

# 國立交通大學

資訊工程學系

碩士論文

建構在模糊擷取器及列表法編解碼器上的認證系統  
Authentication Using Fuzzy Extractor with List Decodable  
Codes



研究生：賴建名

指導教授：蔡錫鈞 教授

中華民國九十四年八月

建構在模糊擷取器及列表法編解碼器的認證系統  
Authentication Using Fuzzy Extractor with List Decodable Codes

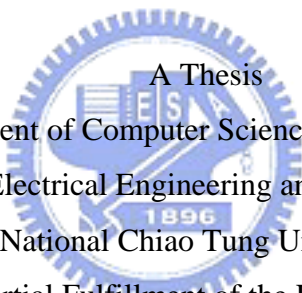
研究生：賴建名

Student : Chian-Ming Lai

指導教授：蔡錫鈞

Advisor : Shi-Chun Tsai

國立交通大學  
資訊工程學系  
碩士論文



A Thesis  
Submitted to Department of Computer Science and Information Engineering  
College of Electrical Engineering and Computer Science  
National Chiao Tung University  
in partial Fulfillment of the Requirements  
for the Degree of  
Master  
in  
Computer Science and Information Engineering

August 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年八月

# 建構在模糊擷取器及列表法編解碼器上的認證系統

學生：賴建名

指導教授：蔡錫鈞

國立交通大學資訊工程學系（研究所）碩士班

## 摘 要

密碼認證系統是最普遍和傳統的認證系統。但是因為人為設定的密碼的亂度不足並且使用者常常使用有意義的字當成密碼，所以密碼認證並不安全。近來有人開始關注生物特徵，例如虹膜、指紋或是手型。Dodis 提出了由模糊擷取器來擷取生物特徵來做密碼學應用的概念。在這一篇論文中，我們以模糊擷取器為基礎建構一個指紋認證系統。除此之外，並在模糊擷取器裡改用列表法編解碼器來加強辨識的能力。

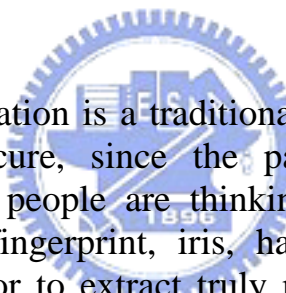
# Authentication Using Fuzzy Extractor with List Decodable Codes

student : Chian-Ming.Lai

Advisors : Dr.Shi-Chun Tsai

Department ( Institute ) of Computer Science and Information Engineering  
National Chiao Tung University

## ABSTRACT



Password authentication is a traditional method to verify rights of a user. But it's not secure, since the password is low-entropy and non-uniform. Recently people are thinking about using the biometric information, such as fingerprint, iris, hand shape...etc. Dodis et al. proposed fuzzy extractor to extract truly random string from biometric data for cryptographic use. In this thesis, based on the fuzzy extractor, we construct a fingerprint authentication system. Furthermore, we use list decodable codes to raise the identification rate of the system.

# 誌 謝

論文的完成要感謝蔡錫鈞老師耐心細心的指導與關心，更要感謝信龍學長和佳蓉學姊不厭其煩的解答我的疑惑。也謝謝家人給我精神上和物質上不虞匱乏的支持，讓我一路可以毫無顧忌的完成學業。在論文寫作期間也要謝謝我的同學和朋友對我的開導和關懷，讓我產生無比勇氣。



# Authentication Using Fuzzy Extractor with List Decodable Codes



Chian-Ming Lai

August 30, 2005



# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
<b>2</b>	<b>Preliminary</b>	<b>13</b>
2.1	Metric Space . . . . .	13
2.2	Error Correcting Codes . . . . .	14
2.3	List Decodable Codes . . . . .	14
2.3.1	Unique Decoding . . . . .	15
2.3.2	List Decoding . . . . .	16
2.4	Fuzzy Extractor . . . . .	17
2.5	Constructions . . . . .	21
2.5.1	Construction under Hamming metric space . . . . .	21
2.5.2	Construction under set difference metric space . . . . .	22
<b>3</b>	<b>Fingerprint authentication system</b>	<b>25</b>
3.1	Background . . . . .	25
3.2	System . . . . .	27
3.2.1	Assumptions . . . . .	27
3.2.2	Some implementation issues . . . . .	27
3.2.3	System structure . . . . .	30
	Sign up . . . . .	30
	Login . . . . .	31
<b>4</b>	<b>Experimental results</b>	<b>35</b>
4.1	The fingerprint authentication system . . . . .	35



4	<i>CONTENTS</i>
4.2 Tests . . . . .	40
<b>5 Conclusion</b>	<b>45</b>
<b>Appendices</b>	<b>48</b>
<b>A</b>	<b>49</b>
A.1 System Requirements . . . . .	49
A.2 Setup . . . . .	49
A.3 File Description . . . . .	50



# List of Figures

2.1	Secure sketch . . . . .	19
2.2	Procedure <i>Gen</i> in fuzzy extractor . . . . .	20
2.3	Procedure <i>Rep</i> in fuzzy extractor . . . . .	21
3.1	Extract biometric data from a fingerprint image . . . . .	26
3.2	Procedure <i>RecList</i> in secure sketch . . . . .	28
3.3	Procedure <i>RepList</i> in fuzzy extractor . . . . .	29
3.4	Sign Up . . . . .	31
3.5	Login . . . . .	32
3.6	Login via list decoding . . . . .	33
4.1	Sign up snapshot 1 . . . . .	36
4.2	Login snapshot 1 . . . . .	37
4.3	Sign up snapshot 2 . . . . .	38
4.4	Login snapshot 2 . . . . .	39
4.5	fingerprint image a to f . . . . .	42
4.6	fingerprint image g to l . . . . .	43
A.1	Relation between classes . . . . .	52



# List of Tables

4.1 Experiments with different fingerprints . . . . .	44
A.1 Files used in the servlet . . . . .	51
A.2 Files used in the applet . . . . .	52





# Chapter 1

## Introduction

Based on the paper [2] proposed by Dodis et al., we construct a fingerprint authentication system with Java Servlet technology and Apache Tomcat web server. Moreover, we enhance the identification rate of the fingerprint authentication by using list decodable code.

The password authentication is a traditional method to verify the access rights of a user. User signs up with a name and a password, and the system uses a hash function (one way function) to get a hashed value by hashing the password and stores the user name and hashed value. As a user logs in with his/her password which is exactly what he/she set, system compares the hashed value of the password with pre-saved hashed value and checks if they are equal. Such an authentication method is traditional and easy, however it's not a secure method. It has two shortcomings. 1) Passwords are rarely truly random, since humans can't remember non-meaningful and long password. 2) Passwords are not reproducible, since user has to type the password exactly what he/she set at the first time. In other words, users always set the meaningful passwords, such as birthday or personal identification number, and usually can't exactly remember long and complicated passwords. Therefore the password authentication can't guarantee security.

There are researchers have proposed methods with nonuniform and low-entropy passwords. For instance, one method is proposed by Ellison et al.[4],

which asks user several private questions and uses the answers to encrypt the secret. The method is useful and practical, but it's not convenient. In order to generate truly randomness for cryptographical use, recently people are thinking about using the biometric information. The biometric data includes iris, fingerprint, hand shape, facial shape, signature and voiceprints (for more biometric data see [5]). The biometric data can not be reproduced precisely each time and is not uniformly distributed. We use the *fuzzy extractor*, which is proposed by Dodis et al.[2], to extract nearly uniform distributed randomness from non-uniform distributed biometric data for cryptographic use.

Dodis also proposes the idea of *secure sketch*, with which a fuzzy extractor can be built. A secure sketch output the sketch  $s$  about the input biometric data  $w$ . While revealing some information about  $w$ , the secure sketch still can reconstruct it from any other input biometric data  $w'$  that is sufficiently close. Moreover, a fuzzy extractor make use of the secure sketch to output a random string  $R$  and a public string  $P$ . We use the random string  $R$  as the strong key used in the fingerprint authentication system of this thesis. The key extracted from the biometric data by the fuzzy extractor is different from the traditional key. We don't need to store the key, since the key can be recovered from the biometric data. It is uniformly distributed and can be easily reproduced.

A fuzzy extractor can generate a strong key as cryptographical use from biometric data in a tolerant way. That is, whatever the biometric data change, if the changes is not much, the fuzzy extractor can correct it. Hence, an error correcting code (see [14] for survey) is needed. In coding theory, given  $n, k$  and  $d$ , a code is a collection of elements (codewords) in  $\{0, 1\}^n$  such that  $d$  is the *minimum distance* of the code and there is a function  $f : \{0, 1\}^k \rightarrow \{0, 1\}^n$  that encodes the message of length  $k$  into a codeword of length  $n$ . The minimum distance of the code is the minimum hamming distance between any two codewords. Since any codeword is at distance at least  $d$  from each other. We can find a unique codeword within distance  $\lfloor \frac{d-1}{2} \rfloor$

from a message of size  $n$ . Let  $t = \lfloor \frac{d-1}{2} \rfloor$ , and  $t$  is called the error correcting bound. The problem to decode a message to at most one codeword within the error correcting code is called *unique decoding*.

The error locating bound is called the traditionally error correcting bound, since no one can correct more errors beyond this bound until Elias[3] and Wozencraft[16] invented the *list decoding* in 1950s. If there is a decoder that can list all possible codewords beyond the traditional bound when receiving a message, we call the decoder a list decoder. The code with this property is a list decodable code. The problem to decode all possible codewords beyond the traditional bound is called list decoding. There are several list decodable codes, including Reed-Muller codes[10], Chinese remainder codes[8], algebraic-geometric codes[7], concatenated codes[6] and Reed-Solomon codes[12]. The first efficient decoding algorithm of Reed-Solomon code is invented by Sudan[15]. We apply Reed-Solomon codes to our system.

Fingerprint is one of the mostly available biometric data. In order to get the fingerprint data, fingerprint readers are needed. It scans the image of the fingerprint and transmits it to the computer for extended visional processes. In order to extract fingerprint data from raw fingerprint image, several graphical processes are applied[11]. In this thesis, we assume there is a method that helps us to extract biometric data from raw fingerprint images. Then we construct a fingerprint authentication system containing a fuzzy extractor to handle the biometric data. Moreover, we use a list decodable code, here is Reed-Solomon Codes, as the error correcting code needed in the fuzzy extractor to raise the recovery capability. The system can be accessed via <https://donna.csie.nctu.edu.tw/FuzzyServlet/>. We develop the system with Java Servlet technology and Apache Tomcat web server(<http://apache.org/>).

Java language(<http://java.sun.com/>) is an object-oriented and cross-platform programming language. The greatest advantages of Java language are reusability and portability, since Java source codes are compiled into Java Bytecodes which can be run under different platforms. Furthermore, Java is simple, since Java has automatic memory management. Programmers don't need to



handle the problem about dangling pointers or memory leaks. The Java technology contains Java Servlet and Java Applet technology. The Java Servlets are modules of code that run in a server application to answer client requests, and the Java Applet is written in Java language and is run in a web browser. However, Java has some security restrictions. It is limited to communicate with the host web server only. We use Java Servlet in the web server to respond the requests from the applet in the client side.

The rest of this thesis is organized as follows. We introduce the preliminaries in chapter 2, including the secure sketch, fuzzy extractor and the Reed-Solomon code. Chapter 3 shows the architecture of the fingerprint authentication system and related implemental issues. Chapter 4 gives some experiments. Last chapter makes a conclusion.



# Chapter 2

## Preliminary

In this chapter, we review several related definitions and introduce fuzzy extractor.

### 2.1 Metric Space

A *metric space* is a set  $M$  and a corresponding distance function  $dist$  that satisfies the following conditions. That is for any three elements  $a, b$  and  $c \in M$ , (1)  $a = b$  iff  $dist(a, b) = 0$ , (2)  $dist(a, b) = dist(b, a)$  and (3)  $dist(a, b) + dist(b, c) \geq dist(a, c)$  (triangle inequality).

For example, the hamming metric space over a field  $\mathbb{F}$ , consists of a metric set  $M \subseteq \mathbb{F}^n$ , consists of a metric set  $M \subseteq \mathbb{F}^n$  with a distance function  $dist : M \times M \rightarrow \mathbb{N} \cup \{0\}$ , which outputs the number of the coordinates in which two metric elements differs. Another example is the set difference metric. A set difference metric space  $M$  contains all the subsets of fixed size  $s$  contained in the universal set  $U = \{0, 1, \dots, n - 1\}$  and for  $A, B \in M$ , the distance function  $dist(A, B)$  outputs the number of elements in  $A$  that are not in  $B$ . Note that  $dist(A, B) = \frac{1}{2}|A \Delta B|$ . Obviously, they all satisfy the above three conditions.

We view biometric data as elements in the metric space. In this thesis we define biometric data over Hamming metric space and set difference metric

space respectively. Besides the metric space issue, a fuzzy extractor needs error correcting codes to provide fault tolerance ability.

## 2.2 Error Correcting Codes

Error correcting codes are widely used in communication. When message is transferred from sender to receiver over a noisy channel, data might be changed. To guarantee correctness and integrity, we add some extended information to original message to ensure that receiver would receive the correct message. Given parameters  $n, k, d$  and a symbol set  $\Sigma$ , codewords are the image of the function  $C : \Sigma^k \rightarrow \Sigma^n$  such that for any two messages  $x$  and  $y$  in  $\Sigma^k$ , the (Hamming) distance of the codewords  $C(x)$  and  $C(y)$  is at least  $d$ . If we receive an encoded message  $y$ , to correct the message is to find the  $x$  with the smallest distance between  $C(x)$  and  $y$ . Since any codeword is at distance  $d$  from any other codewords, for any received encoded message  $y$ , there is at most one codeword within the distance  $t = \lfloor \frac{d-1}{2} \rfloor$ , where  $t$  is called error correcting distance of the code.

In this thesis, we use Reed solomon code which is also a list decodable code.

## 2.3 List Decodable Codes

When we can't find the closest codeword for a received message within error correcting bound  $t$ , we still can find numerous close codewords beyond the bound. We call a code with this property a list decodable code, for example Reed-Solomon code.

**Reed-Solomon Code:** Let  $\Sigma = \mathbb{F}_q$  be a finite field and  $\alpha_1, \alpha_2, \dots, \alpha_n$  be distinct elements of  $\mathbb{F}_q$ . Given  $n, k$  and  $\mathbb{F}_q$ , such that  $k \leq n \leq q$ , a message  $m = \langle m_0, m_1, \dots, m_{k-1} \rangle$  and the polynomial  $p(X) = \sum_{i=0}^{k-1} m_i X^i$ , we can encode a message by a mapping  $C : \Sigma^k \rightarrow \Sigma^n$ , where the codeword  $C(m)$  is  $\langle p(\alpha_1), p(\alpha_2), \dots, p(\alpha_n) \rangle$ .

For any two messages  $M_1$  and  $M_2$  whose corresponding polynomials are  $p_1(X)$  and  $p_2(X)$  of degree at most  $k - 1$ , the distance of the codewords of  $M_1$  and  $M_2$  is at least  $n - (k - 1)$ . We know that the minimum distance  $d \leq n - k + 1$ , and by the Singleton bound  $d \geq n - k + 1$  (see reference [14]), the minimum distance  $d$  is  $n - k + 1$ , Therefore Reed Solomon code is a  $[n, k, d = (n - k + 1)]$  error correcting code and can be used for fuzzy extractor.

### 2.3.1 Unique Decoding

The unique decoding algorithm of the Reed-Solomon code was invented by Berlekamp and Welch [1].

Let  $x_1, x_2, \dots, x_n$  be the distinct elements in  $\mathbb{F}$ , assume we have a received message  $y_1, y_2, \dots, y_n$  with at most  $e \leq \lfloor \frac{n-k}{2} \rfloor$  errors, the original message  $m_0, m_1, \dots, m_{k-1}$ , its corresponding polynomial  $p(X) = \sum_{i=0}^{k-1} m_i X^i$ , and the encoded codeword  $p(x_1), p(x_2), \dots, p(x_n)$ . We can reproduce at most one  $p(x)$ . In order to reconstruct  $p(x)$ , we define an *error locating polynomial*  $E(x) = \prod_{\{i:p(x_i) \neq y_i\}} (x - x_i)$ . That is,  $E(x_i) = 0$  if  $p(x_i) \neq y_i$ .

For all  $i$ , the equation  $E(x_i)(p(x_i) - y_i) = 0$  holds, since  $E(x_i) = 0$  if  $p(x_i) \neq y_i$  and  $p(x_i) - y_i = 0$  if  $p(x_i) = y_i$ . Let  $N(x) = E(x)p(x)$  be a polynomial of degree  $\leq e + k - 1$ , then  $N(x_i) = E(x_i)p(x_i) = E(x_i)y_i$  for all  $i$ . If we can find the  $N(x)$  and  $E(x)$ , we can reconstruct the message. The following is the Berlekamp-Welch unique decoding algorithm.

#### Algorithm 2.3.1. Berlekamp-Welch unique decoding algorithm

*Input:*  $x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n \in \mathbb{F}_q$ , Message size  $k$  and error number  $e \leq \lfloor \frac{n-k}{2} \rfloor$

1. Find polynomial  $N(x)$  and  $E(x)$  such that

- i.  $\deg(E) \leq e$ , and  $E(x) \neq 0$ .
- ii.  $\deg(N) \leq e + k - 1$ , and  $N(x) \neq 0$ .
- iii.  $N(x_i) = y_i E(x_i)$  for all  $i$ .

2. Compute  $p(x) = N(x)/E(x)$  and output the coefficients of the  $p(x)$ .

The algorithm works, if the polynomials  $N(x)$  and  $E(x)$  exists and  $p(x)$  is unique[1]. The brief explanation is given below.

First, to prove the existence of the polynomials  $N(x)$  and  $E(x)$ , we note that the condition  $N(x_i) = y_i E(x_i)$  for all  $i$ . Thus, a solution can be found by solving a homogenous linear system, since there are more equations than the unknowns.

Second, to prove that the polynomial  $p(x)$  is unique, assume that any  $N$  and  $E$  found in step 1 satisfy  $N(x) = p(x)E(x)$ . Let  $R = \{i | p(x_i) = y_i\}$  and  $|R| \geq n - e$ . That is, there are at most  $e$  errors in the received message. Thus, for every  $i \in R$ ,  $N(x_i) - E(x_i)p(x_i) = 0$ , and the polynomial  $N(x) - E(x)p(x)$  has at least  $n - e$  roots and the degree at most  $e + k - 1$ . If the number of roots is more than the degree, the polynomial  $N(x_i) - E(x_i)p(x_i)$  is zero identically. Hence,  $p(x) = \frac{N(x)}{E(x)}$ . The condition is exactly given, since  $e \leq \lfloor \frac{n-k}{2} \rfloor \Rightarrow e + k \leq n - e \Rightarrow e + k - 1 < n - e$ .

The Berlekamp-Welch algorithm uniquely outputs the closest codeword when  $e \leq \lfloor \frac{n-k}{2} \rfloor$ . In next section, the listing decoding algorithm can correct more errors (at most  $n - (k - 1) \left[ \sqrt{\frac{2(n+1)}{k-1}} \right] + \lfloor \frac{k-1}{2} \rfloor$ ).

### 2.3.2 List Decoding

In this section, we describe the list decoding algorithm for Reed Solomon code, which was invented by Sudan [15].

**Weighted degree:** The  $(a, b)$ -weighted degree of the monomial  $q_{ij}x^i y^j$  is defined as  $ai + bj$ . The  $(a, b)$ -weighted degree of a polynomial  $Q(x, y) = \sum_{ij} q_{ij}x^i y^j$  is the maximum, over the monomials with non-zero coefficients, of the  $(a, b)$ -weighted degree of the monomial.

We recall the definition of the Reed-Solomon code. Let  $x_1, x_2, \dots, x_n$  be the distinct elements of  $\mathbb{F}$ , given a message  $(m_0, m_1, \dots, m_{k-1})$  and  $p(X) = \sum_{i=0}^{k-1} m_i X^i$ , the encoded codeword is  $\langle p(x_1), \dots, p(x_n) \rangle$ . Let  $\langle y_1, \dots, y_n \rangle$  be the corrupted codeword. To do the list decoding, we find any close codewords

$\langle c_1, \dots, c_n \rangle$  such that  $|\{i|c_i = y_i\}| \geq t$ . In other words, we have to find all those polynomials  $q$ 's such that  $|\{i|q(x_i) = y_i\}| \geq t$ . The following algorithm gives a way to find all possible polynomials.

**Algorithm 2.3.2.** *List decoding algorithm for Reed Solomon Codes*

*Input:*  $n, k, t$  and distinct pairs  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \in \mathbb{F} \times \mathbb{F}$ .

1. Let  $d = k - 1$ ,  $m = \lfloor \frac{d}{2} - 1 \rfloor$  and  $l = \left\lfloor \left( \frac{t}{d} + \frac{1}{2} \right) - \sqrt{\left( \frac{t}{d} + \frac{1}{2} \right)^2 - \frac{2n}{d}} \right\rfloor$ . Find any function  $Q : \mathbb{F}^2 \rightarrow \mathbb{F}$  that satisfies
  - i.  $(1, d)$ -weighted degree of  $Q(x, y)$  is  $\leq m + ld$ .
  - ii.  $Q(x_i, y_i) = 0$ , for all  $i = 0, \dots, n - 1$ .
  - iii.  $Q \neq 0$ .
2. Factor  $Q$  into irreducible factors.
3. Output all the polynomials  $p$  such that  $(y - p(x))$  is a factor of  $Q$  and  $p(x_i) = y_i$  for at least  $t$  values of  $i$ .

The algorithm works if the bivariate polynomial  $Q(x, y)$  exists and if the polynomial  $p$  can be found[15]. We briefly give the reason why it works.

At first, to prove that  $Q(x, y)$  exists, let  $Q(x, y) = \sum_{j=0}^l \sum_{k=0}^{m+(l-j)d} q_{ij} x^i y^j$ . Try to find the coefficients of  $Q$  such that  $Q(x_i, y_i) = 0$  for all  $i$ . Consider this homogenous linear system, there are more unknowns than equations. Thus, a nonzero solution exists and  $Q(x, y)$  exists.

Second, if  $t > \sqrt{2(n+1)d} - \frac{d}{2} - 1$  and  $|\{i|p(x_i) = y_i\}| \geq t$ , the list decoding algorithm should output  $p$ . To find such polynomial  $p$  is equivalent to prove that  $(y - p(x))$  divides  $Q(x, y)$ . We let  $f(x) = Q(x, p(x))$ . It can be proved that the degree of  $f(x)$  is less than  $|\{i|f(x_i) = 0\}|$ . Therefore  $f(x)$  is identical zero and  $y - p(x)$  divides  $Q(x, y)$ . Hence, the algorithm works.

## 2.4 Fuzzy Extractor

We introduce some involved definitions as follows.

**Min-entropy:** For a random variable  $X$ , the *min-entropy* of  $X$  is  $H_\infty(X) = -\log(\max_x \Pr(X = x))$ . For two random variables  $X$  and  $Y$ , the *average min-entropy* of  $X$  given  $Y$  is defined as  $\tilde{H}_\infty(X|Y) = -\log(E_{y \leftarrow Y}(2^{-H_\infty(X|Y=y)}))$ . We use the average min-entropy to measure the entropy loss of the data source if we give some extended information.

**Statistical distance:** The *statistical distance* of two distributions  $X$  and  $Y$  is defined as  $SD(X, Y) = \frac{1}{2} \sum_w |\Pr(X = w) - \Pr(Y = w)|$ .

**Strong Extractor:** An  $(n, m', l, \epsilon)$ -*strong extractor* is a function  $\text{Ext}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^l$ , such that, for any distribution  $W$  on  $\{0, 1\}^n$  with  $H_\infty(W) \geq m'$ ,  $SD(\langle \text{Ext}(W, U_d), U_d \rangle, \langle U_l, U_d \rangle) \leq \epsilon$ . In other words,  $\text{Ext}(W, X)$  denotes the application of  $\text{Ext}$  to  $W$  using uniformly distributed randomness  $U_d$ .

A secure sketch has two procedures *Sketch* and *Rec*. The procedure *Sketch* produces the public description (called sketch) about the biometric input  $w$ , and the procedure *Rec* recover  $w$  from any biometric input  $w'$  that is close to  $w$  and the original description about  $w$ . The formal definition of the secure sketch is below.

**Definition 2.4.1.** (Def 2. from [2]) An  $(M, m, m', t)$ -secure sketch contains two procedures *Sketch* and *Rec*.

- *Sketch* is a randomized mapping from metric space  $M$  to  $\{0, 1\}^*$ .
- *Rec* is a recovery function, such that for any  $w \in M$  with min-entropy  $m$ , given  $\text{Sketch}(w)$ , and any  $w' \in M$  such that  $\text{dist}(w, w') \leq t$ , we can recover  $w = \text{Rec}(\text{Sketch}(w), w')$ .
- For any random variable  $W$  over  $M$  with min-entropy  $m$ ,  $\tilde{H}_\infty(W | \text{Sketch}(W)) \geq m'$

Note that the random output of *Sketch* is denoted as  $\text{Sketch}(W)$ , or  $\text{Sketch}(W, X)$  when we make the randomness explicit.

A fuzzy extractor provides a similar function. It also contains two procedures *Gen* and *Rep*. The procedure *Gen* extracts randomness  $R$  and public

## Secure Sketch - procedure *Sketch* and *Rec*

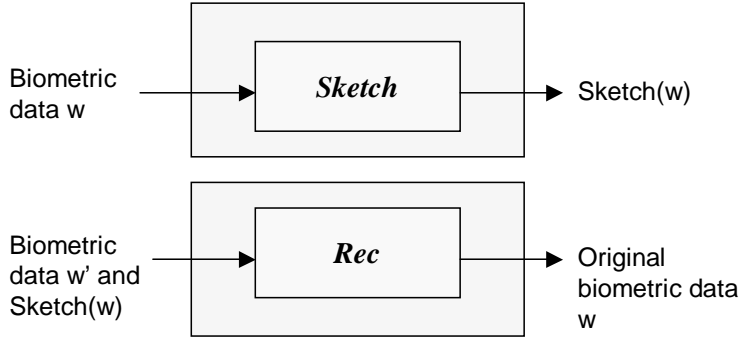


Figure 2.1: Secure sketch

string  $P$  from a biometric input  $w$  which is close to  $w'$ , and the procedure *Rep* extracts  $R$  from biometric input  $w'$  and the public string  $P$ . The following is the formal definition.

**Definition 2.4.2.** (Def 3. from [2]) An  $(M, m, l, t, \epsilon)$ -fuzzy extractor contains two procedures *Gen* and *Rep*:

- *Gen* is a probabilistic procedure, such that for any  $w \in M$  with min-entropy  $m$ , it outputs a string  $R \in \{0, 1\}^l$  and a public string  $P$ , satisfying  $SD(\langle R, P \rangle, \langle U_l, P \rangle) \leq \epsilon$ .
- *Rep* is a procedure that recovers  $R$  from the corresponding public string  $P$  and any  $w'$  close to  $w$ . That is, for all  $w, w' \in M$ , such that  $dist(w, w') \leq t$ ,  $Rep(w', P) = R$ .

A fuzzy extractor addresses the error-tolerance just like a secure sketch. Furthermore, a fuzzy extractor also addresses the nonuniformity. That means that it always extracts a uniformly random string  $R$  reliably from its biometric input  $w$  in an error-tolerant way. Whatever the  $w'$  change, if  $w'$  is still close to  $w$ , the fuzzy extractor always output the same  $R$ .

The procedure *Gen* outputs  $R$  with the restriction  $SD(\langle R, P \rangle, \langle U_l, P \rangle) \leq \epsilon$ . To achieve it, we use a strong extractor to extract  $R$  from  $w$ . The procedure



$Rep$  recovers  $R$  from the public string  $P$  and  $w'$  which is close to  $w$ . Thus, we can use a secure sketch to help a fuzzy extractor to recover  $w$  and use one strong extractor to extract  $R$  from recovered  $w$ . Obviously, a fuzzy extractor can be built by one secure sketch and one strong extractor.

**Lemma 2.4.3.** (Lemma 3.1 from [2]) Given an  $(M, m, l, t, \epsilon)$ -secure sketch containing two procedures  $Sketch$  and  $Rec$ , and an  $(n, m', l, \epsilon)$ -strong extractor  $Ext$  with  $l = m' - 2\log(\frac{1}{\epsilon})$ . Then for  $W \in M$  and uniformly distributed random strings  $X_1, X_2$ , an  $(M, m, l, t, \epsilon)$ -fuzzy extractor is defined as two procedures  $Gen$  and  $Rep$ :

- $Gen(W; X_1, X_2)$  : Compute  $P = \langle Sketch(W, X_1), X_2 \rangle$ ,  $R = Ext(W; X_2)$  and output  $\langle R; P \rangle$ .
- $Rep(W', P = \langle V, X_2 \rangle)$  : Recover  $W = Rec(W', V)$  and output  $R = Ext(W, X_2)$ .

Note that  $V$  is the  $Sketch(W, X_1)$  generated by  $Gen$ .

### Fuzzy Extractor - procedure $Gen$

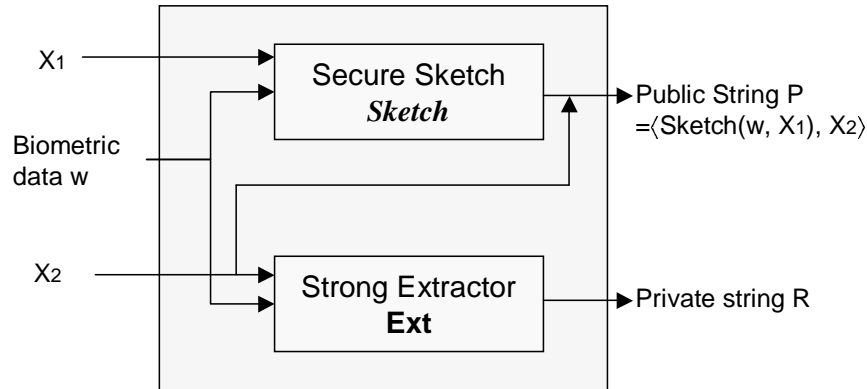


Figure 2.2: Procedure  $Gen$  in fuzzy extractor

Note that the random strings  $X_1$  and  $X_2$  referred in the lemma do not specify the length. It depends on how you make a secure sketch and what strong extractor you choose. There are concrete examples in next section.

### Fuzzy Extractor - procedure *Rep*

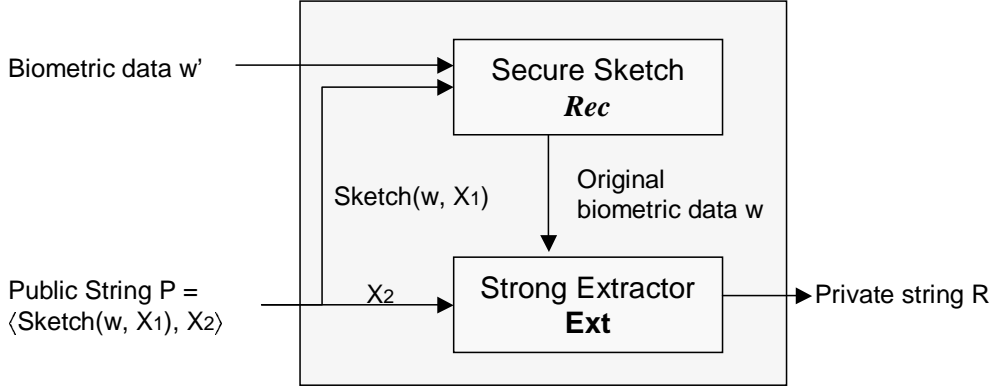


Figure 2.3: Procedure *Rep* in fuzzy extractor

## 2.5 Constructions

This section shows how to make secure sketches under two metric spaces, i.e., Hamming and set difference, and demonstrates how to build fuzzy extractors from those two secure sketches respectively.

### 2.5.1 Construction under Hamming metric space

Given an  $[n, k, d = 2t + 1]$  code  $C$ , the Hamming metric space  $M = \{0, 1\}^n$ , a biometric input  $W \in M$  and a random string  $X_1 \in \{0, 1\}^k$ , the secure sketch is  $Sketch(W; X_1) = W \oplus C(X_1)$ . Let  $S = Sketch(W; X_1)$  and  $dist(W, W') \leq \lfloor \frac{d-1}{2} \rfloor$ . The recovery function is  $Rec(S, W') = S \oplus C(D(W' \oplus S)) = S \oplus C(X_1) = W \oplus C(X_1) \oplus C(X_1) = W$ .

Based on the Hamming secure sketch, given a biometric input  $W \in M$  and random strings  $X_1 \in \{0, 1\}^k$  and  $X_2 \in \{0, 1\}^*$ , the procedure *Gen* of the fuzzy extractor outputs a public string  $P = \langle Sketch(W, X_1), X_2 \rangle$  and a random string  $R = \mathbf{Ext}(W, X_2)$ . The length of  $X_2$  depends on the strong extractor.

To reproduce  $R$ , given a biometric input  $W' \in M$  and  $P = \langle Sketch(W, X_1), X_2 \rangle$ , the procedure *Rep* in the fuzzy extractor first recovers  $W = Rec(W',$

$Sketch(W, X_1)$ ), where  $W'$  is close to  $W$  ( $dist(W, W') \leq t$ ). Second, the procedure  $Rep$  extracts  $R = \mathbf{Ext}(W, X_2)$ . Then we have a fuzzy extractor from a hamming secure sketch.

## 2.5.2 Construction under set difference metric space

Consider a different metric space with the universal set  $U = \{0, 1, \dots, n-1\} = [n]$ . The metric space  $M$  is a collection of all subsets of size  $s$  in  $U$  and the distance function  $dist_{set}$ . Given  $A$  and  $B \in M$ , the distance function is defined as the number of the elements in  $A$  but not in  $B$ . We denote the distance as  $\frac{1}{2}|A \Delta B|$ . Since  $A$  and  $B$  is of the same size,  $|A \Delta B|$  is even and can be divided by 2. Actually, we can represent a set as a binary string. Assume that  $A \in M$ , the  $i$ -th bit of the binary string represents if the  $i$ -th element exists or not. For example, given  $n = 10, s = 5$ , to represent a set  $A = \{0, 1, 3, 5, 7\} \subseteq [n]$  in the set difference metric space, the corresponding binary string is 1101010100.

To make the secure sketch, given a biometric input  $W \in M$ , random string  $X_1 \in \{0, 1\}^*$  and an  $[n, k, d = 4t + 1]$  code  $C$  with constant weight  $s$ , randomly select a codeword  $B$  from  $C$ , and generate one permutation  $\pi : [n] \rightarrow [n]$  such that  $\pi(W) = B$ . Finally output the permutation  $\pi$  as the secure sketch. The length of  $X_1$  depends on how we generate the secure sketch (permutation). Practically we use  $X_1$  as a random seed to select  $B$  from  $C$  and generate the permutation from  $W$  and  $B$ .

We summarize these steps into the following algorithm.

### Algorithm 2.5.1. Make a sketch under set difference metric space

*Input:* a biometric data  $W \in M$ .

1. Randomly select a codeword  $B$  from  $C$ .
2. Choose a random permutation  $\pi : [n] \rightarrow [n]$  such that  $\pi(W) = B$ .
3. Output the permutation  $\pi$  by listing  $\pi(1), \dots, \pi(n)$ .

Given the secure sketch  $Sketch(W, X_1)$  and a biometric input  $W' \in M$  such that  $dist_{set}(W, W') \leq \frac{d-1}{4}$ , the recovery function  $Rec(Sketch(W, X_1), W')$  contains the following steps: Compute  $B' = \pi(W')$ . Since the permutation keeps the distance, the intersection of  $W$  and  $W'$  is the same as of  $B$  and  $B' = \pi(W')$ .  $B'$  can be corrected to  $B$ , and the recovery function finally outputs  $W = \pi^{-1}(B)$ .

In the same way, we can build the fuzzy extractor from the set difference secure sketch. Given a biometric input  $W \in M$  and random strings  $X_1 \in \{0, 1\}^*$  and  $X_2 \in \{0, 1\}^*$ , the procedure  $Gen$  in the fuzzy extractor outputs public string  $P = \langle Sketch(W, X_1), X_2 \rangle$  and  $R = \mathbf{Ext}(W, X_2)$ . There are some differences, i.e., the length of  $X_1$  depends on how you do the secure sketch ( $X_1$  may be used as a seed to generate a random mapping for the permutation), and the length of  $X_2$  depends on the strong extractor.

To reproduce the  $R$ , given a biometric input  $W' \in M$  and  $P = \langle Sketch(W, X_1), X_2 \rangle$ , the procedure  $Rep$  of the fuzzy extractor first recover  $W = Rec(W', Sketch(W, X_1))$ , where  $W'$  is close to  $W$  (that is  $dist_{set}(W, W') \leq t$ ). Second, the procedure  $Rep$  outputs  $R = \mathbf{Ext}(W, X_2)$ . Then we construct a fuzzy extractor from a set difference secure sketch.



# Chapter 3

## Fingerprint authentication system

In this chapter, we show a system that provides fingerprint authentication by using the fuzzy extractor with two metric space.



### 3.1 Background

Fingerprint identification is one of the popular biometrics technologies. When one presses his/her fingerprint on a fingerprint reader, some unique and shorter data are produced to identify the fingerprint owner in an ideal situation. To achieve the goal, several processes are applied to the raw fingerprint image. For example, IBM Exploratory computer vision group gives several comments on the steps to produce data to represent fingerprints[11].

1. Processing - to adjust the size and contrast.
2. Computation of block directions - Determination of primary ridge direction in each sub-region of an image.
3. Foreground/Background Segmentation - Identification of fingerprint area.

4. Ridge Extraction - Extraction of ridge area within the foreground area.
5. Blob Removal - Elimination of non-elongated small structures.
6. Thinning and Morphology - Ridges are thinned into one pixel wide skeletons.
7. Minutia Extraction - Determine location and orientation of ridge bifurcations and ridge terminations.
8. Post processing - Elimination of extraneous minutia.

Usually a fingerprint identification device contains a fingerprint reader and a identification software that extracts biometric data from image. The following figure shows the idea.

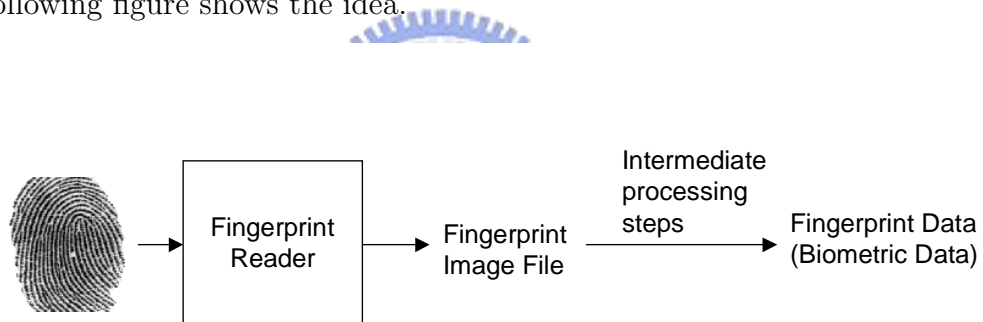


Figure 3.1: Extract biometric data from a fingerprint image

The final fingerprint data is not always unique in real world, since it might be mistakenly identified as some other's fingerprint data. The misidentification rate of the current fingerprint identification systems can be reduced to 1/1000, but the price of the fingerprint identification device with that rate is expensive. Therefore we could consider one device with lower price and high identification rate. We use the fuzzy extractor with list decodable code to raise the identification rate. The method is cheaper and innovative, since the fuzzy extractor can handle the biometric data that is not reproducible and not uniformly distributed.

## 3.2 System

In this section, we discuss the system architecture and related issues.

### 3.2.1 Assumptions

Since we have fuzzy extractor to extract random string  $R$  for cryptographical use, we can build a fingerprint authentication system. However, we have to make some assumptions on fingerprint data. For some reasons, we cannot get a fingerprint identification software that can extract fingerprint data. So we need to generate suitable data in place of those data produced by identification software. Therefore we design an easy method to simulate and generate fingerprint data from fingerprint images.

Suppose user A uses some identification device. One can get two fingerprint images of user A. After extracted by this software, the difference of these two fingerprint data is supposed to be slight. Based on the above observation, our method is to add small noise on the original fingerprint data. Therefore, we get another data which is slightly different from the original one. To sum it up, a fingerprint data corresponds to a user's fingerprint image and such a data with small noise corresponds to the same user's fingerprint image read again later by the device.

In this thesis, we deal with those data generated by the above description. Furthermore, we focus on how to raise the identification rate of the existing methods by fuzzy extractor with list decodable codes.

### 3.2.2 Some implementation issues

When we use list decoding, several possible messages are listed. For better practice, slight changes should be made to the secure sketch and fuzzy extractor for the list decodable code. Intuitively, the recovery function in secure sketch should output several possible results as the figure shows.

Thus, the secure sketch should provide three procedures: sketch, recovery function and additional recovery function for list decoding. The procedures



Secure Sketch - procedure *Rec* with list decodingFigure 3.2: Procedure *RecList* in secure sketch

can be defined in Java interface like this:

```

interface SecureSketch {
Object Sketch(BioData w,BioData X1,Code codec)throws Exception;
BioData Rec(BioData w1,Object result_of_sketch,Code codec);
BioData[] RecList(BioData w1,Object resutl_of_sketch,Code codec);
}
  
```

Procedure *RecList* is added to the secure sketch, and it outputs a biometric data array that stores all possible  $w_1, w_2, \dots$  etc.

Based on the secure sketch, the fuzzy extractor with list decodable code should output all the possible random strings  $R_1, R_2, \dots$  etc. In the same way, we add one reproduction function for list decoding method in fuzzy extractor. The following figure shows the changes.

The following is the Java interface *FuzzyExtractor*.

```

interface FuzzyExtractor {
Object[] Gen(BioData w,BioData X1,BioData X2,Code codec)throws Exception;
BioData Rep(BioData w1,Object[] P,Code codec);
BioData[] RepList(BioData w1,Object[] P,Code codec);
}
  
```

Procedure *RepList* is added to the original fuzzy extractor.

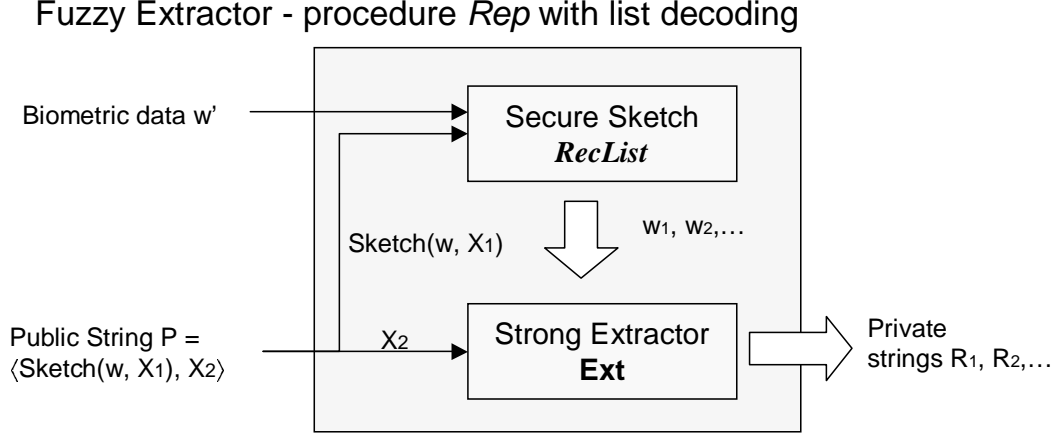


Figure 3.3: Procedure *RepList* in fuzzy extractor

Another implementation issue is the secure sketch under the set difference metric space. The restriction on the constructions is that the code must be a constant weight code. We implement the secure sketch in a tricky way to avoid using the constant weight code. As we discuss in the previous chapter, a set can be view as a binary string. Hence we can view the intersection of two sets as the exclusive-or of two binary strings. Based on this observation, we modify the algorithm as follows.

**Algorithm 3.2.1.** *Modified Secure sketch under set difference*

*Input:* a biometric data  $W \in M$ .

1. Randomly select a codeword  $B$  from an  $[n, k, d = 4t + 1]$ -code  $C$ .
2. Generate a random permutation  $\pi : [n] \rightarrow [n]$
3. Compute  $E = \pi(W) \oplus B$
4. Output the permutation  $\pi$  by listing  $\pi(1), \dots, \pi(n)$  and  $E$ .

This secure sketch also provides recovery capability. For any input biometric data  $W$ , assume that we have sketch  $\langle \pi, E \rangle$  and  $W'$  where  $\text{dist}_{\text{set}}(W, W') \leq$

$t$ , the recovery function contains the following steps: Compute  $B' = \pi(W') \oplus E$ .  $B'$  can be decoded to  $B$ . Finally recovery function outputs  $W = \pi^{-1}(B \oplus E)$ . It works since the modified permutation still keeps the distance. Based on the secure sketch, a fuzzy extractor can be built, but more storage is needed for the sketch.

### 3.2.3 System structure

Some free resources are used, such as Java Servlet and Apache Tomcat web server with SSL(Secure Socket Layer) support. Servlets are modules of code that run in a server application to answer client requests. We use an Apache tomcat web server with servlet. This servlet easily answers two kinds of commands from the applet in the client. One is to sign up a new user, the other is to verify the user. Users can type in the user name and the fingerprint in the applet to ask for service from servlet. The servlet contains Fuzzy extractor to extracts data, and the applet gets the fingerprint data from the raw image and communicates with the server. We introduce the details of how to sign up and verify in the following.

#### Sign up

The run-down processes can be listed in the following steps, where  $h$  is a one way hash function.

1. Key in the user name *username* and read the fingerprint  $w$  from the client side.
2. Send the data from the client side to the server side via secure sockets layer.
3. The server checks if *username* already exists in the database or not. If exists, outputs the message "User existed" to the client and terminate.
4. The fuzzy extractor procedure *Gen* takes the  $w$ , and computes the random string  $R$  and the public string  $P$ .

## Sign up run-down diagram

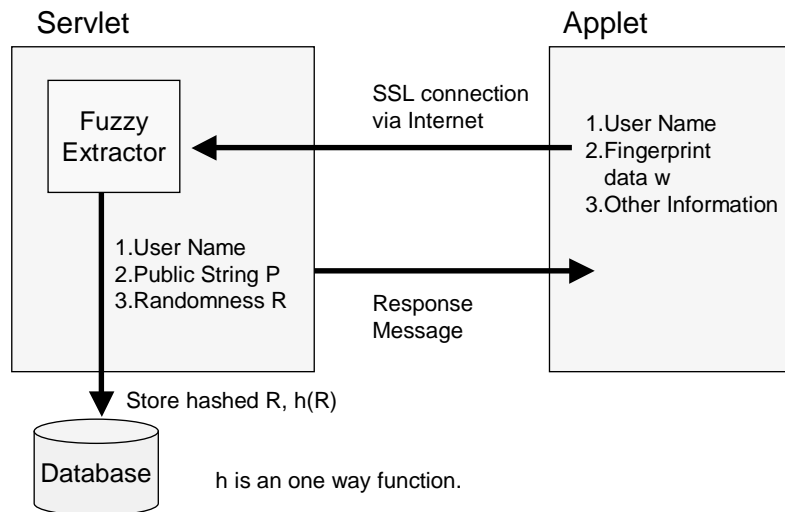


Figure 3.4: Sign Up

5. The server computes  $h(R)$ . Stores  $h(R)$ ,  $P$  and the unique *username* in the database.
6. Output the message "Sign up successfully" to the client.

**Login**

Similarly, a user login with a user name and fingerprint. The following is the whole run-down processes.

1. Key in the user name *username* and read the fingerprint  $w'$  in the client side.
2. Send the data from the client side to the server side via secure sockets layer.
3. The server checks if *username* exists in the database or not. If not, output the message "User does not exist, login fail!" to the client and terminate.

### Login run-down diagram

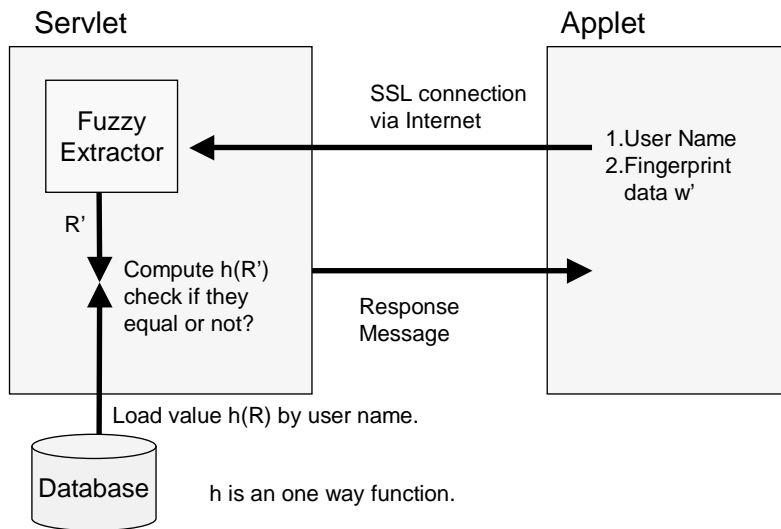


Figure 3.5: Login

4. The server load the stored  $h(R)$  and  $P$  by the *username*.
5. The fuzzy extractor procedure  $Rep$  takes the  $w'$  and  $P$ . computes the random string  $R'$ .
6. The server checks if  $h(R')$  is equal to  $h(R)$  or not. If yes, output message "Login successfully!" ; no, output message "Login fail!" to the client.

If the list decoding is used. The login step 5 and 6 are modified as follows.

5. The fuzzy extractor procedure  $RepList$  takes the  $w'$  and  $P$ , and computes the random strings  $R_1, R_2, \dots$  etc.
6. The server checks if  $h(R)$  is equal to some  $h(R_i)$  or not. If yes, output message "Login successfully" ; no, output message "Login fail" to the client.

Login run-down diagram with list decoding

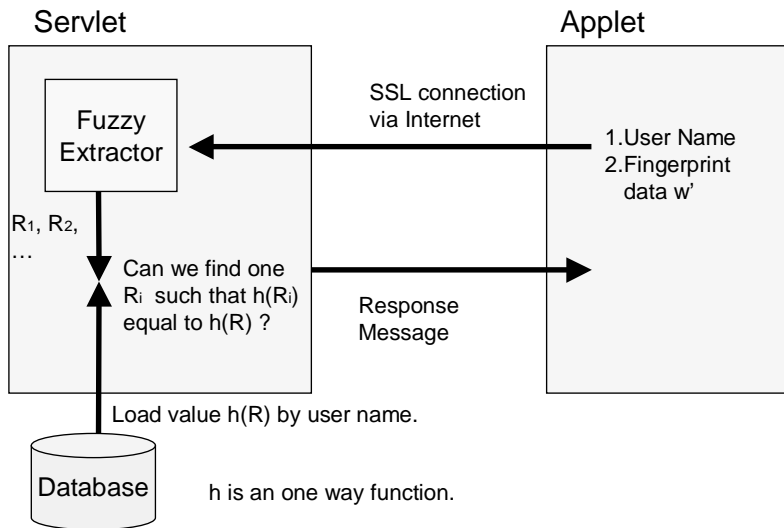


Figure 3.6: Login via list decoding

Because the list decoding take much more time than unique decoding. For better practice, we still try to do unique decoding first. If fails, then it switches to the list decoding.



# Chapter 4

## Experimental results

We use fuzzy extractor to improve the identification rate of existing fingerprint identification methods. This chapter gives some experimental results.

### 4.1 The fingerprint authentication system

At first we take snapshots on the system. The system can be accessed via <https://donna.csie.nctu.edu.tw/FuzzyServlet/>. There is one tabbed panel including two parts. One is for new user registration, the other for login panel.

#### **New User Sign up:**

In the new user registration panel, a user can choose a random string or a sample fingerprint as the fingerprint data. When a user chooses "random string", there is one button "create" and an uneditable text field. The uneditable text field shows the fingerprint in the Base64 format (the characters a-z, A-Z, the numbers 0-9 and the "+", "-" symbols, with "=" symbol as special suffix code). When users key in the user name and click the "create" button, a random string is produced as a biometric data and placed in the uneditable text field.

If one uses the sample fingerprint, there shows a list in the left side and a gray block in the right side. The list has several samples, choose one fingerprint and the image will show in the gray block. Finally you can construct



Fuzzy Extractor Application: Authentication via Biometric or Noisy Data (Simulation)

New User Registration Login

**Login with your Id and noisy Fingerprint**

User Name:

Noisy FingerPrint:  Paste my fingerprint string.  
 Select sample fingerprint image.

\* Copy your fingerprint and paste here.

Add Noises  0 15 30 45 60 75 90 22 %

Noisy Fingerprint:

Bound:  \* List decoding will spend more time.

Messages:

Response Msg: \*\* user "Flora" login successfully.

Figure 4.1: Sign up snapshot 1

Fuzzy Extractor Application: Authentication via Biometric or Noisy Data (Simulation)

**New User Registration**   **Login**

**Register your User Name**

User Name:

FingerPrint:  Use random string as biometric data.  
 Use sample fingerprint as biometric data

Construct under:  Hamming distance    Set distance

\* After sign up, data will be copied to next page.

**Messages:**

User: Jean  
FingerPrint(Length:20):  
BfF6JN9/frHttjF6zhF6To5tlN+yB1ICjAZbqHE1TmaW0NMfXiqDRHK7mwbFHNr+

Randomly generate an unique fingerprint for user "Flora"

Figure 4.2: Login snapshot 1

Fuzzy Extractor Application: Authentication via Biometric or Noisy Data (Simulation)

**New User Registration**   **Login**

**Register your User Name**

User Name:

FingerPrint:  Use random string as biometric data.  
 Use sample fingerprint as biometric data

Sample1  
Sample2  
Sample3  
Sample4



Construct under:  Hamming distance    Set distance

**Sign Up** \* After sign up, data will be copied to next page.

**Messages:**

Sign up successfully.  
You could save your personal information below for future simulation use.  
User: Jimmy  
FingerPrint(Length:20):  
r9CJ5Q1co8FJZ00BtW+/Mv0h40mWhYoz2+JsuJ4FuT8d47K89XN8Kwa79OD2eKqr

Figure 4.3: Sign up snapshot 2

Fuzzy Extractor Application: Authentication via Biometric or Noisy Data (Simulation)


New User Registration Login

**Login with your Id and noisy Fingerprint**

User Name:

Noisy FingerPrint:  Paste my fingerprint string.  
 Select sample fingerprint image.

Sample1  
Sample2  
Sample3  
Sample4



\* Use mouse to make some noises on image.

Bound:  \* List decoding will spend more time.

Messages:

Response Msg: \*\* user "Jimmy" login successfully.

Figure 4.4: Login snapshot 2

your fingerprint data under the different metric spaces by setting the lower radio buttons.

As you sign up, some information is shown in the message area, including the user name you signed up and the fingerprint data in the Base64 format. These data will be directly copied to the "login" panel for your own use.

### **Login:**

The login panel also provides similar look. Users type in the user name with noisy fingerprint.

If choose "paste my fingerprint string", it shows an editable text field, a button "Add noises", a slide bar and an uneditable text field "Noisy fingerprint" in the middle of the panel. If one just generates a random string as the fingerprint data in the sign up panel, the editable text field will be filled automatically. Otherwise, one can copy the fingerprint data and paste it. Then one can adjust the slide bar to decide the noise rate for the fingerprint data in the upper editable text field and click the button "Add Noises" to display the result in the following text field "Noisy fingerprint".

When select a sample fingerprint, there shows a list on the left side and a gray block will be on the right side. One can choose a sample from the list and selected image will be displayed in the gray block. Then one can press the mouse and drag on the image to make some black sketches on it.

In the lower part of the panel, there is a bound setting. If the verification fails, then one can choose "List decoding" for better identification rate.

## **4.2 Tests**

Based on the assumption we made earlier. When a user's fingerprint is read, the fingerprint reader always gives the same fingerprint image with some errors. We design an easy method to simulate and generate fingerprint data from fingerprint images.

Since the size of a fingerprint image is larger than fingerprint data, the tradeoff between the efficiency and the size of the fingerprint data should be

considered. In our system, the fingerprint image size is  $160 \times 160$  pixels and we cut the image into 64 blocks, that is,  $8 \times 8$  blocks on the image. Each block ( $20 \times 20$  pixels) can be represented as an integer between 0 and 63. We use  $(64, 8)_{26}$  Reed Solomon Code for implementation. Once you login or sign up with fingerprint image, a message is produced as a fingerprint data.

We design the method to extract biometric data from processed fingerprint image, since we assumed that we do not focus on how to extract fingerprint data from raw fingerprint image. If there are better intermediate processing steps to process the fingerprint image into the fingerprint data, we can apply our method to these steps to gain some advantages from the fuzzy extractor.

We sign up with the first sample fingerprint image and verify with following the noisy fingerprint images. There are twelve fingerprint images  $a$  to  $l$ . Images  $a$  to  $e$  can be verified successfully via unique decoding. Images  $f$  to  $h$  fail with unique decoding but pass with list decoding. Images  $i$  to  $l$  fail in both decoding methods.

These error patterns are reasonable, since there might be objects, such as hair, on the scanning device. If the error is minor, then it can be handled via unique decoding. Besides, users may wound their fingers, which we simulate with images  $f$ ,  $g$  and  $h$ . For these cases we can choose list decoding for larger recovery capability.

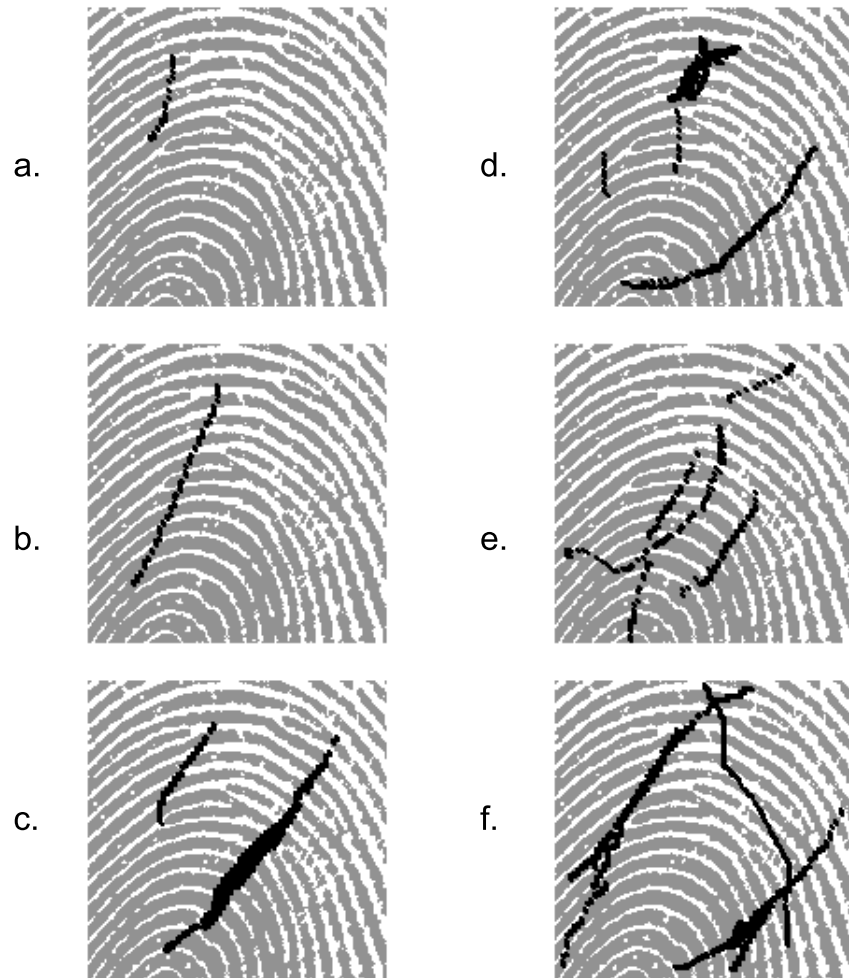


Figure 4.5: fingerprint image a to f

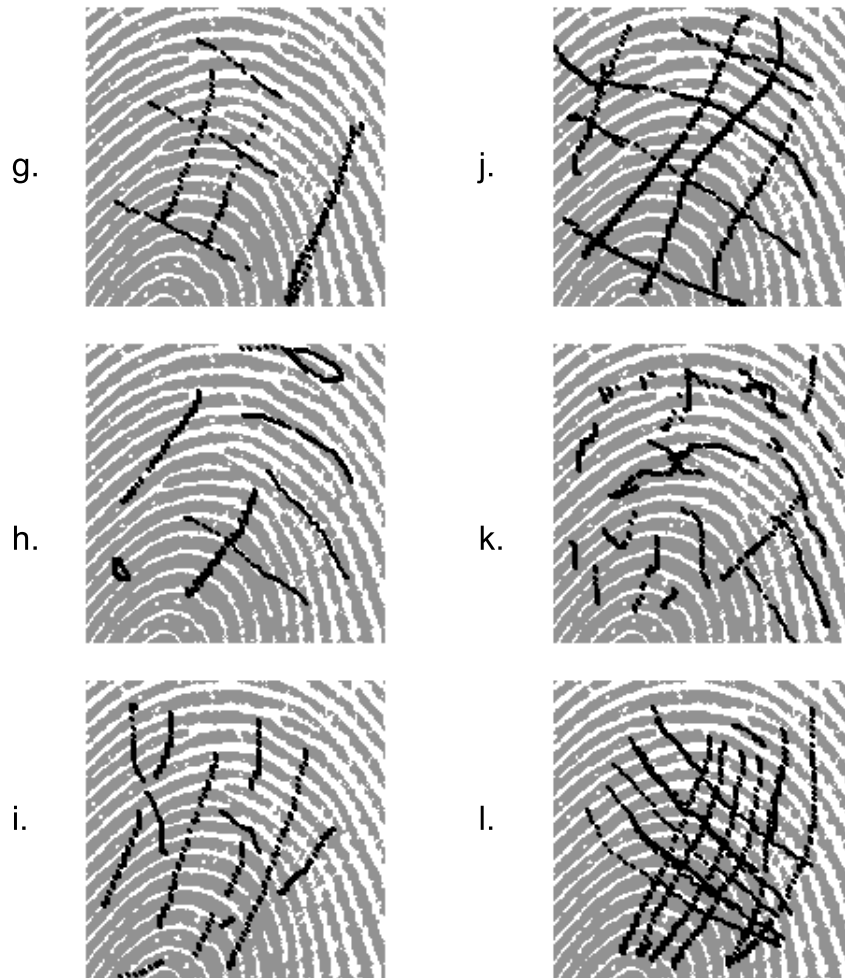


Figure 4.6: fingerprint image g to l



Image	Results with unique decoding	Results with list decoding
a	pass	pass
b	pass	pass
c	pass	pass
d	pass	pass
e	pass	pass
f	fail	pass
g	fail	pass
h	fail	pass
i	fail	fail
j	fail	fail
k	fail	fail
l	fail	fail

Table 4.1: Experiments with different fingerprints

# Chapter 5

## Conclusion

In this thesis, we construct a fingerprint authentication system by using fuzzy extractor with unique decoding and list decoding. We not only implement the fuzzy extractor under two different metric space, but also provide list decoding for more powerful recovery capability. Although some assumptions are made for the some restriction, our system provides the evidence of usability.

We assume there is a method that helps us to extract fingerprints data. Actually the method plays an important part of the identification system. The future work is to design steps that extract fingerprint data that actually reflects important features on fingerprint image.



# Bibliography

- [1] E.R. Berlekamp and L.R. Welch, Error correction for algebraic block codes, *US Patent 4 633 470*, 1986.
- [2] Y. Dodis, L. Reyzin, and A. Smith, Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data, *In Advances in Cryptology - EUROCRYPT*, May 2004.
- [3] P.Elias, List decoding for noisy channels, *Institute of Radio Engineers*, 94-104, 1957.
- [4] C.Ellison, C.Hall, R.Milbert, B.Schneier, Protecting keys with personal entropy. *Future Generation Computer Systems*, 16, pp. 311-318, 2000.
- [5] N.Frykholm, Passwords: Beyond the Terminal Interaction Model, *Master's Thesis, UMEA University*.
- [6] V.Guruswami and M.Sudan, List decoding algorithm for certain concatenated codes, *Proceedings of the 32nd annual ACM symposium on Theory of computing*, 181-190, May 2000.
- [7] V.Guruswami and M.Sudan, Improved Decoding of Reed-Solomon Codes and Algebraic Geometric Codes, *IEEE Transactions on Information Theory*, vol. 45, no 6, pp. 1757-1767, September 1999.
- [8] V.Guruswami, A.Sahai and M.Sudan, Soft-Decision Decoding of Chinese remainder Codes, *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 2000.

- [9] N.Nisan, D.Zuckerman, Randomness is Linear in Space, *In JCSS*,52(1), pp.43-52,1996.
- [10] R.Pellikaan and Xin-Wen Wu, List decoding of q-ary Reed-Muller codes, *IEEE Transactions on Information Theory*, April 2004.
- [11] IBM Exploratory Computer Vision Group, Fingerprint Steps, <http://www.research.ibm.com/ecvg/biom/fp-steps.html>
- [12] I.S.Reed and G.Solomon, Polynomial codes over certain finite field, *J.SIAM*, 8:300-304, 1960.
- [13] D.R. Stinson, Universal hash families and the leftover hash lemma, and application to cryptography and computing, *J. Combin. Math. Combin. Comput.* 42 (2002), 3-31.
- [14] M. Sudan, Coding Theory: Tutorial and Survey. *Online talks from* <http://theory.csail.mit.edu/madhu/>, April 2001.
- [15] M. Sudan, Decoding of Reed Solomon codes beyond the error-correction bound, *Journal of Complexity*,13(1):180-193, 1997.
- [16] J.M. Wozencraft, List decoding, *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, 48:90-95, 470, 1958.

# Appendix A

## A.1 System Requirements

Our system are constructed by some free resources as follows.

- Apache Tomcat web server 5.5.9(<http://jakarta.apache.org/tomcat/>)
- Java 2 Platform, Standard Edition 1.5 (J2SE 1.5 <http://java.sun.com/j2se/index.jsp>)  
Java 2 Platform, Enterprise Edition 1.4 (J2EE 1.4 <http://java.sun.com/j2ee/index.jsp>)

Note that J2EE 1.4 is necessary for developing a servlet.

## A.2 Setup

In this section, We setup Apache Tomcat web server and add our servlet on it.

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. In order to setup the web server with SSL (Secure Socket Layer) supported, install the Apache Tomcat web server and do the steps provided in <http://jakarta.apache.org/tomcat/tomcat-5.5-doc/ssl-howto.html>.

We summarize it into the following steps.

1. Create a certificate keystore by executing the following command:  
Windows:

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
```

Unix:

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
```

and specify a password value of "changeit".

2. Uncomment the "SSL HTTP/1.1 Connector" entry in `$CATALINA_HOME/conf/server.xml` and tweak as necessary.

Note that the variable name `$CATALINA_HOME` is the directory into which you have installed Tomcat 5.

In step 1, we create a certificate keystore in default file name ".keystore". We can create one keystore with specific name by the option "-keystore [FileName]". To use this keystore file in the web server, move the file to the directory `$CATALINA_HOME`.

In step 2, we enable the web server to accept SSL connection. If we want to use specific keystore file, we should setup the attributes "keystoreFile" and "keystorePass" in the "SSL HTTP/1.1" entry.

Finally, copy the folder "FuzzyServlet" to the directory `$CATALINA_HOME/webapps/` to complete the setup. The system will create a file "bData.txt", which stores user information, in the directory `user.home`. The variable name `user.home` is the user's "home" directory. If we install Tomcat as system service, the variable `user.home` denotes the local service directory. For example, it denotes the directory "`\Documents and Settings\LocalService`" in the Microsoft Windows XP Operating system.

### A.3 File Description

File name	Description
Account	Object that stores the user information
BdataAccess	Object that maintains user information database
BioData	Basic object that stores the fingerprint data
Code	Java interface for a code
Command	The object that received by the servlet
ExtGF	Extended Field object
FiniteField	Java interface for finite field
FuzzyExtractor	Java interface for fuzzy extractor
FuzzyServlet	Our servlet
GF	Galois Field
HammingFE	Hamming Fuzzy extractor
HammingSS	Hamming Secure sketch
HashF1	Hashing function used in strong extractor
Permutation	Object in the set difference secure sketch
Poly	Polynomial object.
ResponseMsg	Object that sent by the servlet
RS	Reed-Solomon code
SecureSketch	Java interface for secure sketch
SetFE	Set difference fuzzy extractor
SetSS	Set difference secure sketch
StrongExtractor.java	Strong extractor used in a fuzzy extractor.

Table A.1: Files used in the servlet



Class name	Description
Account	Object that stores user information
BioData	Basic object that stores the fingerprint data
Command	Object that sent by the applet to the servlet
FingerImage	Graphic panel that handles fingerprint images
JM	Our applet.
index.html	Web page that contains the applet "JM"
ResponseMsg	Object that received by the applet
waitProcess	System uses this thread to wait for response

Table A.2: Files used in the applet

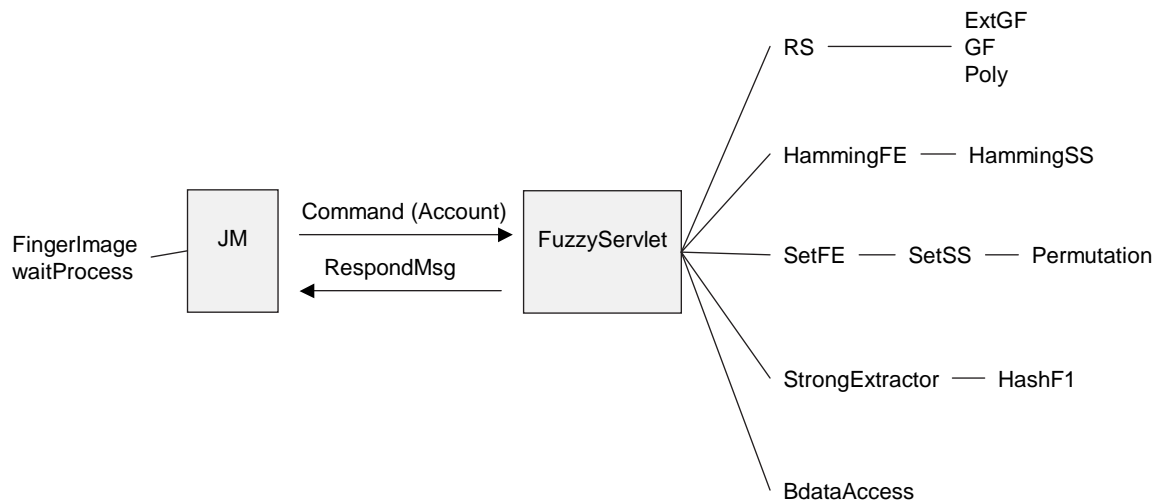


Figure A.1: Relation between classes