丘成桐中学应用数学科学奖-工程应用 (2012)

S. -T. Yau High School Mathematics Awards- Engineering Applications (2012)

学生:谭知微

Student: TAN, Chih Wei

澳门培正中学

Pui Ching Middle School, Macau

中华人民共和国澳门特别行政区

Macao SAR , People's Republic of China

指导老师:黄灿霖

Supervisor: Mr. WONG, Chan Lam

澳门培正中学

Pui Ching Middle School, Macau

中华人民共和国澳门特别行政区

Macao SAR , People's Republic of China

云深不知处

Clouds Thick, Whereabouts Unknown

-代数学在云端储存上的应用-

-Application of Algebra to Cloud Storage-

Clouds Thick, Whereabouts Unknown
-Application of Algebra to Cloud Storage-

Summary

This project aims on the research of application of algebra to cloud computing.

First of all, we associate every two bits of a file with a quadratic function, then by making use of the mathematical theory “three distinct points on the plane can uniquely determine a quadratic function”, we are able to design a distributive storage scheme which enable us to divide a particular file into $n \geq 3$ other different container files separately. In reverse, if we want to recover the original file, we could finish the procedure conveniently by just making use of any three different container files.

On the other hand, the employment of the technique of permutations could allow us to encrypt the container files according to their layers as well as the entries within the layers. Therefore the security level of this technique could be greatly improved. What's more, we could apply the Ruffini Theorem which could help in deciding the greatest value of the period (or order) of permutations. Therefore it would benefit not only the process of implementation, but could also maintain a high level of security.

In practice, we have developed a C program to implement the proposed algorithm. It allows the program to generate the container files and recover the original files swiftly, and the sizes of the resulted container files are even reaching up to 68% of that of the original file after compression. Therefore with no doubt it would benefit on the speed of transfer of the container files via internet. In addition, we could associate a pair of bits with the other quadratic functions or even non-polynomial functions in order to make the encryption more complex and harder to be broken. What’s more, since the distributive storage and encryption algorithms are highly parallel, we could develop a parallel version of the C program for speedy implementation.

The cloud computing system, with its original exquisite design as well as simple and easy implementation character which shows the practical value of the program, we believe that it would be a satisfying and user-friendly technology for the modern society.

Last but not least, on both theoretical and practical level, this newly developed program, with its propound sights on modern data storage, should worth more space for further advance studies and development in future.

云深不知处

-代数学在云端储存上的应用-

摘要

本项目主要是代数学原理在云端计算上应用的研究。

首先，我们将档案的每两个位元(bit)对应一元二次函数，然后利用“三个不同点确定唯一的一元二次函数”这一数学原理来设计分散储存技术。此技术可以将一个档案分散储存在 $n \geq 3$ 个彼此相异的容器档；如果要回复原档，则只需要同时使用任何三个不同的容器文件就能完成任务。

另一方面，我们亦会应用置换函数，设计出各个容器档的分层加密及层内的数字加密算法，藉此大大提高此储存技术的安全等级。我们还可以应用 Ruffini 定理来得到最大循环周期的置换函数，这样既能方便云端存储的编程和实践，亦能确保加密方法的高度安全性。

在技术实践方面 ,我们运用 C 语言来编写其加密分散储存系统的软件部分。此程序不但能迅速产生容器档及还原与其对应的原档，而且只要容器档经过压缩后，其大小甚至可以低至原档大小的 68%，这样便能更有利于容器档在互联网上的往来传送。除此之外，我们还可以更进一步，让每两个位元能对应到其他的一元二次函数，甚至非多项式函数，使原来的分散储存

方法变得更富弹性、容器文件变得更复杂和加密法更难以破解。而另外具有优势的是，无论是分散技术的储存恢复及加密方法都是高度平行的，所以我们亦可编写平行程序来加速系统的运作。

这个原创的云端存储系统，灵活巧妙地运用了代数学原理，其创新思维设计、精巧简洁的系统，都令整个程序的运行变得既保险，又容易掌握和使用，相信必定是一项对现代社会有功用的研究。

最后，我们相信这新研发项目，无论在理论设计及运行实践上，都有更大及更多的进步及研究空间，可期待日后继续发展。

云深不知处

Clouds Thick, Whereabouts Unknown

-Algebra Applications to Cloud Storages-

Chapter 1: Introduction

The world keeps evolving, so as our life. Alvin Toffler, an America futurist, described in his famous book *The Third Wave* [1], that human progress could be divided into three 'waves': The Agricultural Revolution constitutes the First wave; the Industrial Revolution, the Second Wave and the Third Wave, which is a different world we have just entered, comprises the Information Age based on the revolution brought by Computer Technology. To review from the past, IBM developed the mainframe computer in the 60s of the 20th century; personal computer (PC) become popular in the 90s which followed immediately by information explosion brought by internet. Nowadays, network servers and users exist everywhere in our world, perform various calculation tasks according to different enquires. Now, in the early 21st century, we are living in the Third Wave society which represented as the Cloud Computing Era. Cloud computing has become increasingly important owing to the continuous economic development. Apart from performing calculation and providing storage at supercomputer-level at any time, cloud computing require much lesser cost compared to the supercomputers.

Cloud computing is characterized by the following 5 basic properties [2] :

1. Multi-Tenancy (Shared Resources)

   Resources are shared by all the users.

2. Massive Scalability

   Serve many users at the same time with sufficient bandwidth and data storage.

3. Elasticity

   The numbers of users could be elastic. After the quitting of one user, the resources would be redistributed to the others.

4. Pay as You Go

   The users need to pay only for the used resources.

5. Self-provisioning of Resources

   The users could apply more resources according to their needs.

In this report, we will focus on one part of cloud computing, that is distributive storage. We always want resources to be used in a more safe, convenient and economic manner, and distributive storage could help us with this: It could divide a file into several parts and save them into different storages. We could also collect and recombine them into the original file from the storages whenever we need it.

An ideal distributive storage should satisfy the following conditions:

1. Safety

   Data provides only for the authorized users, and could not be easily stolen during the transfer process.

2. Availability

   Data access limits no time and places.

3. Reliability

   Minimize the possibilities of the stop of service when the system fails.

4. Implementation

   The system could run smoothly.

In order to satisfy the above conditions, we have the following proposal:

Assume that there is file. We could divide it into $n \geq 3$ container files through a specific program and save them into different storages respectively. The divided files are different from each other and could be encrypted. If we want to recover the

original file, we only need to collect ***any*** three different container files, and make use of another specific program in the host computer to decrypt and combine them, and then we can get the original file. The process is shown by the following diagram:

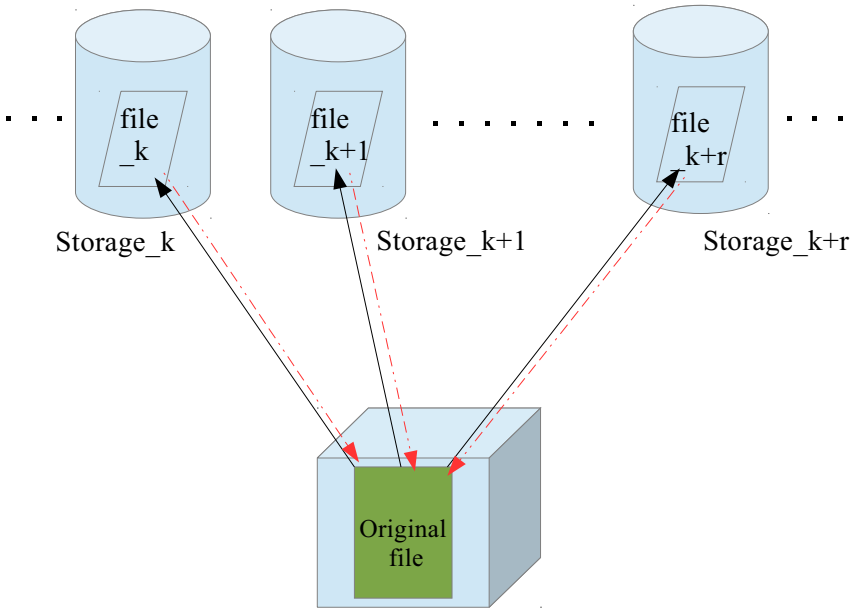

Figure 1: A computer program divides the original file into $n \geq 3$ container files.
It could be recovered by combing any three different container files.

To view the distributive storage from the point of technology structuring, our proposal meets the standard of Reliability since the number of storage (e.g. 10) far exceeds 3 which is the number of the container files need. So the system could still function as usual even the storage fails. Moreover, because of the fact that any single container file only contains incomplete information of the original file, we need not worry about information leakage if the storage or the network is intruded. Therefore system security is greatly improved.

By practical operation, we could make use of the quadratic functions to produce the container files and recover the original file. The advantage is: it is easy to be implemented and detect errors during the operation. On the other hand, owing to the fact that the recovery process is invariant under permutations, we could further encrypt the container files to improve the security.

We will introduce mathematical concepts which are relevant to the design and implementation of the systems in chapter 2. In chapter 3, we will introduce how to

producing container files and the recovery of the original files and at the same time, prove the feasibility of these methods. In chapter 4, we will introduce the multi-layer encryption method utilizing the permutation functions to encrypt the container files and prove the recovery method is invariant under permutation. Besides, for the decimal encryption, we will show how the Ruffini theorem to calculate degree or period of a permutation. Chapter 5 is the implementation and its results. Chapter 6, the conclusion and prospects will be provided. The last part is the references.

This proposal is originated by the author and her supervisor. No other similar ideas were found so far.

Chapter 2: Mathematical Foundations

This project introduces the division and recovery methods of an original file, the encoding and decoding of the container files and the feasibility on practical operation. In this chapter, we are going to introduce the relative mathematical concepts. First we will introduce some basic properties of polynomial functions which are relate to dividing and recovering the original file. Especially, we need the facts $n+1$ distinct correspondences of a polynomial function of degree n uniquely determines the polynomial function and calculation of the y-intercept of quadratic Newton polynomial. They are the keys of proposed distributive storage and recovery methods. Finally, we also introduce permutations and the Ruffini theorem which is the backbone of encryption method employed in this project.

2.1 Polynomial Functions of Degree n

We are going to introduce some elementary properties of polynomial functions [3].

Definition 2.1.1: Let f be a function from to . A real number x_0 is a zero of f if $f(x_0)=0$.

Definition 2.1.2: Let f be a function from to . If there exist a sequence of numbers $a_0, a_1, \cdots, a_n$ such that for all $x \in$, we have

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

then f is called a polynomial functions and the real numbers $a_0, a_1, \cdots, a_n$ are called the coefficients of f. If $a_n \neq 0$, then we call n the degree of the polynomial function f, denoted $deg(f)$. So f is said to be a polynomial function of degree n. In particular when $n=2$, f is said to be a quadratic function.

Theorem 2.1.3: If $f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ is a polynomial function with coefficients $a_0,a_1,\cdots,a_n$ such that f has more than n distinct zeros, then for any $x\in$, we always have

$$f(x)=0$$

and $a_0=a_1=\cdots=a_n=0$.

Proof： Suppose that $f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ such that f has $n+1$ distinct zeros $\alpha_1,\alpha_2,\cdots,\alpha_n,\alpha_{n+1}$, then

$$f(x)=a_n(x-\alpha_1)\cdots(x-\alpha_n).$$

Therefore , we have

$$f(\alpha_{n+1})=a_n(\alpha_{n+1}-\alpha_1)\cdots(\alpha_{n+1}-\alpha_n)=0.$$

Since $a_0,a_1,\cdots,a_n$ are distinct, we have $(\alpha_{n+1}-\alpha_1)\cdots(\alpha_{n+1}-\alpha_n)\neq 0$. So $a_n=0$ and we have all any real number x, we have

$$f(x)=0$$

and f can be represented as

$$f(x)=a_{n-1}x^{n-1}+\cdots+a_1x+a_0.$$

Using the same argument above, we have $a_{n-1}=0$ and inductively, we are able to obtain

$$a_{n-2}=\cdots=a_1=a_0=0.\ \blacklozenge$$

Lemma 2.1.4: A polynomial function of degree n has almost n distinct zero.

Lemma 2.1.5: Assume that functions f and g are polynomial functions of degree n and they agree with each other on $n+1$ distinct points, then $f=g$ and their coefficients are also equal.

Proof： Let $f(x)=a_nx^n+\cdots+a_1x+a_0$ and $g(x)=b_nx^n+\cdots+b_1x+b_0$. Define

$$F(x)=f(x)-g(x)=(a_n-b_n)x^n+\cdots+(a_1-b_1)x+(a_0-b_0).$$

Then the polynomial function F which is of degree n has $n+1$ distinct zero and by Theorem 2.1.3, we have

$$f(x) = 0$$

and

$$a_n - b_n = \cdots = a_1 - b_1 = a_0 - b_0 = 0 \text{ .♦}$$

Lemma 2.1.6: Coefficients of polynomial functions of degree n are unique.

Proof： Let $g(x) = f(x)$ in the Lemma 2.1.5 and the result follows♦

2.2 Newton polynomials

Let $(x_0, y_0), (x_1, y_1)$ and (x_2, y_2) be three points on the plane ^{2} such

that x_0, x_1 and x_2 are distinct. We are going to construct a quadratic function f passing through these points. First, let f be of the following form:

$$f(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

where a_0, a_1 and a_2 are unknown constant. Since $f(x_k) = y_k$ ($k = 1, 2, 3$), we have

$$\begin{aligned} f(x_0) &= a_0 \\ f(x_1) &= a_0 + a_1(x_1 - x_0) \\ f(x_2) &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \end{aligned}$$

and

$$a_0 = y_0, \quad a_1 = \frac{y_1 - y_0}{x_1 - x_0} \text{ and } a_2 = \frac{1}{x_2 - x_0}\left[\frac{y_2 - y_0}{x_2 - x_0} - \frac{y_1 - y_0}{x_1 - x_0}\right].$$

So we are able to have the Newton polynomial [4]

$$f(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + \frac{1}{x_2 - x_0}\left[\frac{y_2 - y_0}{x_2 - x_0} - \frac{y_1 - y_0}{x_1 - x_0}\right](x - x_0)(x - x_1)\cdots(2.2.1)$$

When $x = 0$, we obtain

$$f(0) = y_0 - \frac{y_1 - y_0}{x_1 - x_0}(x_0) + \frac{x_0 x_1}{x_2 - x_0}\left[\frac{y_2 - y_0}{x_2 - x_0} - \frac{y_1 - y_0}{x_1 - x_0}\right]\cdots(2.2.2)$$

which is the y-intercept of the quadratic function f .

2.3 Permutations and Ruffini Theorem

Definition 2.3.1： Let σ be a function from $\{1,2,3,\ldots,M\}$ to itself. We call σ a permutation on the set if σ is a one to one function. (i.e. for any $j_1, j_2 \in \{1,2,3,\ldots,M\}$. If $\sigma(j_1)=\sigma(j_2)$, we always have $j_1 = j_2$.) We also define

S_M is the set of all the permutations on $\{1,2,\cdots,M\}$.

Note that there are $M!$ permutations on $\{1,2,3,\ldots,M\}$. Suppose that $M=10$. There are totally 3628800 permutations on the set $\{1,2,3,\ldots,10\}$.

Theorem 2.3.2 : Let $\sigma \in S_M$ and $j \in \{1,2,3,\ldots,M\}$. Let

$$\sigma^0(j) = j$$

and

$$\sigma^{k+1}(j) = \sigma\left(\sigma^k(j)\right)$$

where $k = 0,1,2,\cdots$, Then there is the smallest number positive number $k_{\min}$ such that

$$k_{\min} \le M$$

and

$$\sigma^{k_{\min}}(j) = j \text{。}$$

We call $k_{\min}$ is the period of permutation σ at j and it is also denoted by T_j.

Proof: If there is an $k \in \{1,2,\cdots,M-1\}$ such that

$$\sigma^k(j) = j,$$

then let $k_{\min} = k$.

Therefore, assume that for any $k = 1,2,\cdots,M-1$,

$$\sigma^k(j) \neq j \text{。}$$

If $1 \le k_1 \le k_2 \le M-1$ such that

$$\sigma^{k_1}(j) = \sigma^{k_2}(j),$$

then $\sigma^{k_2-k_1}(j) = \sigma^0(j) = j$. Since $0 \le k_2 - k_1 \le M-2$, $k_2 = k_1$. So

$\sigma^1(j), \cdots, \sigma^{M-1}(j)$ are all distinct. Since σ is a permutation, we have

$\sigma^M(j) = j$. i.e. $k_{\min} = M$. ♦

Now we are ready to show the Ruffini Theorem [5].

Theorem 2.3.2 (P. Ruffini 1799) : Let $\sigma \in S_M$ and let T_σ be the smallest positive integer such that for any $j = 1, 2, \cdots, M$,

$$\sigma^T(j) = j.$$

Then T_σ is the LCM (The least common multiple) of the period $T_1, T_2, \cdots, T_M$ of the permutation σ at $1, 2, \cdots M$ respectively. Moreover T_σ is called the degree of the permutation σ.

Proof: Let $\hat{T}$ be the LCM of the periods $T_1, T_2, \cdots, T_M$. For any j and a positive integer m, we have

$$\sigma^{mT_j}(j) = \sigma^{(m-1)T_j + T_j}(j) = \sigma^{(m-1)T_j}(\sigma^{T_j}(j)) = \sigma^{(m-1)T_j}(j).$$

So $\sigma^{mT_j} = \sigma^0(j) = j$. Since $\hat{T}$ is the LCM of $T_1, T_2, \cdots, T_M$, we obtain

$$\sigma^{\hat{T}}(j) = j$$

where $j = 1, 2, \cdots, M$. So $T_\sigma \le \hat{T}$.

For any $j = 1, 2, \cdots, M$, we have $\sigma^{T_\sigma}(j) = j$. Therefore, $T_j \le T_\sigma$. By long division,

$$T_\sigma = m_j T_j + r_j$$

where m_j, r_j are integers such that $0 < m_j$ and $0 \le r_j < T_j$. So

$$\sigma^{T_\sigma}(j) = \sigma^{m_j T_j + r_j}(j) = \sigma^{r_j}\left(\sigma^{m_j T_j}(j)\right) = \sigma^{r_j}(j).$$

Since $\sigma^{T_\sigma}(j) = j$ and by the definition of period, we get $r_j = 0$. So, for any $j = 1, 2, \cdots, M$, T_σ is a multiple of T_j and hence T_σ is a common multiple of $T_1, T_2, \cdots, T_M$. We finally have $\hat{T} \le T_\sigma$.♦

Given $\sigma \in S_M$, let $j_1 = 1$ and

$$A_1 = \{\sigma^k(j_1) \mid k = 0, 1, \cdots, T_{j_1}\}.$$

Let $j_2 \le M$ such that it is the smallest positive integer which is not in A_1 and let

$$A_2 = \{\sigma^k(j_2) \mid k = 0, 1, \cdots, T_{j_2}\}.$$

Now, let $j_3 \le M$ such that it is the smallest positive integer which is not in A_1 and A_2 and let

$$A_3 = \{\sigma^k(j_3) \mid k = 0, 1, \cdots, T_{j_3}\}.$$

Continue this construction inductively. Note that the number of elements in A_i is at least one. So we can only have a finite sequence of A_i since $\{1, 2, \cdots, M\}$ is a finite set. So suppose that we have a finite sequence of sets $A_1, A_2, \cdots, A_r$ and the size of the set A_i is $T_{j_i}\left(i = 1, 2, \cdots, r\right)$. Then $M = T_{j_1} + T_{j_2} + \cdots + T_{j_r}$ and T_σ is the LCM of $T_{j_1}, T_{j_2}, \cdots, T_{j_r}$. So we define the pattern of the permutation σ as $\left(T_{j_1}\right)\left(T_{j_2}\right)\cdots\left(T_{j_r}\right)$.

Example 2.3.3 : Consider S_{10} and without loss of generality, if

$\sigma = (T_{j_1})(T_{j_2})\cdots(T_{j_r})$, $T_{j_1}, T_{j_2}, \cdots, T_{j_r}$ is in decreasing order. So the permutations on the set $\{1, 2, \cdots, 10\}$ such that $T_1 \geq 5$ are listed as below:

(10)
(9)(1)
(8)(2),(8)(1)(1)
(7)(3),(7)(2)(1),(7)(1)(1)(1)
(6)(4),(6)(3)(1),(6)(2)(2),(6)(2)(1)(1),(6)(1)(1)(1)(1)
(5)(5),(5)(4)(1),(5)(3)(2),(5)(3)(1)(1),(5)(2)(2)(1),(5)(2)(1)(1)(1),(5)(1)(1)(1)(1)(1)

It is clear that permutations of the pattern (5)(3)(2) have maximum degree. When $T_1 < 5$, possible periods are $1, 2, 3$ and 4. Their LCM is only 12. Therefore, among the permutations S_{10}, permutations with pattern (5)(3)(2) has over all maximum degree which is 30. For example

j	1	2	3	4	5	6	7	8	9	10
$\sigma(j)$	2	3	4	5	1	7	8	6	10	9

Then the degree of σ is $T_\sigma = 30$ and the permutation σ can be represented by $(12345)(678)(9\ 10)$.

Chapter 3: Applications of Quadratic Functions to Distributive Storages

In this chapter, we will engage in investigation of how to utilize the fact, “three distinct points on the plane can uniquely determine a quadratic function”, to design a model of distributive storage. We suppose that a file is a binary string with even number of bits. It is because the size of a file is counted by ‘byte’ and 1 byte equals to 8 bits. First, we have to find out the way which maps a bit pair in the binary string to a quadratic function. Then we could make use of these quadratic functions to produce the container files and save them into the different storages respectively. This process is called the encoding of the container files. If we need to recover the original file, we only need to utilize container files from any 3 different storages, and apply Newton polynomials to the y-intercepts of the correspondent quadratic functions, and then we could get the correspondent bit pair and recover the original file.

3.1 Bit Pairs and Their Correspondent Quadratic Functions

Let m_1 and m_2 be real numbers with $1 < m_1 < m_2$. We call the ordered pair (m_1, m_2) a key pair of the encoding method. We also call $\alpha\beta$ a bit pair if α and β take values either 0 or 1. Let $\alpha\beta$ be a bit pair. We define a quadratic function $f_{\alpha\beta}$ as

$$f_{\alpha\beta}(x) = (x - m_1^{\alpha})(x - m_2^{\beta})$$

where x is in . So we obtain the following table:

α	β	$f_{\alpha\beta}(x)$	$f_{\alpha\beta}(0)$
0	0	$f_{00}(x) = (x-1)(x-1)$	$f_{00}(0) = 1$
1	0	$f_{10}(x) = (x-m_1)(x-1)$	$f_{10}(0) = m_1$
0	1	$f_{01}(x) = (x-1)(x-m_2)$	$f_{01}(0) = m_2$
1	1	$f_{11}(x) = (x-m_1)(x-m_2)$	$f_{11}(0) = m_1 m_2$

Since $1, m_1, m_2$ and $m_1 m_2$ are distinct, We can observe that there is an one to one corresponding relation between a bit pair $\alpha\beta$ and the y-intercept of $f_{\alpha\beta}(x), f_{\alpha\beta}(0)$

Therefore, for fixed key pair (m_1, m_2), we can define a look up table with respect to

the key pair (m_1, m_2) as in Table 1

Table 1: Look up table

α	β	$f_{\alpha\beta}(0)$
0	0	1
1	0	m_1
0	1	m_2
1	1	$m_1 m_2$

where $f_{\alpha\beta}(x) = (x - m_1^{\alpha})(x - m_2^{\beta})$.

Example 3.1.1 Let $m_1 = 2$ and $m_2 = 3$. Then the look up table with respect to the key pair $(2,3)$ is

α	β	$f_{\alpha\beta}(0)$
0	0	1
1	0	2
0	1	3
1	1	6

and $f_{\alpha\beta}(x) = (x - 2^{\alpha})(x - 3^{\beta})$.

3.2 Creating Contianers

Let N be a positive integer. s is a binary string with length $2N$ if

$$s = \alpha_1\beta_1\alpha_2\beta_2\cdots\alpha_N\beta_N$$

Where α_k and β_k have value either 0 or 1.

Given a fixed key pair (m_1, m_2) and M distinct real numbers $c_1, c_2, \cdots, c_M$ with $M > 3$. For any binary string with length $2N$,

$$s = \alpha_1\beta_1\alpha_2\beta_2\cdots\alpha_N\beta_N \ ,$$

We define a sequence of container with respect to the binary string s, the key pair (m_1, m_2) and the positive integer M to be a sequence of N by 2 matrices $C_1, C_2, \cdots, C_M$ such that

$$C_j = \begin{bmatrix} c_j & f_1(c_j) \\ c_j & f_2(c_j) \\ \vdots & \vdots \\ c_j & f_N(c_j) \end{bmatrix}$$

where $j = 1, 2, \cdots, M$ and $f_k = f_{\alpha_k \beta_k}$.

Example 3.2.1: Let $(m_1, m_2) = (2,3)$, $M = 5$ and $c_j = j$ $(j = 1,2,3,4,5)$. If a binary string s is $\alpha_1\beta_1\alpha_2\beta_2\alpha_3\beta_3\alpha_4\beta_4 = 00101101$, then the j^{th} container of s, C_j, is a 4×2 matrix as below:

$$C_j = \begin{bmatrix} j & f_1(c_j) \\ j & f_2(c_j) \\ j & f_3(c_j) \\ j & f_4(c_j) \end{bmatrix} = \begin{bmatrix} j & (j-1)^2 \\ j & (j-2)(j-1) \\ j & (j-2)(j-3) \\ j & (j-1)(j-3) \end{bmatrix}$$

where $j = 1,2,3,4,5$. So we have

$$C_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 2 \\ 1 & 0 \end{bmatrix}, C_2 = \begin{bmatrix} 2 & 1 \\ 2 & 0 \\ 2 & 0 \\ 2 & -1 \end{bmatrix}, C_3 = \begin{bmatrix} 3 & 4 \\ 3 & 2 \\ 3 & 0 \\ 3 & 0 \end{bmatrix}, C_4 = \begin{bmatrix} 4 & 9 \\ 4 & 6 \\ 4 & 2 \\ 4 & 3 \end{bmatrix}, C_5 = \begin{bmatrix} 5 & 16 \\ 5 & 12 \\ 5 & 6 \\ 5 & 8 \end{bmatrix}$$

Note that there is a -1 in the second container

$$C_2 = \begin{bmatrix} 2 & 1 \\ 2 & 0 \\ 2 & 0 \\ 2 & -1 \end{bmatrix}.$$

To eliminlate negative number from containers, we can redefine

$$C_j = \begin{bmatrix} j & f_1(c_j) \\ j & f_2(c_j) \\ j & f_3(c_j) \\ j & f_4(c_j) \end{bmatrix} = \begin{bmatrix} j & (j-1)^2 + 1 \\ j & (j-2)(j-1) + 1 \\ j & (j-2)(j-3) + 1 \\ j & (j-1)(j-3) + 1 \end{bmatrix}.$$

Basically, it does not change anything of the recovery method introduced in next section.

3.3 Recovery Method

Let (m_1, m_2) be a key pair and let C_{j_1}, C_{j_2} and C_{j_3} be three different containers of a binary string s. We shall apply the recovery method defined as below to recover the binary s from the containers C_{j_1}, C_{j_2} and C_{j_3}.

So given three distinct containers of a binary string s namely C_{j_1}, C_{j_2} and C_{j_3} such that

$$C_{j_1} = \begin{bmatrix} c_{j_1} & f_1(c_{j_1}) \\ c_{j_1} & f_2(c_{j_1}) \\ \vdots & \vdots \\ c_{j_1} & f_N(c_{j_1}) \end{bmatrix}, C_{j_2} = \begin{bmatrix} c_{j_2} & f_1(c_{j_2}) \\ c_{j_2} & f_2(c_{j_2}) \\ \vdots & \vdots \\ c_{j_2} & f_N(c_{j_2}) \end{bmatrix}, C_{j_3} = \begin{bmatrix} c_{j_3} & f_1(c_{j_3}) \\ c_{j_3} & f_2(c_{j_3}) \\ \vdots & \vdots \\ c_{j_3} & f_N(c_{j_3}) \end{bmatrix}$$

Let

$$K = [C_{j_1}, C_{j_2}, C_{j_3}].$$

Then the matrix K is said to be a collector matrix and the k^{th} row of K is denoted by K_k. Therefore,

$$K_k = [c_{j_1}, f_k(c_{j_1}), c_{j_2}, f_k(c_{j_2}), c_{j_3}, f_k(c_{j_3})].$$

Recovery method is defined by the following steps：

Step 01 Read the key pair (m_1, m_2) and input the collector matrix K.

Step 02 Let $k = 1$ and set a binary string $s_R = s_1 s_2 s_3 s_4 \cdots s_{2N-1} s_{2N}$ to be an zero string i.e. $s_1 = s_2 = s_3 = \cdots = s_{2N} = 0$.

Step 03 Read

$$K_k = [c_{j_1}, f_k(c_{j_1}), c_{j_2}, f_k(c_{j_2}), c_{j_3}, f_k(c_{j_3})].$$

Step 04 From K_k, we are able to read three distinct points

$$\left(c_{j_1}, f_k\left(c_{j_1}\right)\right), \left(c_{j_2}, f_k\left(c_{j_2}\right)\right) \text{ 及 } \left(c_{j_3}, f_k\left(c_{j_3}\right)\right)$$

and then apply formula 2.2.2

$$f_k(0) = f_k\left(c_{j_1}\right) - \frac{f_k\left(c_{j_2}\right) - f_k\left(c_{j_1}\right)}{c_{j_2} - c_{j_1}}(c_{j_1}) + \frac{1}{c_{j_3} - c_{j_1}}\left[\frac{f_k\left(c_{j_3}\right) - f_k\left(c_{j_1}\right)}{c_{j_3} - c_{j_1}} - \frac{f_k\left(c_{j_2}\right) - f_k\left(c_{j_1}\right)}{c_{j_2} - c_{j_1}}\right](c_{j_1})(c_{j_2}),$$

to find the y-intercept of a function f_k which passes through these points.

Step 05 According to the look up table for the key pair (m_1, m_2), find the corresponding bit pair $\alpha\beta$ by using the y-intercept obtained in Step 04.

Step 06 Let $s_{2k-1} = \alpha$ and $s_{2k} = \beta$.

Step 07 If $k = N$, then output the binary string s_R and end. Otherwise, let $k = k + 1$ and go to Step 03.

Example 3.3.1 Following previous example 3.2.1, the key pair is $(2,3)$ and $c_j = j$, $j = 1,2,3,4,5$. Now let $j_1 = 1$, $j_2 = 3$ and $j_3 = 4$. Then

$$C_{j_1} = C_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 2 \\ 1 & 0 \end{bmatrix}, C_{j_2} = C_3 = \begin{bmatrix} 3 & 4 \\ 3 & 2 \\ 3 & 0 \\ 3 & 0 \end{bmatrix}, C_{j_3} = C_4 = \begin{bmatrix} 4 & 9 \\ 4 & 6 \\ 4 & 2 \\ 4 & 3 \end{bmatrix}$$

and their collector matrix

$$K = \begin{bmatrix} 1 & 0 & 3 & 4 & 4 & 9 \\ 1 & 0 & 3 & 2 & 4 & 6 \\ 1 & 2 & 3 & 0 & 4 & 2 \\ 1 & 0 & 3 & 0 & 4 & 3 \end{bmatrix}.$$

Let $s_R = s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 = 00000000$ and read the first row

$$K_1 = \begin{bmatrix} 1 & 0 & 3 & 4 & 4 & 9 \end{bmatrix}.$$

We obtain three distinct points $\left(c_1, f_1(c_1)\right) = (1,0), \left(c_3, f_1(c_3)\right) = (3,4)$ and $\left(c_4, f_1(c_4)\right) = (4,9)$. Apply the formula 2.1.1, the y-intercept of f_1 is

$$\begin{aligned} f_1(0) &= -\frac{4}{3-1} + \frac{1}{4-3}\left[\frac{9}{4-1} - \frac{4}{3-1}\right]3 \\ &= 1 \end{aligned}.$$

According to the look up table for $(2,3)$,

α	β	$f(0)$
0	0	1
1	0	2
0	1	3
1	1	6

where $f(x) = (x-2^{\alpha})(x-3^{\beta})$, We have $\alpha\beta = 00$ and hence $f_1(x) = (x-2^0)(x-3^0)$. Therefore, let $s_1 = 0$ 及 $s_2 = 0$ and set $k = k+1$. Repeat above steps until $k > 4$. Hence, we have $s_3 = 1$ and $s_4 = 0$, $s_5 = 1$ and $s_6 = 1$ and $s_7 = 0$ and $s_8 = 1$. Finally, it output

$$s_R = 00101101$$

which is exactly the original binary string s.

Theorem 3.3.2: Let a key pair be (m_1, m_2) and let $c_1, c_2, \cdots, c_M$ be M distinct real numbers with $M > 3$. For binary string s of the form $s = \alpha_1\beta_1\alpha_2\beta_2\cdots\alpha_N\beta_N$ and $j = 1, 2, \cdots, M$, we define

$$C_j = \begin{bmatrix} c_j & f_1(c_j) \\ c_j & f_2(c_j) \\ \vdots & \vdots \\ c_j & f_N(c_j) \end{bmatrix}$$

where $f_k(x) = (x - m_1^{\alpha_k})(x - m_2^{\beta_k})$ $(k = 1, 2, \cdots, N)$. Then for any three distinct C_j, we can apply the recovery method to them in order to recover the original binary string $s = \alpha_1\beta_1\alpha_2\beta_2\cdots\alpha_N\beta_N$.

Proof: Let $C_{j_1}, C_{j_2}, C_{j_3}$ be three distinct containers and $k = 1, 2, \cdots, N$. Then the k^{th} their collector matrix K is

$$K_k = [c_{j_1}, f_k(c_{j_1}), c_{j_2}, f_k(c_{j_2}), c_{j_3}, f_k(c_{j_3})].$$

where $f_k(x) = (x - m_1^{\alpha_k})(x - m_2^{\beta_k})$. First, we obtain the y-intercept of the quadratic function which passes the following distinct points,

$$\left(c_{j_1}, f_k\left(c_{j_1}\right)\right), \left(c_{j_2}, f_k\left(c_{j_2}\right)\right) \text{ and } \left(c_{j_3}, f_k\left(c_{j_3}\right)\right),$$

by Newton polynomial. According to lemma 2.1.5 , such quadratic function is f_k and their y-intercepts are uniquely determined. Since

$$f_{00}(0) = 1 \qquad f_{10}(0) = m_1 \qquad f_{01}(0) = m_2 \qquad f_{00}(0) = m_1 m_2$$

And $1, m_1, m_2, m_1 m_2$ are distinct $(1 < m_1 < m_2 < m_1 m_2)$. After having $f_k(0)$, we are able to find $\alpha_k\beta_k$ from the look up table. ♦

Chapter 4: Permutation Encryption

We will discuss the methods to encrypt the container files and therefore improve the security of storage method. We will mainly apply properties of permutations which we introduced in chapter 2 and the Ruffini theorem for calculating the degree of a permutation. Knowing that the recovery method is permutation invariance, we learn that recovering every pair of bits is independent on the order of three distinct containers. Hence, we will introduce the multi-layer encryption in section 4.1. In section 4.2, we will talk about a second as well as a classical encryption method called decimal encryption. Basically, we just rearrange the numbers 1 to 9 by a permutation. Since a permutation is invertible, the inverse of the permutation can be used for decryption. Moreover, the Ruffini theorem can give us the pattern of a permutation which allows better encryption.

4.1 Permutation Invariance and Multi-layer Encryption

In section 2.2, Given three points $(x_0, y_0), (x_1, y_1)$ and (x_2, y_2) on the plane ^{2} such that x_0, x_1 and x_2 are distinct, then the Newton polynomial which passes through these three points is

$$f(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + \frac{1}{x_2 - x_0}\left[\frac{y_2 - y_0}{x_2 - x_0} - \frac{y_1 - y_0}{x_1 - x_0}\right](x - x_0)(x - x_1)$$

.

In fact, from the lemma 2.1.5, the Newton polynomial is the same regardless the order of these three points. Hence, we have the following theorem.

Theorem 4.1.1: Let (m_1, m_2) be a key pair and let $c_1, c_2, \cdots, c_M$ be M distinct real numbers with $M > 3$. Given N permutations on the set $\{1, 2, \cdots, M\}$, namely, $\sigma_1, \sigma_2, \cdots, \sigma_N$, for a binary string $s = \alpha_1\beta_1\alpha_2\beta_2\cdots\alpha_N\beta_N$ and $j = 1, 2, \cdots, M$, we define

$$C_{\sigma(j)} = \begin{bmatrix} c_{\sigma_1(j)} & f_1\left(c_{\sigma_1(j)}\right) \\ c_{\sigma_2(j)} & f_2\left(c_{\sigma_2(j)}\right) \\ \vdots & \vdots \\ c_{\sigma_N(j)} & f_N\left(c_{\sigma_N(j)}\right) \end{bmatrix}$$

where $f_k(x) = (x - m_1^{\alpha_k})(x - m_2^{\beta_k})$ $(k = 1, 2, \cdots, N)$. Then for three distinct $C_{\sigma(j)}$, we are able to use the recovery method in section 3.3 to recover the original binary string $s = \alpha_1\beta_1\alpha_2\beta_2\cdots\alpha_N\beta_N$。

Proof：One should refer to the above discussion and theorem 3.3.2 and we are done。

♦

In fact, since the length of the given binary string N is different from time to time. It is not practical if $\sigma_1, \cdots, \sigma_N$ have no relation among them. Therefore, we propose that we just fix a permutation σ and let $\sigma_k = \sigma^k$ where $k = 1, 2, \cdots, N$. According to section 2.3, we have the sequence $\sigma^1, \sigma^2, \cdots$ is periodical and its period can be determined by Ruffini Theorem. To raise the security level, the period is bigger and better.
If $N = 7 = 2 + 5$, then σ has maximum degree $10 = 5 \times 2$ and its pattern is (5)(2). If $N = 10 = 2 + 3 + 5$, then σ has maximum degree $30 = 5 \times 3 \times 2$ and therefore, its pattern is (5)(3)(2).

Example 4.1.2：Let $N = 10$. The number of all the permutation of the pattern (5)(3)(2) is 120960 and we list some of them below:

(0, 8, 6, 7, 2)(1, 4, 9)(3, 5)	(0, 4, 2, 7, 5)(3, 8, 6)(1, 9)	(1, 4, 2, 3, 5)(7, 8, 9)(0, 6)
(2, 4, 7, 6, 9)(0, 8, 5)(1, 3)	(0, 1, 2, 8, 6)(4, 9, 5)(3, 7)	(1, 7, 3, 4, 9)(0, 2, 8)(5, 6)
(0, 3, 7, 9, 2)(4, 5, 8)(1, 6)	(0, 5, 3, 2, 7)(6, 9, 8)(1, 4)	(0, 8, 2, 1, 3)(6, 7, 9)(4, 5)
(1, 6, 9, 4, 5)(0, 2, 8)(3, 7)	(1, 9, 6, 5, 3)(2, 4, 8)(0, 7)	(0, 3, 1, 2, 4)(7, 8, 9)(5, 6)
(0, 1, 8, 9, 4)(2, 5, 6)(3, 7)	(0, 4, 6, 5, 7)(2, 8, 3)(1, 9)	(2, 3, 8, 6, 7)(1, 9, 5)(0, 4)
(0, 2, 8, 4, 9)(1, 6, 5)(3, 7)	(0, 7, 4, 1, 2)(3, 9, 5)(6, 8)	(2, 9, 4, 8, 5)(0, 3, 1)(6, 7)
(0, 8, 9, 5, 4)(1, 6, 7)(2, 3)	(1, 2, 6, 8, 3)(0, 9, 4)(5, 7)	(0, 9, 1, 6, 4)(2, 5, 8)(3, 7)
(0, 2, 7, 8, 9)(1, 4, 3)(5, 6)	(2, 4, 3, 6, 5)(0, 1, 7)(8, 9)	(2, 3, 7, 9, 5)(0, 6, 1)(4, 8)
(0, 9, 5, 4, 7)(1, 8, 3)(2, 6)	(0, 3, 4, 6, 8)(2, 7, 5)(1, 9)	(1, 2, 5, 8, 4)(0, 9, 7)(3, 6)
(0, 2, 9, 5, 3)(1, 8, 4)(6, 7)	(1, 3, 9, 2, 5)(6, 7, 8)(0, 4)	(0, 5, 6, 7, 2)(1, 3, 9)(4, 8)
(1, 9, 6, 7, 3)(4, 5, 8)(0, 2)	(0, 6, 8, 4, 2)(1, 3, 9)(5, 7)	(0, 3, 8, 1, 5)(4, 7, 9)(2, 6)
(1, 6, 3, 7, 4)(0, 8, 2)(5, 9)	(0, 1, 5, 2, 9)(3, 7, 8)(4, 6)	(0, 5, 8, 1, 4)(3, 7, 9)(2, 6)
(0, 5, 1, 9, 8)(2, 3, 7)(4, 6)	(1, 9, 6, 8, 4)(0, 3, 2)(5, 7)	(0, 1, 8, 2, 6)(4, 7, 5)(3, 9)
(0, 9, 4, 2, 8)(1, 7, 5)(3, 6)	(1, 6, 3, 8, 9)(0, 2, 5)(4, 7)	(3, 6, 7, 5, 4)(0, 1, 9)(2, 8)
(0, 3, 9, 4, 1)(2, 5, 8)(6, 7)	(0, 6, 2, 9, 4)(3, 5, 8)(1, 7)	(0, 7, 5, 3, 8)(2, 4, 6)(1, 9)

(1, 7, 5, 6, 8)(2, 3, 9)(0, 4)
(0, 7, 6, 3, 5)(2, 4, 8)(1, 9)
(1, 8, 2, 3, 6)(0, 7, 4)(5, 9)
(1, 3, 7, 9, 5)(2, 8, 4)(0, 6)
(4, 6, 7, 5, 9)(2, 8, 3)(0, 1)
(0, 6, 8, 5, 9)(1, 2, 4)(3, 7)

Example 4.1.3 Let $\sigma = (124)(35)$. So

$$\begin{aligned}\sigma &= (124)(35)\\ \sigma^2 &= (142)(3)(5)\\ \sigma^3 &= (1)(2)(4)(35)\\ \sigma^4 &= (124)(3)(5)\end{aligned}.$$

Then applying the multi-layer encryption with respect to σ to the containers in example 3.2.1

$$C_1 = \begin{bmatrix}1 & 0\\ 1 & 0\\ 1 & 2\\ 1 & 0\end{bmatrix}, C_2 = \begin{bmatrix}2 & 1\\ 2 & 0\\ 2 & 0\\ 2 & -1\end{bmatrix}, C_3 = \begin{bmatrix}3 & 4\\ 3 & 2\\ 3 & 0\\ 3 & 0\end{bmatrix}, C_4 = \begin{bmatrix}4 & 9\\ 4 & 6\\ 4 & 2\\ 4 & 3\end{bmatrix}, C_5 = \begin{bmatrix}5 & 16\\ 5 & 12\\ 5 & 6\\ 5 & 8\end{bmatrix},$$

we have

$$C'_1 = \begin{bmatrix}2 & 1\\ 4 & 6\\ 1 & 2\\ 2 & -1\end{bmatrix}, C'_2 = \begin{bmatrix}4 & 9\\ 1 & 0\\ 2 & 0\\ 4 & 3\end{bmatrix}, C'_3 = \begin{bmatrix}5 & 16\\ 3 & 2\\ 5 & 6\\ 3 & 0\end{bmatrix}, C'_4 = \begin{bmatrix}1 & 0\\ 2 & 0\\ 4 & 2\\ 1 & 0\end{bmatrix}, C'_5 = \begin{bmatrix}3 & 4\\ 5 & 12\\ 3 & 0\\ 5 & 8\end{bmatrix}$$

4.2 Decimal Encryption

It has been a long time since adopting the re-arrangement of the alphabets and letters as an encoding method. For example, the typewriter code [6], the typist does not need to enter the alphabet directly, he types the respective upper-left button instead. Therefore the sentence

"I love you"

becomes

"8 o9f3 697"

We could revert the codes by typing the right-bottom buttons of "8 o9f3 697". Let $\sigma = (03579)(146)(28)$. Applying the decimal encryption to the containers in example 3.2.1

$$C_1=\begin{bmatrix}1&0\\1&0\\1&2\\1&0\end{bmatrix},C_2=\begin{bmatrix}2&1\\2&0\\2&0\\2&-1\end{bmatrix},C_3=\begin{bmatrix}3&4\\3&2\\3&0\\3&0\end{bmatrix},C_4=\begin{bmatrix}4&9\\4&6\\4&2\\4&3\end{bmatrix},C_5=\begin{bmatrix}5&16\\5&12\\5&6\\5&8\end{bmatrix},$$

We obtain

$$C_1'=\begin{bmatrix}4&3\\4&3\\4&8\\4&3\end{bmatrix},C_2'=\begin{bmatrix}8&4\\8&3\\8&3\\8&-4\end{bmatrix},C_3'=\begin{bmatrix}5&6\\5&8\\5&3\\5&3\end{bmatrix},C_4'=\begin{bmatrix}6&0\\6&1\\6&8\\6&5\end{bmatrix},C_5'=\begin{bmatrix}7&41\\7&48\\7&1\\7&2\end{bmatrix}。$$

Clearly, applying $\sigma^{-1}=(09753)(641)(28)$ to the encrypted containers, we can have the original containers back.

Futhermore, if we consider

$$\begin{aligned}\sigma&=(03579)(146)(28)\\\sigma^2&=(05937)(164)(2)(8)\\\sigma^3&=(07395)(1)(4)(6)(28)\\\sigma^4&=(09753)(146)(2)(8)\end{aligned},$$

we can use $\sigma,\sigma^2,\sigma^3,\sigma^4$ to the first, second third and fourth layers of the original containers respectively for decimal encryption to obtain

$$C''_1=\begin{bmatrix}4&3\\6&5\\1&8\\4&9\end{bmatrix},C''_2=\begin{bmatrix}8&4\\2&5\\8&7\\2&-4\end{bmatrix},C''_3=\begin{bmatrix}5&6\\7&2\\9&7\\0&9\end{bmatrix},C''_4=\begin{bmatrix}6&0\\1&4\\4&8\\6&0\end{bmatrix},C''_5=\begin{bmatrix}7&41\\9&62\\0&6\\3&8\end{bmatrix}$$

If we want to decrypt the decimal encrypted containers above, it can be done by applying $\sigma^{-1},\sigma^{-2},\sigma^{-3},\sigma^{-4}$ to the corresponding layers.

Definition 4.2.1：We call the permutation σ on the set $\{0,1,\cdots,9\}$ a decimal encryption permutation. Suppose that a decimal representation of a number is $a_1a_2\cdots a_R$, where for all $a_r\in\{0,1,\cdots,9\}$ and $a_1\neq 0$. Then define

$$T_\sigma(a_1a_2\cdots a_R)=\sigma(a_1)\sigma(a_2)\cdots\sigma(a_R).$$

4.3 Mixed Encryption Method

Example 4.3.1 Follow example 4.1.3. After multi-layer encryption, the resulted containers are

$$C'_1=\begin{bmatrix}2&1\\4&6\\1&2\\2&-1\end{bmatrix},C'_2=\begin{bmatrix}4&9\\1&0\\2&0\\4&3\end{bmatrix},C'_3=\begin{bmatrix}5&16\\3&2\\5&6\\3&0\end{bmatrix},C'_4=\begin{bmatrix}1&0\\2&0\\4&2\\1&0\end{bmatrix},C'_5=\begin{bmatrix}3&4\\5&12\\3&0\\5&8\end{bmatrix}$$

Let

$$\sigma=(03579)(146)(28)$$
$$\sigma^2=(05937)(164)(2)(8)$$
$$\sigma^3=(07395)(1)(4)(6)(28)$$
$$\sigma^4=(09753)(146)(2)(8)\ .$$

Apply the decimal encryption to C'_is, we have a sequence of fixed encrypted containers

$$C''_1=\begin{bmatrix}8&4\\1&4\\1&8\\2&-4\end{bmatrix},C''_2=\begin{bmatrix}6&0\\6&5\\8&7\\6&0\end{bmatrix},C''_3=\begin{bmatrix}7&41\\7&2\\0&6\\0&9\end{bmatrix},C''_4=\begin{bmatrix}4&3\\2&5\\4&8\\4&9\end{bmatrix},C''_5=\begin{bmatrix}5&6\\9&62\\9&7\\3&8\end{bmatrix}.$$

4.4 Quadratic Encrypted Distributive Storage and Decrypted Recovery Algorithms

Let's fix a key pair $(m_1,m_2)=(2,3)$, the number of the storages M and a permutation σ_{num} on the set $\{0,1,\cdots,9\}$ for decimal encryption.

Encrypted Distributive Storage Algorithm

Step 01 Read $s=\alpha_1\beta_1\alpha_2\beta_2\cdots\alpha_N\beta_N$ and let $k=1$;

Step 02 Generate a permutation σ on the set $\{1,2,\cdots,M\}$ for multi-layer encryption.

Step 03 Let $f(x)=(x-2^{\alpha_k})(x-3^{\beta_k})+1$;

% To eliminate the only negative value -1.

Step 04 j=1 ;

%Steps 04-07，performing mixed encrypted distributive storage.

Step 05 $C_j(k,1)=\sigma^k(j), C_j(k,2)=f(\sigma^k(j))$；

%Multi-layer encryption

Step 06 $C_j(k,1)=T_{\sigma_{num}^k}\left(C_j(k,1)\right), C_j(k,2)=T_{\sigma_{num}^k}\left(f(\sigma^k(j))\right)$；

% decimal encryption

Step 07 If $j=M$, then go to Step 08；Otherwise, go to Step 05；

Step 08 If $k=N$, then go to Step 09；Otherwise, go to Step 03；

Step 09 j=1；

Step 10 Store C_j to the j^{th} storage；

Step 11 If $j=M$, then quit；Otherwise, go to Step 09.

Decrypted Recovery Algorithms

Step 01 Read three different containers C_{j_1}, C_{j_2} and C_{j_3} of a binary string s such that

$$C_{j_1}=\begin{bmatrix} c_{j_1}(1,1) & c_{j_1}(1,2) \\ c_{j_1}(2,1) & c_{j_1}(2,2) \\ \vdots & \vdots \\ c_{j_1}(N,1) & c_{j_1}(N,2) \end{bmatrix}, C_{j_2}=\begin{bmatrix} c_{j_2}(1,1) & c_{j_2}(1,2) \\ c_{j_2}(2,1) & c_{j_2}(2,2) \\ \vdots & \vdots \\ c_{j_2}(N,1) & c_{j_2}(N,2) \end{bmatrix}, C_{j_3}=\begin{bmatrix} c_{j_3}(1,1) & c_{j_3}(1,2) \\ c_{j_3}(2,1) & c_{j_3}(2,2) \\ \vdots & \vdots \\ c_{j_3}(N,1) & c_{j_3}(N,2) \end{bmatrix}$$

Step 02 Let

$$K=\begin{bmatrix} c_{j_1}(1,1) & c_{j_1}(1,2) & c_{j_2}(1,1) & c_{j_2}(1,2) & c_{j_3}(1,1) & c_{j_3}(1,2) \\ c_{j_1}(2,1) & c_{j_1}(2,2) & c_{j_2}(2,1) & c_{j_2}(2,2) & c_{j_3}(2,1) & c_{j_3}(2,2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{j_1}(N,1) & c_{j_1}(N,2) & c_{j_2}(N,1) & c_{j_2}(N,2) & c_{j_3}(N,1) & c_{j_3}(N,2) \end{bmatrix}.$$

% The matrix K is collector matrix. It's k^{th} row is denoted by K_k. Therefore,

% $K_k = [c_{j_1}(k,1), c_{j_1}(k,2), c_{j_2}(k,1), c_{j_2}(k,2), c_{j_3}(k,1), c_{j_3}(k,2)]$。

Step 03 Let $k = 1$ and set a binary string $s_R = s_1 s_2 s_3 s_4 \cdots s_{2N-1} s_{2N}$ to be an zero string.

Step 04 Read the k^{th} row of the collector matrix K

$$K_k = [c_{j_1}(k,1), c_{j_1}(k,2), c_{j_2}(k,1), c_{j_2}(k,2), c_{j_3}(k,1), c_{j_3}(k,2)]$$

Step 05

$$K_k = [T_{\sigma_{num}^{-k}}\left(c_{j_1}(k,1)\right), T_{\sigma_{num}^{-k}}\left(c_{j_1}(k,2)\right), T_{\sigma_{num}^{-k}}\left(c_{j_2}(k,1)\right), T_{\sigma_{num}^{-k}}\left(c_{j_2}(k,2)\right), T_{\sigma_{num}^{-k}}\left(c_{j_3}(k,1)\right), T_{\sigma_{num}^{-k}}\left(c_{j_3}(k,2)\right)]$$

% Decimal decryption for K_k.

Step 06 From K_k, we have three distinct points

$\left(c_{j_1}(k,1), c_{j_1}(k,2)\right), \left(c_{j_2}(k,1), c_{j_2}(k,2)\right)$ and $\left(c_{j_3}(k,1), c_{j_3}(k,2)\right)$. Then apply formula 2.2.

$$f_k(0) = c_{j_1}(k,2) - \frac{c_{j_2}(k,2) - c_{j_1}(k,2)}{c_{j_2}(k,1) - c_{j_1}(k,1)} c_{j_1}(k,1) + \left[\frac{c_{j_3}(k,2) - c_{j_1}(k,2)}{c_{j_3}(k,1) - c_{j_1}(k,1)} - \frac{c_{j_2}(k,2) - c_{j_1}(k,2)}{c_{j_2}(k,1) - c_{j_1}(k,1)} \right] \frac{c_{j_1}(k,1) c_{j_2}(k,1)}{c_{j_3}(k,1) - c_{j_1}(k,1)},$$

To obtain the y-intercept of the quadratic polynomial passes through these points.

Step 07 According to the look up table of the key pair $(2,3)$. We are able to find the bit pair $\alpha\beta$ corresponding to the y-intercept from step 04.

Step 08 Let $s_{2k-1} = \alpha$ and $s_{2k} = \beta$.

Step 09 If $k = N$, then out the binary string s_R and quit. Otherwise, let $k = k + 1$ go to step 04.

Note that the look up table of $(2,3)$ is

α	β	$f_{\alpha\beta}(0)$
0	0	2

1	0	3
0	1	4
1	1	7

where $f_{\alpha\beta}(x) = (x - 2^{\alpha})(x - 3^{\beta}) + 1$.

Chapter 5: Implementation and Results of the Algorithms

5.1 Purpose and Methods of the Performance Tests

We wrote a “C program” named “cloud.exe” according to the quadratic distributive storage and decoding recovery calculation which are introduced in the last chapter. We will test the performance of the program with different types and sizes of files according to its running time and compression ratio.

We have two groups of files. The first group are randomly generated text files with sizes of 2, 4, 6, 8, 10 MB (megabyte). The second group is consisted of office files, images and pdf. First, we use cloud.exe to run on the two groups of files in the same computer, encode them with different encryption options and try to recover them with any 3 different container files, record its performance. The process would produce container files which are 8 times larger in size of the original file on regular base. Then we will compare the results after the compression of the files.

5.2 Review of the Result of Performance Tests

We could see from the data that the relation of the running time and the size of the original file represent a linear growth. Different encryption method would have different influence on the running time. Basically, multi-layer encryption uses less resource and therefore has almost no influence on the running time as we expected. Although decimal encryption and decryption is the principal part of the calculation, it has small influence on the average size of the container files after compression.

Let’s look at the compression ratio of the original file and the container files. In the first text files group, the size of the compressed container files randomly produced takes 60% of the size of the original file. In the second group, the result shows that the compression ratio has direct relation with the format of the original file. We could learn that image formats such as BMP, JPG would produce bigger container files, and DOC, TIF vice versa. The phenomenon may owe to the format of the original file: most of the images are compressed before division while text files may contain many markups to express its contents.

In the next section, we will present the trial results with tables and line charts.

5.3 The trial results

For the convenience of the discussion, in this chapter, a quadratic storage method is called a type 0 storage. Type 1, type 2and type 3 refer to a type 0 storage plus multi-layer encryption, decimal encryption or mix encryption respectively.

5.3.1 Speed Tests

a) Producing Containers

The following table shows the time needed for producing compressed containers of the first group for all types:

Size of the txt file (MB)	Type 0 (sec)	Type 1 (sec)	Type 2 (sec)	Type 3 (sec)
2	9.136	8.623	29.997	29.369
4	16.153	15.903	58.931	58.392
6	23.996	22.549	88.308	87.362
8	37.062	33.596	118.25	117.533
10	42.381	33.872	147.051	147.145

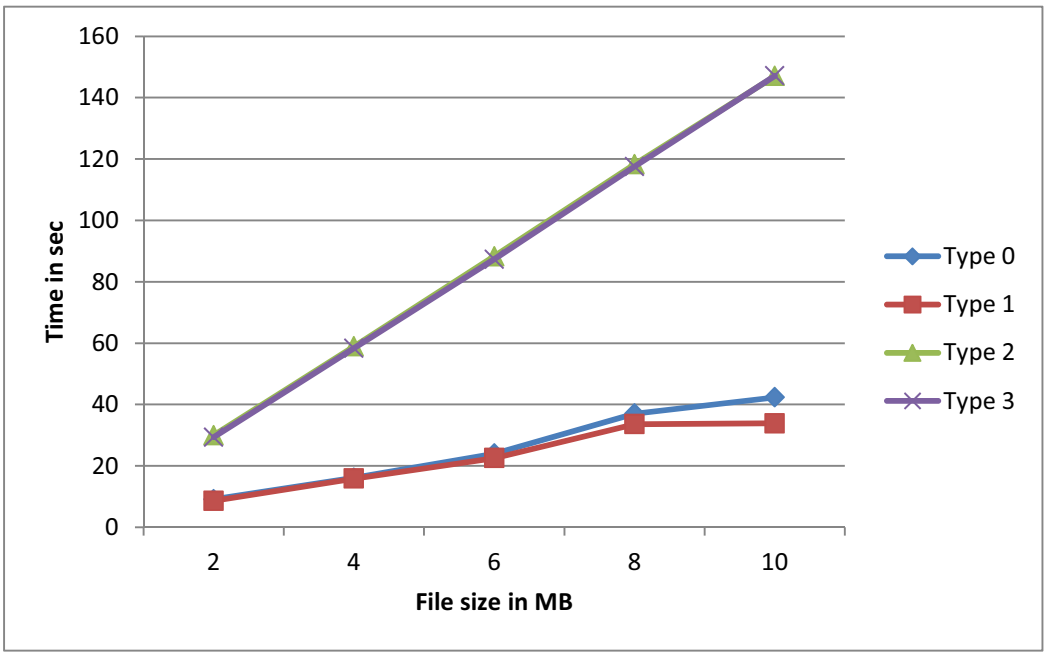

Figure 2: The container producing time needed for the first group and all types

b) Recovery the Original File

The following table shows the time needed for recovering original files from their containers in a) for all type of storages :

Size of the txt file (MB)	Type 0 (sec)	Type 1 (sec)	Type 2 (sec)	Type 3 (sec)
2	0.535	0.588	9.097	8.891
4	1.099	1.151	17.466	17.555
6	1.553	1.7	26.23	26.362
8	2.13	2.264	34.957	35.037
10	2.618	2.829	43.71	43.708

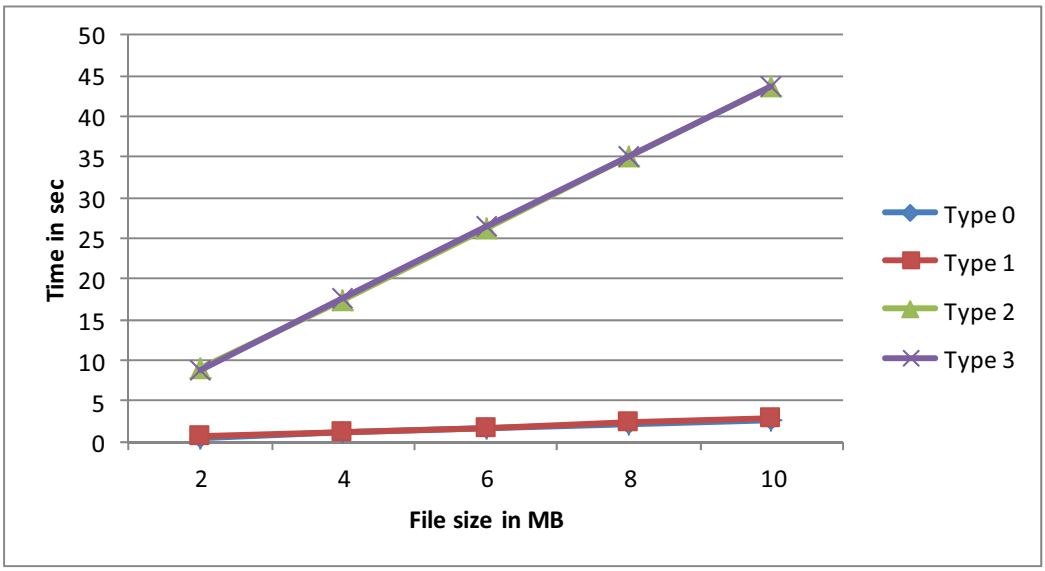

Figure 3: The time needed for recovering the original files from the container files for all type of storages.

5.3.2 Container Size Tests

a) The following table shows that average sizes of the compressed container files of the first group for all type of storages after produced :

Encryption options	2MB	4MB	6MB	8MB	10MB
Type 0	1,286	2,572	3,857	5,142	6,428
Type 1	1,282	2,564	3,844	5,125	6,407
Type 2	1,388	2,794	4,165	5,552	6,937
Type 3	1,368	2,731	4,094	5,458	6,821

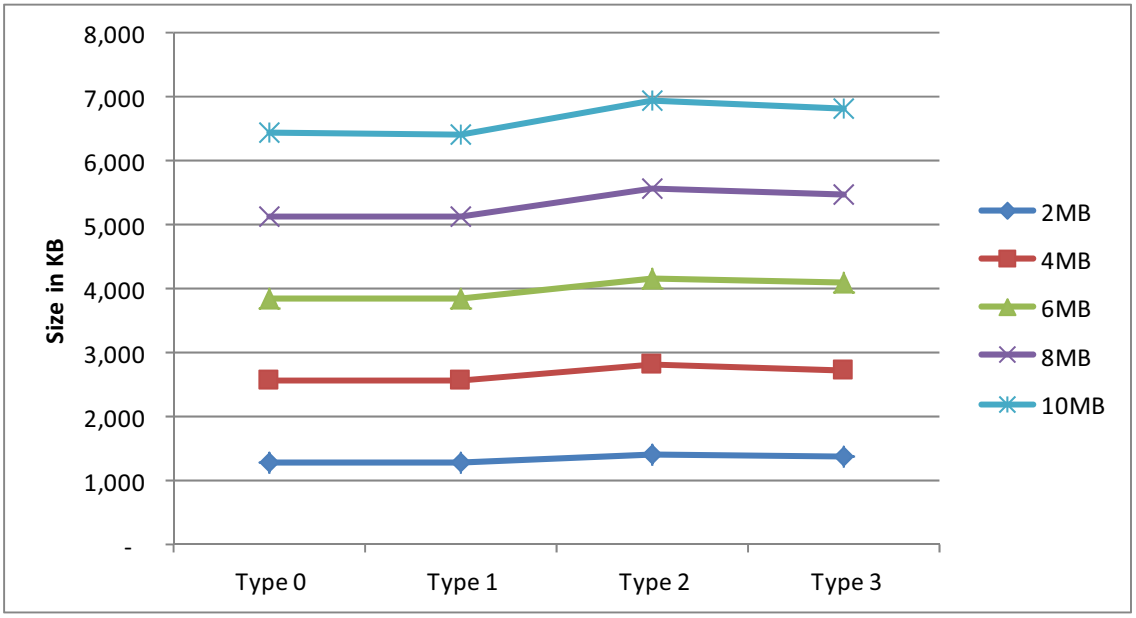

Figure 4: The average sizes of the container files of the first group after compression.

b) The following table shows the ratio of the compressed container files compare to the first group original files after all type of storage.

Encryption options	2MB	4MB	6MB	8MB	10MB
Type 0	62.80%	62.79%	62.78%	62.77%	62.77%
Type 1	62.60%	62.59%	62.57%	62.57%	62.56%
Type 2	67.76%	68.22%	67.79%	67.78%	67.74%
Type 3	66.78%	66.68%	66.64%	66.63%	66.62%

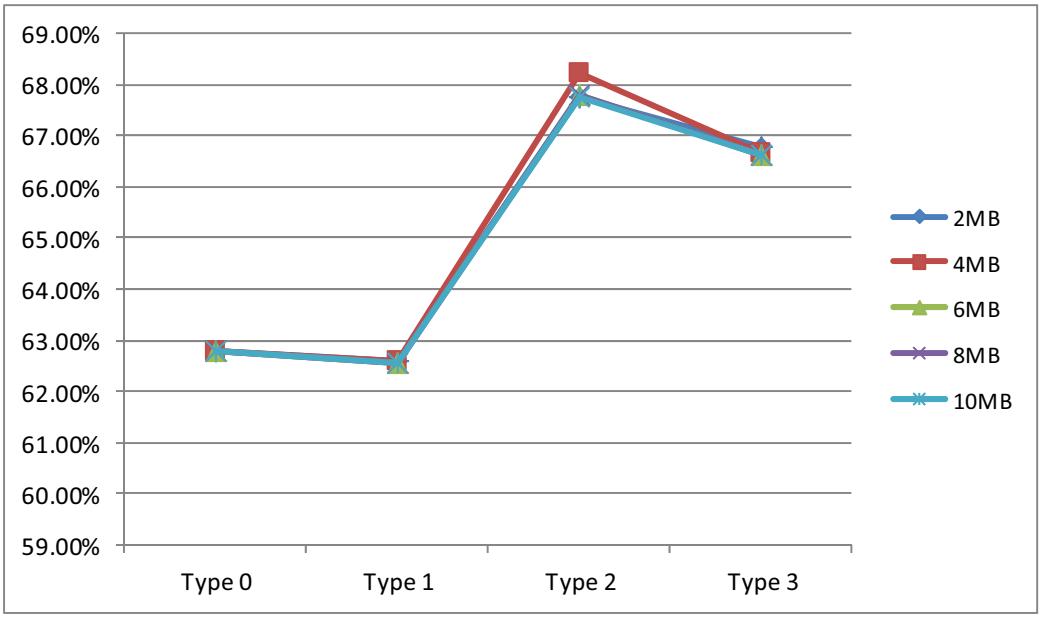

Figure 5: The ratio of the size of the compressed container files compare to the first group original files for all type of storages.

5.3. The Performance of Type 3 Storage for Various Common File Formats

The results are summarized in the table below:

Common Format	Size of the original files (KB)	Time for producing containers (sec)	Time for recovery (sec)	Average size of containers (KB)	Average size of containers With compression (KB)	Ratio of size of compressed container and the size of original file (%)
bmp	788.8	11.306	3.601	6310.8	1301.4	164.98%
jpg	841.5	11.931	3.723	6731.7	1070.0	127.15%
png	852.7	12.379	3.816	6821.9	1969.5	230.96%
tif	817.4	11.540	3.616	6539.2	2113.1	258.52%
doc	986.5	14.049	4.310	7892.0	806.3	81.74%
xls	1006.0	14.309	4.332	8048.0	513.8	51.07%
ppt	826.0	11.866	3.714	6688.0	779.9	93.29%
pdf	879.8	12.574	3.948	7,038.7	785.2	89.25%

Chapter 6: Conclusion and Prospects

6.1 Conclusion

This project studies the application of algebra to cloud computing, precisely, distributive storage.

On theory, first, we associate the bit pairs of a file with quadratic functions, then we use the fact “three distinct points on the plane can uniquely determine a quadratic function” to design distributive storage which enable us to divide the file into $n \geq 3$ other different container files. In reverse, we could recover the original file by making use of any three different container files. Second, by applying permutation to design the multi-layer encryption and decimal encryption method of container files, the security of distributive storage could be improved greatly. Third, the application of the Ruffini theorem in deciding the largest degree of permutation could provide convenience in programming and practical operation, and maintain a high level of security at the same time.

In practice, we use a “C program” to develop the distributive storage system which allows swift generation of container files and recovery of original file. Moreover, the compressed container files take only 68% of the size of the original file, which benefits the transfer process on internet.

Effectively combing practice and theory, the application of algebra in cloud computing provides us a better storage method of data.

6.1 Prospects

In Chapter 2, we relate the quadratic function $f_{\alpha\beta}(x) = (x - m_1^{\alpha})(x - m_2^{\beta})$ to the byte pair $\alpha\beta$. In fact, $f_{\alpha\beta}$ could be chosen different kind of function other quadratic polynomial as long as we could find that x_0 could satisfy the condition that $f_{00}(x_0)$, $f_{01}(x_0)$, $f_{10}(x_0)$ and $f_{11}(x_0)$ are different. As a result, the relative container files could be more complex and diversified. In addition, Let $x_0 = 0$ and let byte pair $\alpha\beta$ correspond to $f_{\alpha\beta}(\sin x)$ where $x \in [0, \pi/2]$. Therefore, we can also obtain the

similar distributive storage and encryption methods. However, $f_{\alpha\beta}(\sin x)$ is not a polynomial function.

In practice, both the distributive storage and encryption and decryption method are highly parallel. We could make use of the MPI protocol [7] to develop a cloud program which support parallel operation in order to increase the operation speed and reduce the running time. Generally, a double blade server in a computing cloud contains 25 CPUs, therefore shorten the operation time by 25 times.

Finally, we hope that there will be further development of this project both on theory design and practical operation.

Reference

1. Toffler, A. *"The third wave."* Bantam Books, 1989.

2. Mather, T., Kumaraswamy, S.and Latif, S. *"Cloud Security and Privacy:An Enterprise Perspective on Risks and Compliance."* O' Reilly Media, 2009.

3. Fine, H. B. " A College Algebra." Ginn & company, 1904.

4. Hildebrand, F. B. *"Introduction to Numerical Analysis."* New York: McGraw-Hill, 1956.

5. Gallian, J. *"Contemporary Abstract Algebra."* Brooks Cole, 6 edition, 2004.

6. Gardner, M. *"Codes, Ciphers and Secret Writing (Dover Children's Activity Books)."* Dover Publications, 1984.

7. Message Passing Interface (MPI), http://en.wikipedia.org/wiki/Message_Passing_Interface. (last modified on 28 July 2012)

TAN CHIH WEI

Gender: Female
Date of birth: 27 September 1996
Education: Pui Ching Middle School, Macau (Senior 2)

AWARDS

Year	Contest/Activity	Organizer	Achievement	Remarks
2012	Outstanding Student Award	Pui Ching Middle School, Macau	Prize winner	
2012	Scholarship sponsored by 鄭仕豪(鄭仕豪獎學金)	Pui Ching Middle School, Macau	Prize winner	
2012	Intel International Science and Engineering Fair	Society for Science & the Public (SSP)	Finalist	International contest
2011	The 26th China Adolescents Science and Technology Innovation Contest	China Association for Science and Technology 和 the People's Government of Inner Mongolia Region	Champion selected by Chairperson of the People's Government of Inner Mongolia Region	National contest
2011	Outstanding Student Award	Pui Ching Middle School, Macau	Prize winner	
2010	Macau's Top 10 Outstanding Teenager Competition	Young Men's Christian Association of Macau	Prize winner	
2010	The 8th Chinese Adolescents Mathematical Forum	China Youth Academy of Sciences	Winner	National contest
2010	The 5th Writing Contest for High School Students in China(Macau)	Macao Youth Federation	Outstanding Award	Selected as Macau representative
2010	St. Pius X School of Music Scholarship	Académia de Musica de S. Pius X	Prize winner (senior level)	Only one winner at each stage

2009	The 27th Macao Young Musicians Competition- piano duet (junior level)	Cultural Affairs Bureau	1st runner up	Open championship
2007	“World Heritage and Me”School Writing Contest in China	Chinese National Commission for UNESCO	Winner	National contest

Works

Tan, C.W., & 曾學為. (2012). Application of Principia Mathematica to Electronic access control system. *China Science Techonology Education*,2, 26-28.

Membership

1. 2012- member of Society for Science & the Public (SSP)

履历表

姓名：谭知微

性别：女

出生日期：1996/09/27

就读：澳门培正中学-高中二年级

所获奖项

年份	比赛 / 奖励名称	颁发机构	所获奖励名次	备注
2012	红蓝之光	澳门培正中学	得奖者	
2012	郑仕豪奖学金	澳门培正中学	得奖者	
2012	Intel 国际科学与工程赛	Society for Science & the Public (SSP)	Finalist	国际大赛
2011	第 26 届年全国青少年科技创新大赛	中国科协和内蒙古自治区人民政府	内蒙古自治区主席奖和一等金奖	全国奖项
2011	红蓝之光	澳门培正中学	得奖者	
2010	首届澳门十大杰出少年选举	澳门基督教青年会	十大杰出少年	
2010	第八届走进美妙的数学花园-中国青少年数学论坛	中国少年科学院	一等一名金奖	全国奖项
2010	第五届中国中学生作文比赛-澳区比赛	澳门青年联合会	优异奖	获选澳区代表
2010	圣庇护十世音乐院奖学金	圣庇护十世音乐院	高阶组得奖者	各阶选一人得奖
2009	第 27 届澳门青年音乐比赛-钢琴四手联弹初级	澳门文化局	第二名	公开赛奖项
2007	“我与世界遗产”中国校际作文征集活动	中国联合国教科文组织全国委员会	一等奖	全国奖项

著作

1. 谭知微 曾学为, ““众妙之门”——数学原理在电子门禁系统上的应用”, 《中国科技教

育》2012 年 第 2 期 26-28 页

会员

1. 2012- member of Society for Science & the Public (SSP)