technique included image adaptivity and host image analysis to modulate the amount of change due to watermarking. We were able to extract the embedded information with 100% accuracy. Further, it is shown that the fingerprint verification accuracy is preserved with the proposed system.

Thirdly, fingerprint matching is integrated into a multimedia content distribution framework. Based on layered encryption/decryption using fingerprints as keys, the proposed scheme eliminates the feasibility of illegal key sharing, which hampers the content protection schemes based solely on traditional keys. To accommodate the intra-class variability of fingerprint templates, we devised a novel file transfer scheme that allowed us to generate repeatable keys. Utilizing widely available cryptographic modules such as AES and MD5, it was shown that multimedia files can be feasibly secured using the proposed system.

Finally, a biometric cryptosystem making use of fingerprints was developed. Merging the advantages of biometrics (user-friendliness) and cryptography (theoretical and practical security), such systems are shown to be good candidates for addressing the security and privacy issues related to biometric templates. Based on the cryptographic construct called fuzzy vault, we developed techniques to successfully contain 128-bit secrets using fingerprint minutiae. It was shown that helper data based alignment scheme performs well, considering the small amount of data used for alignment and the criteria for not leaking critical information outside the vault. The proposed helper data based on orientation field of fingerprints is immune to several noise sources that can affect image quality. Further, utilizing maximum curvature information (invariant to translation and rotation of fingerprints) of the curves originating from orientation field inside a variant of Iterative Closest Point (ICP) based alignment routine achieved reasonable alignment accuracy.

6.2 Future Research

We conclude this thesis by elaborating on possible techniques that can be used to expand the research presented here:

1. The hill-climbing based attack system was shown to be quite feasible. It was able to synthesize the templates that mimic the real-world templates closely. The attack system can be improved to decrease the number of required access attempts even further. For example, the co-occurrence probabilities of minutiae that are spread over the entire image plane can be used. This may help in synthesizing the template faster since, with the incorporation of every minutia into the synthesis, the probability densities for all successive minutiae change.

Further, the proposed system can act as a precursor to the fingerprint *image* synthesis algorithms (e.g., Ross et al.'s [54]) that aim to generate fingerprint image, *given* the secret minutiae template. Our algorithm can synthesize that minutiae template, and the aforementioned algorithm can create the corresponding fingerprint image.

2. In order to address illegitimate template modifications, the current watermarking scheme can be combined with a fragile scheme: a fragile scheme can detect and localize changes to the template. The proposed watermarking algorithm functions as the carrier of watermark data, such as the template origin. Further, as with any watermarking scheme, encryption can be used on the watermarked templates to increase the security even further.

3. The proposed multimedia content distribution framework can benefit from cryptographic architectures designed solely for multimedia data. Such architectures (though less secure than the currently used AES symmetric system) make use of characteristics of multimedia data, such as redundancy, to encrypt only parts of the data. As a result, encryption and decryption times can be decreased considerably. Also, the proposed framework can be combined with a watermarking scheme for multimedia data. As a result, the proposed encryption routine can guarantee security until the multimedia data is played, and watermarking can provide security even when the data is played (e.g., by eliminating the possibility of a pirate recording the video of a movie played on the monitor).

4. Incorporating error-correction schemes into the fingerprint-based fuzzy vault

system has the potential to decrease the vault decoding times. However, methods to achieve this within the critical automatic alignment methodology is a challenge. Errors originating from chaff points and errors originating from imperfect alignments need to be taken care of individually, but there is no way to label the errors without accessing the original database template.

New helper data constructs can be used for alignment purposes, for example, ridge curves can assist the maximum curvature-based current scheme. As long as the helper data can be guaranteed to not leak critical information about the vault (or the associated fingerprint), using additional data will increase the alignment accuracy, hence increasing vault decoding performance (but with possible size limitations for resource constrained devices such as smart cards). Even simple alignment schemes (e.g., based on core points) can work in tandem with the proposed scheme when applicable (e.g., when a core can be localized with high fidelity).

The fundamental problem of biometric cryptosystems, namely, how to perform biometric matching when the biometric templates are transformed into the cryptographic domain that *does not* allow traditional biometric operations (e.g., filtering, enhancement, alignment, matching) will continue to be a challenging problem for the years to come. However, improvements in accuracy (via cryptographic architectures designed with biometric data considerations) and speed (hardware implementation of algorithms) are needed.

Bibliography

Bibliography

[1] A. Adler. Sample images can be independently restored from face recognition templates. Available at `http://www.site.uottawa.ca/~adler/publications/2003/adler-2003-fr-templates.pdf`, 2003.

[2] I. Armstrong. Passwords exposed: users are the weakest link . Available at `http://www.safestone.com/downloads/news/news_passwords_exposed_sc_magazine_may03.pdf`, 2003.

[3] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[4] R. M. Bolle, N. K. Ratha, A. Senior, and S. Pankanti. Minutiae template exchange format. In *Proc. AutoID 1999, IEEE Workshop on Automatic Identification Advanced Technologies*, pages 74–77, 1999.

[5] R. Cappelli, A. Erol, D. Maio, and D. Maltoni. Synthetic fingerprint image generation. In *Proc. International Conference on Pattern Recognition (ICPR), vol. 3*, pages 475–478, 2000.

[6] T. C. Clancy, N. Kiyavash, and D. J. Lin. Secure smartcard-based fingerprint authentication. In *Proc. ACM SIGMM Multimedia, Biometrics Methods and Applications Workshop*, pages 45–52, 2003.

[7] Colorado State University. Evaluation of face recognition algorithms. Available at `www.cs.colostate.edu/evalfacerec/index.html`.

[8] Congress of the United States of America. Enhanced Border Security and Visa Entry Reform Act of 2002. Available at `http://unitedstatesvisas.gov/pdfs/Enhanced_Border_SecurityandVisa_Entry.pdf`, 2002.

[9] S. Dass and A. K. Jain. Fingerprint classification using orientation field flow curves. In *Proc. Indian Conference on Computer Vision, Graphics and Image Processing*, pages 650–655, 2004.

[10] S. C. Dass. Markov random field models for directional field and singularity extraction in fingerprint images. *IEEE Transactions on Image Processing*, 13(10):1358–1367, October 2004.

[11] J. G. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1148–1161, November 1993.

[12] G. I. Davida, Y. Frankel, and B. J. Matt. On enabling secure applications through off-line biometric identification. In *Proc. 1998 IEEE Symposium on Privacy and Security*, pages 148–157, 1998.

[13] G. I. Davida, Y. Frankel, B. J. Matt, and R. Peralta. On the relation of error correction and cryptography to an offline biometric based identification scheme. In *Proc. Workshop on Coding and Cryptography (WCC)*, pages 129–138, 1999.

[14] R. Derakhshani, S. A. C. Schuckers, L. A. Hornak, and L. O. Gorman. Determination of vitality from a non-invasive biomedical measurement for use in fingerprint scanners. *Pattern Recognition*, 36(2):383–396, February 2003.

[15] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In *Proc. International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 523–540, 2004.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, New York, Second edition, 2001.

[17] B. Gunsel, S. Sener, and Y. Yaslan. An adaptive encoder for audio watermarking. *WSEAS Transactions on Computers*, 2(4):1044–1048, October 2003.

[18] B. Gunsel, U. Uludag, and A. M. Tekalp. Robust watermarking of fingerprint images. *Pattern Recognition*, 35(12):2739–2747, December 2002.

[19] F. Hartung and M. Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7):1079–1107, July 1999.

[20] C. J. Hill. Risk of masquerade arising from the storage of biometrics. Available at `http://chris.fornax.net/biometrics.html`, 2001.

[21] Institute of Paper Science and Technology, Georgia Institute of Technology. Watermarks. Available at `http://www.ipst.gatech.edu/amp/education/watermark/watermarks.htm`.

[22] A. K. Jain, R. Bolle, and S. Pankanti, editors. *Biometrics: Personal Identification in Networked Society.* Kluwer Academic Publishers, New York, 1999.

[23] A. K. Jain, L. Hong, S. Pankanti, and R. Bolle. An identity authentication system using fingerprints. *Proceedings of the IEEE*, 85(9):1365–1388, September 1997.

[24] A. K. Jain and U. Uludag. Hiding biometric data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1494–1498, November 2003.

[25] A. Juels and M. Sudan. A fuzzy vault scheme. In A. Lapidoth and E. Teletar, editors, *Proc. IEEE International Symposium on Information Theory*, page 408, 2002.

[26] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In G. Tsudik, editor, *Proc. Sixth ACM Conference on Computer and Communications Security*, pages 28–36, 1999.

[27] D. V. Klein. Foiling the cracker: a survey of, and improvements to Unix password security. In *Proc. United Kingdom Unix User's Group*, 1990.

[28] M. Kutter, F. Jordan, and F. Bossen. Digital signature of color images using amplitude modulation. In *Proc. SPIE, Storage and Retrieval for Image and Video Databases V, vol. 3022*, pages 518–526, 1997.

[29] E. T. Lin and E. J. Delp. A review of fragile image watermarks. In *Proc. ACM Multimedia and Security Workshop*, pages 25–29, 1999.

[30] S. Lin and D. J. Costello. *Error Control Coding.* Pearson Prentice Hall, New Jersey, Second edition, 2004.

[31] J.-P. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *Proc. 4th International Conference on Audio- and Video-based Biometric Person Authentication*, pages 393–402, 2003.

[32] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain. FVC2002: Second Fingerprint Verification Competition. In *Proc. International Conference on Pattern Recognition*, pages 811–814, 2002.

[33] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain. FVC2004: Third Fingerprint Verification Competition. In *Proc. International Conference on Biometric Authentication (ICBA)*, pages 1–7, 2004.

[34] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition.* Springer, New York, 2003.

[35] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. Impact of artificial gummy fingers on fingerprint systems. In *Proc. of SPIE, Optical Security and Counterfeit Deterrence Techniques IV, vol. 4677*, pages 275–289, 2002.

[36] N. Memon and P. W. Wong. Protecting digital media content. *Communications of the ACM*, 41(7):34–43, July 1998.

[37] F. Mintzer, J. Lotspiech, and N. Morimoto. Safeguarding digital library contents and users. Available at `http://www.dlib.org/dlib/december97/ibm/12lotspiech.html`.

[38] F. Monrose, M. K. Reiter, Q. Li, D. P. Lopresti, and C. Shih. Towards speech-generated cryptographic keys on resource constrained devices. In *Proc. 11th USENIX Security Symposium*, pages 283–296, 2002.

[39] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. Cryptographic key generation from voice. In *Proc. IEEE Symposium on Security and Privacy*, pages 202–213, 2001.

[40] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. Using voice to generate cryptographic keys. In *Proc. 2001: A Speaker Odyssey, The Speaker Recognition Workshop*, pages 237–242, 2001.

[41] F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *Proc. 6th ACM Conference on Computer and Communications Security*, pages 73–82, 1999.

[42] National Institute of Standards and Technology. NIST Special Database 4, 8-Bit Gray Scale Images of Fingerprint Image Groups (FIGS). Available at `http://www.nist.gov/srd/nistsd4.htm`.

[43] National Institute of Standards and Technology. Data Encryption Standard (DES). Available at `http://www.itl.nist.gov/fipspubs/fip46-2.htm`, 1993.

[44] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Available at `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`, 2001.

[45] NIST. Fingerprint SDK (Software Development Kit) Testing, 2005. Available at `http://fingerprint.nist.gov/SDK/sdk.html`.

[46] NIST. Fingerprint Vendor Technology Evaluation (FpVTE), 2003. Available at `http://fpvte.nist.gov/`.

[47] S. Pankanti, S. Prabhakar, and A. K. Jain. On the individuality of fingerprints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1010–1025, August 2002.

[48] S. Pankanti and M. M. Yeung. Verification watermarks on fingerprint recognition and retrieval. In *Proc. SPIE, Security and Watermarking of Multimedia Contents, vol. 3657*, pages 66–78, 1999.

[49] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, Second edition, 1992.

[50] T. Putte and J. Keuning. Biometrical fingerprint recognition: don't get your fingers burned. In *Proc. IFIP TC8/WG8.8, Fourth Working Conference on Smart Card Research and Advanced Applications*, pages 289–303, 2000.

[51] N. K. Ratha, J. H. Connell, and R. M. Bolle. Secure data hiding in wavelet compressed fingerprint images. In *Proc. ACM Multimedia*, pages 127–130, 2000.

[52] N. K. Ratha, J. H. Connell, and R. M. Bolle. An analysis of minutiae matching strength. In *Proc. AVBPA 2001, Third International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 223–228, 2001.

[53] N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, 2001.

[54] A. Ross, J. Shah, and A. K. Jain. Generating fingerprints from minutiae using streamlines. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (submitted)*, 2006.

[55] R. K. Rowe, S. P. Corcoran, and K. Nixon. Biometric identity determination using skin spectroscopy. Available at `http://www.lumidigm.com/skiinmorph.html`.

[56] B. Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C.* John Wiley & Sons, New York, Second edition, 1996.

[57] B. Schneier. The uses and abuses of biometrics. *Communications of the ACM*, 42(8):136, August 1999.

[58] G.C. Sharp, S.W. Lee, and D.K. Wehe. ICP registration using invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):90–102, January 2002.

[59] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. K. Jain. Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):450–455, March 2005.

[60] C. Soutar. Biometric system security. Available at `http://www.bioscrypt.com/assets/security_soutar.pdf`.

[61] C. Soutar, D. Roberge, S. A. Stojanov, R. Gilroy, and B. V. K. Vijaya Kumar. Biometric encryption - enrollment and verification procedures. In *Proc. SPIE, Optical Pattern Recognition IX, vol. 3386*, pages 24–35, 1998.

[62] C. Soutar, D. Roberge, S. A. Stojanov, R. Gilroy, and B. V. K. Vijaya Kumar. Biometric encryption using image processing. In *Proc. SPIE, Optical Security and Counterfeit Deterrence Techniques II, vol. 3314*, pages 178–188, 1998.

[63] C. Soutar, D. Roberge, S. A. Stojanov, R. Gilroy, and B. V. K. Vijaya Kumar. Biometric encryption. In R. K. Nichols, editor, *ICSA Guide to Cryptography.* McGraw Hill, 1999.

[64] W. Stallings. *Cryptography and Network Security: Principles and Practices.* Prentice Hall, New York, Third edition, 2003.

[65] M. D. Swanson, M. Kobayashi, and A. H. Tewfik. Multimedia data-embedding and watermarking technologies. *Proceedings of the IEEE*, 86(6):1064–1087, June 1998.

[66] U. Uludag and A. K. Jain. Multimedia content protection via biometrics-based encryption. In *Proc. IEEE Conference on Multimedia & Expo (ICME)*, pages 237–240, 2003.

[67] U. Uludag and A. K. Jain. Attacks on biometric systems: a case study in fingerprints. In *Proc. SPIE, Security, Steganography and Watermarking of Multimedia Contents VI, vol. 5306*, pages 622–633, 2004.

[68] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain. Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, 92(6):948–960, June 2004.

[69] United States Copyright Office. The Digital Millennium Copyright Act. Available at `http://www.copyright.gov/legislation/dmca.pdf`, 1998.

[70] USC Viterbi School of Engineering. The USC-SIPI Image Database. Available at `http://sipi.usc.edu/services/database/Database.html`.

[71] C. P. Wu and C. C. J. Kuo. Efficient multimedia encryption via entropy codec design. In *Proc. SPIE, vol. 4314*, pages 128–138, 2001.

[72] Y. Chen and A. Jain. Fingerprint deformation for spoof detection. Biometric Consortium Conference, 2005, available at `http://www.biometrics.org/bc2005/program.htm`.

[73] S. Yang and I. Verbauwhede. Automatic secure fingerprint verification system based on fuzzy vault scheme. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 609–612, 2005.