How to Attack Biometric Systems in Your Spare Time

Ahmed Obied
Department of Computer Science
University of Calgary
2500 University Drive N.W.
Calgary, Alberta, Canada T2N 1N4
obieda@cpsc.ucalgary.ca

ABSTRACT

Biometric systems were proposed and developed to provide a better and stronger factor of authentication. Such systems authenticate individuals based on physical and behavioral traits such as fingerprints, iris, face, palm print, hand geometry, voice, etc. The use of biometric traits to replace existing passwords or as access keys has proven to be highly secure against physical attacks. It is a fact that malicious attacks are getting smarter, more widespread and increasingly difficult to detect, and dozens more are added to the menagerie each day. Attacking biometric systems physically is quite difficult indeed. However, by mounting digital attacks one can see how biometric systems are vulnerable. We discuss the different types of software and hardware vulnerabilities that exist in biometric systems, and show how biometric template security can be compromised. We present a new attack point at the application level that has not been addressed and discussed in previous work. We also describe how biometric cryptosystems can overcome some of the disadvantages in traditional biometric recognition systems, and show how such systems can be used effectively in Digital Rights Management (DRM) systems.

Keywords

Biometrics, Templates, Security, Privacy, Cryptography, Attacks, Hash Functions, Digital Rights Management Systems

1. INTRODUCTION

In today's highly technical world, access to restricted computers, networks, services, areas, etc is usually done via the use a token (e.g., key) or knowledge (e.g., password). A password is something a user (or a group of users) knows and nobody elses, and a token is something a user (or a group of users) has and nobody does. Traditional authentication systems that are based on tokens or knowledge have been used for many years despite their disadvantages. How many times have you tried to login to your email account to read important email but the mail server refused to grant you access because you simply forgot the password? How many times have you tried to go to your apartment or house but you were not able to access it because you simply lost the keys? How many times you have been a victim of identify theft because some person was able to get your password which you wrote down somewhere or found your keys which you left somewhere to gather sensitive information about you and impersonate you? The list of questions can go on and on with the existing authentication schemes used in today's highly technical world. So what can we do to overcome these problems? The answer to this question is surprisingly simple: we use biometric technologies.

Almost everyone in the world have fingers, eyes, voice, hands, and a face. Have you ever forgotten any of those body parts? Not very likely! [8] What if we could use those body parts instead of tokens and passwords to authenticate you? Wouldn't that be more convenient and more secure? The answer is simply yes. Humans have used body characteristics such as face, voice, and gait for thousands of years to recognize each other [3] but only recently humans started developing and deploying biometric based systems to authenticate individuals. Biometrics is the science of establishing or determining an identity [4] based on the physical or behavioral traits of an individual such as fingerprints, iris, face, palm print, hand vein, voice, keystroke, retina, facial thermogram, etc. Biometric systems are essentially pattern recognition systems that read as input biometric data, extract a feature set from such data, and finally compare it with a template set stored in a database. If the extracted feature set from the given input is close to a template set stored in the database then the user is granted access.

The use of biometric systems has introduced a convenient, efficient, and secure alternative to traditional authentication schemes (e.g., token-based and knowledge-based). However, we must note that such systems are susceptible to various types of security attacks that are aimed at undermining the integrity of the authentica-

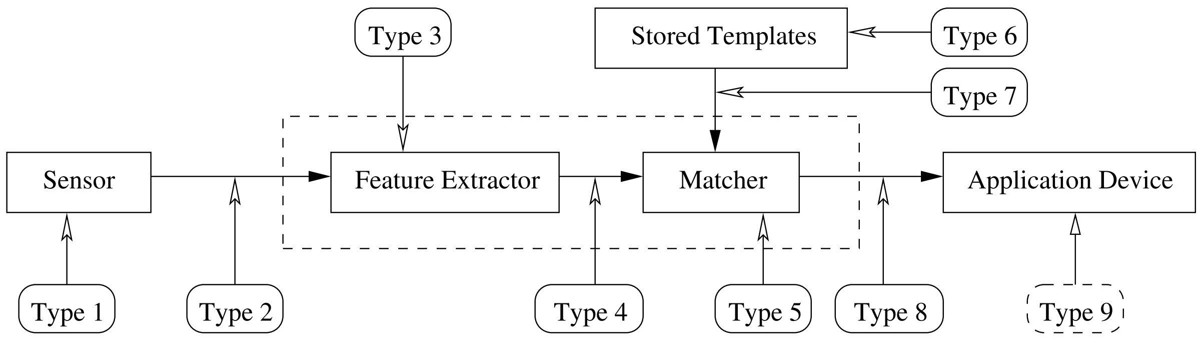


Figure 1: Biometric System and the nine different points of attack.

tion process by either circumventing the security afforded by the system or deterring the normal functioning of the system [4]. Such attacks are raising concerns as more and more biometric systems are being deployed and used in commercial, government and forensic applications. This, along with the increase size of the population using these systems and the expanding application area (visa, border control, health care, welfare distribution, e-commerce, etc) may lead to possible finance, privacy, and security breaches [12].

In this paper, we show an in-depth analysis of the different types of software and hardware vulnerabilities that exist in biometric systems, and describe how such vulnerabilities can be exploited. We present a new attack point at the application level of a biometric system. We also discuss the idea behind biometric cryptosystems and show how such systems can be effectively used in some applications.

2. BIOMETRIC SYSTEM MODEL

To fully understand how biometric systems can be attacked, we must first understand the biometric system model and the different modules it consists of. Figure 1 shows a generic biometric system and the eight different points of attack as proposed by [7]. We claim that a 9th point of attack exists at the application level. This claim will be discussed in section 4.1.9. Any biometric system consists of the following modules:

2.1 Scanner

The scanner module in a biometric system is used to scan and acquire the biometric data (e.g., fingerprint, hand vein, palm print, etc) of an individual in a form of an image, video, audio or some other signal. The scanner module is vulnerable to a type 1 attack which is discussed in section 4.1.1.

2.2 Feature Extractor

The feature extractor module in a biometric system operates on the signal sent by the scanner module to extract a feature set i that represents the given signal. The extracted feature set i is sent to the matcher for processing. The feature extractor module is vulnerable to a type 3 attack which is discussed in section 4.1.3.

2.3 Stored Templates

The stored templates module in a biometric system is usually a database that stores pre-acquired (usually during users enrolment) feature sets called templates. These templates are queried by the matcher module to find a match for a given feature set i. The database that stores the templates is vulnerable to a type 6 attack which is discussed in section 4.1.6.

2.4 Matcher

The matcher module in a biometric system is the main module in such system. The matcher receives a feature set i from the feature extractor module and compares i with the templates stored in the database. Match scores are generated after each comparison and once all comparisons are done, the matcher processes these match scores in order to either determine or verify the identify of an individual [4]. The matcher module is considered the main module in a biometric system because its the part that makes the decision ("yes" if there is a match or "no" if there is no match). The matcher is vulnerable to a type 5 attack which is discussed in section 4.1.5.

2.5 Application Device

The application device module in a biometric system receives an answer from the matcher and acts accordingly. If the application device receives a "no" then it denies access. If the application device receives a "yes" then it grants access. The application device is vulnerable to a type 9 attack as we claim.

3. GENERIC SECURITY THREATS

Any system (including biometric systems) are susceptible to various types of threats. These threats are discussed below:

3.1 Denial of Service

An adversary overwhelms computer and network resources to the point that legitimate users can no longer access the resources. For example, an adversary may

Manufacturer	Model	Technology	Date	Difficulty
Identix	TS-520	Optical	Nov. 1990	First attempt
Fingermatrix	Chekone	Optical	Mar. 1994	Second attempt
Dermalog	DemalogKey	Optical	Feb. 1996	First attempt
STMicroelectronics	TouchChip	Solid state	Mar. 1999	First attempt
Veridicon	FPS110	Solid stat	Sep. 1999	First attempt
Identicator	DFR200	Optical	Oct. 1999	First attempt

Table 1: Attack at the scanner (adapted from [7]).

exploit a weakness in the TCP/IP stack's 3-way handshake, and overwhelm an HTTP server with a large number of SYN packets with spoofed and unused IP addresses to start the first step in the 3-way handshake. When the HTTP server tries to reply with a SYN/ACK packets (second step in the 3-way handshake), the packets will be sent to the spoofed and unused IP addresses. Since the IP addresses are unused (technically they do not exist), the packets will be traversing the network for quite a while before any router in the Internet starts sending the HTTP server ICMP packets with the error message "host unreachable". This type of attack exhausts both the HTTP server and the network resources. If the adversary's packets are sent very fast and by large numbers, the HTTP server can go down and no longer reply to any connection requests from legitimate users.

3.2 Circumvention

An adversary gains access to data or computer resources that he may not be authorized to access. For example, if a network service (e.g., telnet) in some network N has a bug x (e.g., buffer overflow), an adversary can write an exploit for x to download (from some IP address on the Internet) and install a sniffer on the vulnerable machine to gather usernames and passwords. The adversary can use these usernames and passwords to access other computers on network N and find confidential data.

3.3 Repudiation

A legitimate user accesses the resources offered by an application and then claim that an intruder had circumvented the system. For example, a bank clerk may modify the financial records of a customer C and then deny responsibility by claiming that an intruder could have possibly stolen from C [4].

3.4 Covert acquisition

An adversary compromises and abuses the means of identification without the knowledge of a legitimate user. For example, a worker W_a can write his password on a piece of paper and leave it on his desk at work. His co-worker W_b finds the piece of paper and copies the password. Since W_b has the password of W_a then W_b can use it to access W_a's account.

3.5 Collusion

In any system, there are different user privileges. Users with super-user privileges have access to all of the system's resources. Collusion occurs when a user with super-user privileges abuses his privileges and modifies the system's parameters to permit incursions by an intruder. For example, a system administrator of a Linux server can open up a privileged port (between 1 and 1024) on the server to give an intruder a chance to access the server easily. Privileged ports are usually known as trusted ports so any traffic that is sent to a privileged port usually passes through firewalls without any checking. It is often mentioned that the easiest way to break a security system is to compromise the system administrator [5].

3.6 Coercion

A legitimate user is forced to give an intruder access to the system. For example, an ATM user could be forced to give away her ATM card and PIN at gunpoint [5].

4. BIOMETRIC SECURITY THREATS

Figure 1 shows a biometric system modules and nine different points of attack. These points of attack are discussed in detail below.

4.1 Points of Attack

4.1.1 Type 1

This point of attack is known as *"Attack at the scanner"*. In this attack, the attacker can physically destroy the recognition scanner and cause a denial of service as described in section 3.1. The attacker can also create a fake biometric trait such as an artificial finger to bypass fingerprint recognition systems, or inject an image between the sensing element and the rest of the scanner electronics to bypass facial recognition systems.

Putte and Keuning [6] mentioned that the problem with most of the fingerprint scanners used these days is distinguishing between a real finger and a well created artificial (dummy) finger. The authors tested several fingerprint scanners to check whether such scanners will accept the dummy finger. Two methods were used to duplicate a real finger: with and without the co-operation of the owner. With the co-operation of the owner, the authors first created a plaster cast of the finger and the cast is then filled with silicon rubber to create a wafer-thin silicon dummy. The authors mentioned that the dummy can be glued to anyone's finger without it being noticeable to the eye. Without the co-operation of the owner, it is necessary to obtain a print of the finger from a surface or a glass. The authors claim that every dental technician has the skills and equipment to create a dummy from a print of the finger. Putte and Keuning claim that since 1990 several fingerprint scanners have been tested using dummy fingers and all tested scanners accepted a dummy finger as a real finger. Table 1 shows in detail the scanner's manufacturer, model, technology, date on which the scanners have been tested, and finally the number of attempts required to get the dummy finger accepted.

4.1.2 Type 2

This point of attack is known as *"Attack on the channel between the scanner and the feature extractor"* or *"Replay attack"*. When the scanner module in a biometric system acquires a biometric trait, the scanner module sends it to the feature extractor module for processing. In this attack, the attacker intercepts the communication channel between the scanner and the feature extractor to steal biometric traits and store it somewhere. The attacker can then replay the stolen biometric traits to the feature extractor to bypass the scanner.

4.1.3 Type 3

This point of attack is known as *"Attack on the feature extractor module"*. In this attack, the attacker can replace the feature extractor module with a Trojan horse. A Trojan horse program, named after the wooden artifact from Greek methodology that contained more than could be seen on the surface, refers to an executable code that is not a translation of the original program but was added later, usually maliciously, and comes into the system disguised as the original program [5]. Trojan horses in general can be controlled remotely. Therefore, the attacker can simply send commands to the Trojan horse to send to the matcher module feature values selected by him.

4.1.4 Type 4

This point of attack is known as *"Attack on the channel between the feature extractor and matcher"*. This attack is similar to the attack described in section 4.1.2. The difference is that the attacker intercepts the communication channel between the feature extractor and the matcher to steal feature values of a legitimate user and replay them to the matcher at a later time.

An example of this type of attack would be what Adler [1] proposed. Adler presented an approach that expands on the idea of Hill [2] and uses a "Hill Climbing Attack". The author proposed an algorithm that regenerates sample images from templates using only match score results generated by the matcher module. Adler implemented a software that has access to a local database that contains facial images and a network access to 3 commercial face recognition servers to achieve his goals of synthetically reconstructing a face image. Adler's algorithm begins with only the database ID of the target person. The algorithm starts by selecting an initial image from the local database and sending the initial image feature values to the matcher module. The matcher module generates match scores and based on these match scores, the initial image is modified. In every step, several facial images are multiplied with a weight and added to the current candidate image. The modified image that generate the highest matching score is used as the new candidate image. This process is repeated several times until there is no major improvement in the matching score. Once this happens, the current candidate image will have a good resemblance to the target image.

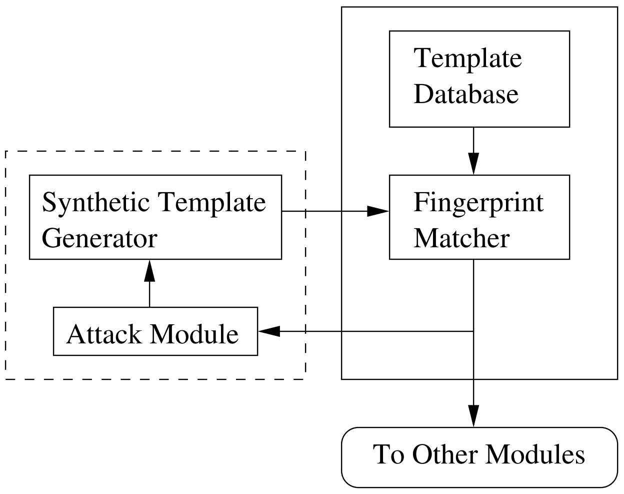


Figure 2: Attack System (adapted from [13])

A similar attack was proposed by Uludag and Jain [12]. Uludag and Jain implemented an attack system as shown in figure 2 for a minutiae-based fingerprint authentication system that uses a "Hill Climbing Attack". The attack system generates a minutiae set that results in a high matching score to gain access to the system in place of a legitimate user. Minutiae points consist of ridge endings and ridge bifurcations. All minutiae based systems use the 2D location (x, y) of the minutiae and the orientation θ associated with the minutiae as the attributes [12]. Uludag and Jain used the attributes of the minutiae (x, y, θ) in their attack system. To attack user's i account, Uludage and Jain's attack system does the following (Assuming that there is a decision threshold value used by the fingerprint matcher module $S_{threshold}$ that the attacking system does not know):

1. Generate a number of synthetic templates (T_i^1, T_i^2, T_i^3, ..., T_i^n) where n is fixed.

2. Send the generated templates in step 1 to the fingerprint matcher module and accumulate the corresponding matching scores ($S(D_i, T_i^1)$), $S(D_i, T_i^2)$, $S(D_i, T_i^3)$, ..., $S(D_i, T_i^n)$) where D_i is the template corresponding to user i in the database and $S(D_i, T_i^j)$ is the matching score between D_i and T_i^j.

3. Declare (T_i^{best}) to be the template resulting in the highest matching score and ($S^{best}(D_i)$) to be the highest matching score.

4. Modify T_i^{best} to obtain $T_i^{modified}$ by perturbing an existing minutia or adding a new minutia or replacing an existing minutia or deleting an existing minutia. For every time you create $T_i^{modified}$, check if the matching score $S(T_i^{modified})$ is larger than $S^{best}(D_i)$. If yes then change T_i^{best} to $T_i^{modified}$ and change $S^{best}(D_i)$ to $S(T_i^{modified})$.

5. If the best matching score $S^{best}(D_i) > S_{threshold}$ then the matcher will return "yes" and allow the attacking system to access user's i account. Otherwise, if $S^{best}(D_i) < S_{threshold}$ then go to step 4.

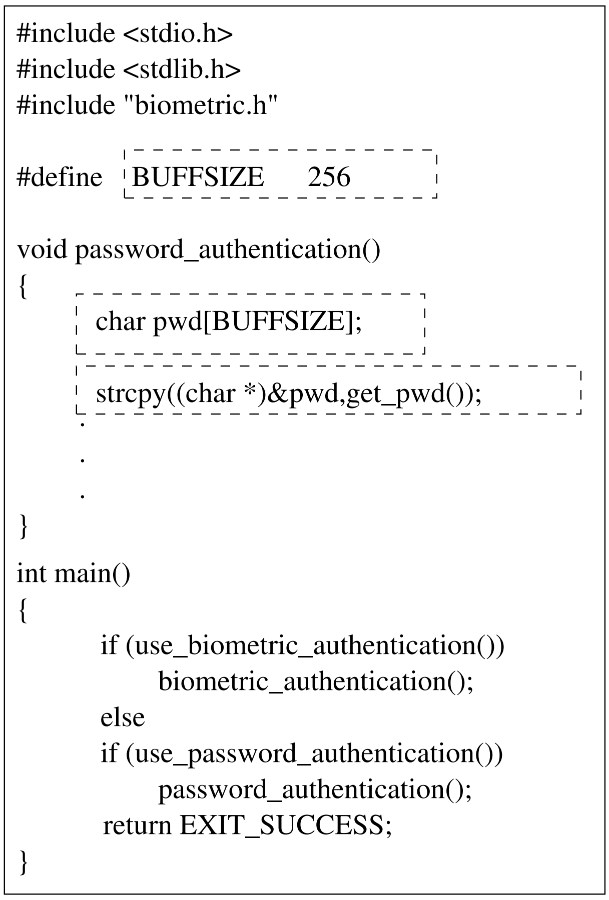
Uludag and Jain's attack system was able to break three accounts at the 132nd, 271st, and 871st attempts (for simulating an "easy" account, a "medium" account, and a "hard" account, respectively, in terms of the number of access attempts necessary to break them) [12].

4.1.5 *Type 5*

This point of attack is known as *"Attack on the matcher"*. This attack is similar to the attack described in section 4.1.3. The difference is that the attacker replaces the matcher with a Trojan horse. The attacker can send commands to the Trojan horse to produce high matching scores and send a "yes" to the application to bypass the biometric authentication mechanism. The attacher can also send commands to the Trojan horse to produce low matching scores and send a "no" to the application all the time causing a denial of service.

4.1.6 *Type 6*

This point of attack is known as *"Attack on the system database"*. In this attack, the attacker compromises the security of the database where all the templates are stored. Compromising the database can be done by exploiting a vulnerability in the database software or cracking an account on the database. In either way, the attacker can add new templates, modify existing templates or delete templates.



```
#include <stdio.h>
#include <stdlib.h>
#include "biometric.h"

#define BUFFSIZE     256

void password_authentication()
{
      char pwd[BUFFSIZE];
      strcpy((char *)&pwd,get_pwd());
      .
      .
      .
}
int main()
{
        if (use_biometric_authentication())
             biometric_authentication();
        else
        if (use_password_authentication())
             password_authentication();
        return EXIT_SUCCESS;
}
```

Figure 3: Vulnerable C code

Hill [2] described a way to create an image of a fingerprint based on the information contained within the stored template (reverse engineering). Hill used a neural network classifier to predict the shape of the fingerprint based on minutiae points. The neural network takes as input minutiae points (where each minutiae point is characterized using its 2D location, ridge, curvature and orientation) and predicts the fingerprint's class. Once the class has been predicted, a synthetic fingerprint image is generated. Hill's proposed technique is observed to work on a database of 25 fingerprints from arch class [4].

4.1.7 *Type 7*

This point of attack is known as *"Attack on the channel between the system database and matcher"*. This attack is again similar to the attack in section 4.1.2. In this attack, the attacker intercepts the communication channel between the database and matcher to either steal and replay data or alter the data.

4.1.8 *Type 8*

This point of attack is known as *"Attack on the channel between the matcher and the application"*. In this attack, the attacker intercept the communication channel between the matcher and the application to replay previously submitted data or alter the data.

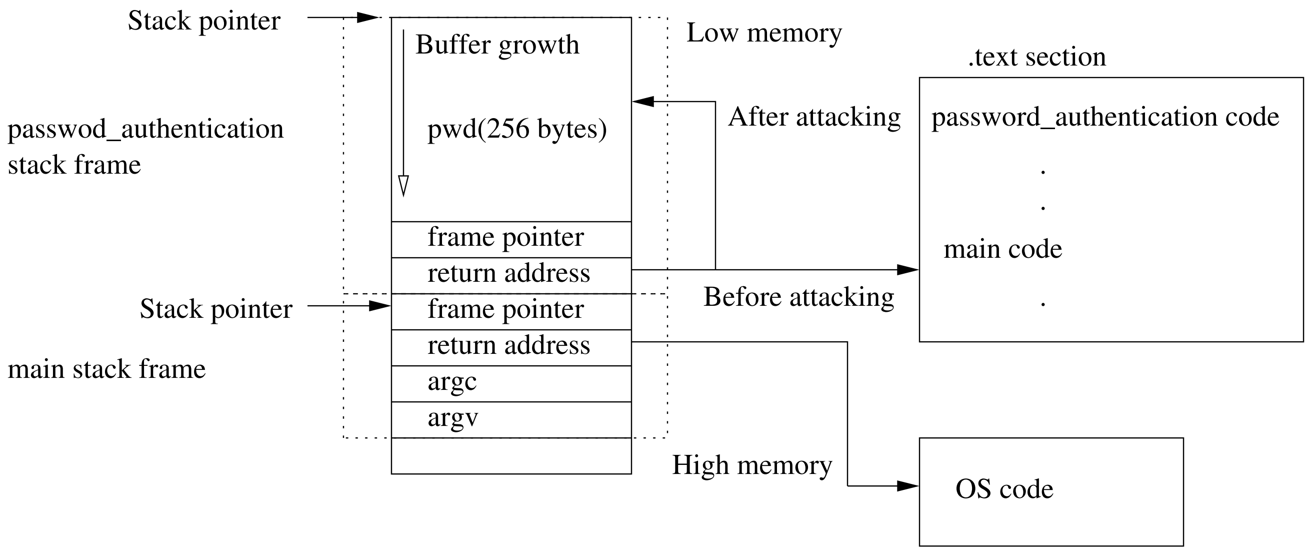


Figure 4: Stack Smashing Attack

4.1.9 Type 9

We claim that a 9th point of attack exists in biometric systems. We call this attack *"Attack on the application"*. Bugs are a consequence of the nature of the programming task that no one can deny. It is a fact that any software has at least one bug in it. Since biometric authentication systems are not 100% accurate, most of these systems use traditional authentication schemes as a backup. For instance, fingerprint readers shipped with laptops these days force you to create a backup password that you can use to access the system if the fingerprint reader does not recognize you for some reason. If a thief tries to break into a house, he can use the main entrance (door) to break into the house or he can figure out another way to enter the house (e.g., window). Therefore, instead of attacking the biometric system directly, one can indirect ways to attack it. One way is by using brute force attacks (e.g., dictionary attacks) that can be mounted against the application to figure out the password. Another way would be to exploit a bug in the application that does the password authentication. If the application in a biometric system suffers from a critical bug (e.g., buffer overflow, double free, etc) then a skilled attacker can exploit this bug by sending the application a crafted input to change the control flow of the program. By changing the flow of the program, the attacker can run code of his choosing with the privileges that the running application have. If the attacker can run code of his choosing then the security of all the modules in a vulnerable biometric system is compromised. For example, the attacker can run code that will download a Trojan horse code from the Internet to replace the modules in the biometric system. The attacker can also run code that will install a sniffer on the compromised system to make it easier to intercept the communication channels in the biometric system.

In figure 3, we show a simple C code that can possibly be part of a biometric system code. The vulnerable points in the code are in the dashed boxes which clearly show that the program is vulnerable to a stack smashing attack. A 256 bytes local buffer has been declared and the C function "strcpy" has been used to copy the given password into the local buffer. The C function "strcpy" does not do any bounds checking on the buffer (the same goes for other C functions like strcat, gets, etc). What do think will happen if we try to input 1024 bytes into the buffer pwd? Of course a segmentation fault will occur. Segmentation faults occur when you try to access an address that does not belong to your program's address space. Since we know that the buffer grows towards high memory (in little-endian architectures), then it is possible to overwrite the frame pointer and the return address in the password authentication function's stack frame. The return address in the password authentication function's stack frame points to the instruction that follows the call to it in main. In our case, it points to "return EXIT_SUCCESS" somewhere in the .text section in memory. To be able to run a code of our choosing which is 200 bytes long then out crafted exploit string (which will be in machine code) can be as follows: *56 bytes nop instructions (90 in machine code) + 200 bytes code + 4 bytes nop instructions (to overwrite the frame pointer) + address of pwd on the stack (4 bytes long).* Figuring out the address of pwd on the stack can be done using trial and error. If we successfully inject the exploit string into pwd then once the password authentication function is done executing, there will be a jump to where the return address is pointing to. Since we changed the return address to point to pwd on the stack (say 0x12345678), then the

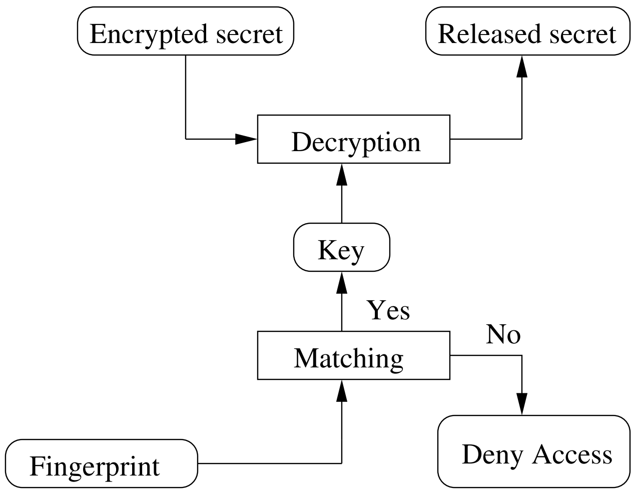


Figure 5: DRM system based on biometrics

following instruction: jmp 0x12345678 will be executed and the injected code in pwd will run (the nop instructions only waste CPU cycles). The attack is shown in figure 4.

5. BIOMETRICS AND CRYPTOGRAPHY

Biometric cryptosystems combine biometrics and cryptography at a level that allows biometric matching to effectively take place in cryptographic domain, hence exploiting the associated higher security [12]. These systems are gaining popularity since they can be used effectively in Digital Rights Management (DRM) systems to make it hard to copy or share digital media illegally. Uludag, Pankanti, Prabhakar, and Jain [13] presented several methods that bind a cryptographic key with the biometric template of a user stored in a database in such a way that the key cannot be revealed without a successful biometric authentication. The idea is as follows (shown in Figure 5): suppose that Alice, a legitimate user, is enrolled on a DRM system that uses biometric based authentication. If Alice wants to access a certain digital content C then she will have to go through the following steps (we will refer to the DRM system as S_{DRM}):

1. Alice selects the digital content C that she wants to access.
2. The S_{DRM} asks Alice to provide her sample biometric data.
3. Alice provides her sample biometric data and the S_{DRM} tries to find a match in the database. Since Alice is enrolled in the system then the S_{DRM} will find a match and release a key K.
4. The S_{DRM} retrieves the content that Alice wanted to access but in an encrypted form $E(C)$.

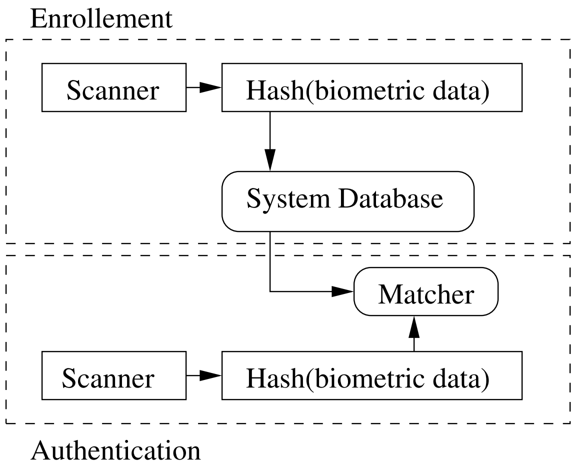


Figure 6: Using hash functions in biometric systems

5. The S_{DRM} uses the key K it released in step 3 to decrypt $E(C)$ and obtain C.
6. The S_{DRM} sends C to Alice.

Now suppose that Mallory, an illegitimate user, tries to impersonate Alice to access a certain media content C. In step 3 above, Mallory will provide the S_{DRM} with her sample biometric data but since Mallory is not enrolled in the system then the S_{DRM} will not find a match and hence deny Mallory from accessing C. Uludag, Pankanti, Prabhakar, and Jain [13] refer to the above method of integrating biometrics into a cryptosystem as the method of *biometric based key release.* As you might have noticed, the above method is vulnerable to many different types of attacks (e.g., replay attacks, Trojan horse attacks, etc). To be able to improve the above method, we need to ask ourselves the following questions (adapted from [13]):

1. Is it possible to design a biometric system such that if the biometric template in an application is compromised, the biometric signal itself is not lost forever and a new biometric template can be issued?
2. Is it possible to design a biometric template such that different applications are not able to use the same biometric template, thus securing the biometric signal as well as preserving privacy?
3. Is it possible to generate/release a cryptographic key using biometric information such that the cryptographic key management is secure and convenient?

According to Uludag, Pankanti, Prabhakar, and Jain it is possible to integrate biometric matching and cryptographic techniques to solve all of the above problems.

To solve questions 1 and 2 above, one can use a one-way hash function (e.g., MD5, SHA-1, etc). One-way functions are known to be computationally infeasible to invert so instead of storing the original biometric signal, say x, in the system database we store its transformed version $H(x)$. During authentication, the biometric sensor would morph the signal using the same transform H (figure 6) and the biometric matching would be carried out in the transformed space [13]. Since one-way functions cannot be inverted then if $H(x)$ ever gets compromised, it will be impossible to get x. Furthermore, new biometric templates can always be issued using a different transform (function). One of the properties of one-way functions is that it is computationally infeasible to find x and y such that $H(x) = H(y)$. Therefore, if the biometric signal x varies then the matcher will have difficulty in carrying out the matching in the transformed space and hence increase the authentication error rate significantly.

To solve question 3 that was addressed earlier, we hide the cryptographic key in the user's biometric template itself (e.g., via a trusted and secret bit-replacement algorithm that can replace, say, the least significant bits of the pixel values/features of the biometric template with the cryptographic key [13]). All what the system has to do is extract the key from the biometric data upon a successful biometric match and release it. The security of this method is highly dependent on the retrieval algorithm and the secrecy of key hiding.

Many other approaches (e.g., the use of digital signatures, fuzzy commitment, etc) has been proposed to utilize the use of biometric cryptosystems. However, all of the proposed approaches still suffer from security problems.

6. CONCLUSION

Biometric systems provide a better and stronger factor of authentication. Such system authenticate individuals based on what they are (e.g., fingerprint, palm print, etc) instead of what they know (e.g., password, PIN) or what they have (e.g., smartcard). Biometric systems proved to be more effective and convenient for users since they do not have to worry about forgetting passwords or losing smartcards, etc. In traditional authentication systems, however, if someone compromises your password or smartcard then the system administrator can revoke the password or smartcard and issue you a new one. In biometric authentication systems, if someone compromises your biometric data template then it is lost forever. No one can issue you, for instance, a new fingerprint unless you figure out a way to replace your fingers.

In this paper, we discussed in-depth the different types of vulnerabilities that exist in biometric systems. We presented a new attack point at the application level that allows the attacker to compromise the biometric system without going through the sensor module. We described how biometric cryptosystems can overcome some of the disadvantages in traditional biometric authentication systems and improve their security. Finally, we showed how biometric cryptosystems can be used effectively in Digital Rights Management (DRM) systems.

7. REFERENCES

[1] A. Adler. Can images be generated from biometric templates? In *Biometric Consortium Conference*, 2003.

[2] C. Hill. Towards reconstructing fingerprints from minutiae points, b.s. thesis, australian national university, http://chris.fornax.net/biometrics.html. 1999.

[3] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. In *Proc. of IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Image- and Video-Based Biometrics*, volume 14, pages 4 – 20, 2004.

[4] A. K. Jain, A. Ross, and U. Uludag. Biometric template security: Challenges and solutions. In *Proc. of 13the European Signal Processing Conference (EUSIPCO)*, 2005.

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

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

[7] N. Ratha, J. H. Connell, and R. M. Bolle. An analysis of minutiae matching strenght. In *Proc. Audio and Video-based Biometric Person Authentication (AVBPA)*, volume 5306, pages 223 – 228, 2001.

[8] P. Reid. *Biometrics for Network Security.* Prentice Hall, 2003.

[9] A. Ross, J. Shah, and A. K. Jain. Towards reconstructing fingerprints from minutiae points. In *SPIE, Biometric Technology for Human Identification II*, volume 5779, pages 68 – 80, 2005.

[10] B. Schneier. The uses and abuses of biometrics. *Comm. ACM*, 42:136, 1999.

[11] C. Soutar. Biometric system security, white paper, bioscrypt, http://www.bioscrypt.com/.

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

[13] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain. Biometric cryptosystems: issues and challenges. In *Proc. of IEEE*, volume 92, pages 948 – 960, 2004.