出國報告（出國類別：國際會議）

加速三重模指數的方法

服務機關：國防大學理工學院電機電子系
姓名職稱：中校教師談光雄
派赴國家：杜拜
出國期間：104/01/28-104/02/02
報告日期：104/01/30

摘要

2015 年「數位訊號處理，資料擷取及無線通訊國際研討會」(The International Conference on Digital Information Processing, Data Mining, and Wireless Communications, DIPDMWC2015)，於 104 年 1 月 28 日至 30 日在阿拉伯聯合大公國杜拜伊斯蘭大學教育城舉行，本人投稿該研討會論文乙篇，論文題目：加速三重模指數的方法，因榮獲刊登及大會議程邀請於 1 月 30 日上午場次進行口頭發表，故於 1 月 28 日搭機前往與會。當日該場次會議中，計有來至台灣銘傳大學及萬能科大等七篇論文發表，期間發表人均詳細報告其研究成果，報告完後，台下與台上學者討論熱絡，彼此交流受益良多。

目次

頁次

壹、目的：

2015 年「數位訊號處理，資料擷取及無線通訊國際研討會(DIPDMWC2015)」，於 104 年 1 月 28 日至 30 日，由全球學生人數前三多的伊斯蘭大學位於阿拉伯聯合大公國分校舉行，該研討會主要探討的主題為：數位資料處理(Digital Information Processing)、資料探勘(Data Mining)及無線通訊(Wireless Communications)等資、通訊現代新科技主題，其屬於探討資、通訊於現代科技應用之國際學術研討會，而參與投搞發表之專家學者有來至美、日、韓、台灣、馬來西亞、奈及利亞及阿拉伯聯合大公國等各國學者專家，合計發表論文 31 篇。對於參加本次國際研討會，將使自己在學術方面及國際觀增廣見聞，學術上，可以思考其他專家學者所專研之領域是否可以與自己的領域相結合，進而產生新火花，創造出新思維，也對於未來研究與方向產生更大的動力，國際觀方面也可瞭解各國歷史及文化差異。

貳、過程：

會議議程

Session 7, January 30, 2015, Room A: 09:00 – 11:00

Chair: Ping Sheng Huang. Department of Electronic Engineering, Ming Chuan University, Taiwan.

本人發表之論文名稱：

A Strategy Speeds Up the Triple Modular Exponentiation

作者：

Te-Jen Chang, Kuang-Hsiung Tan(談光雄), Ping-Sheng Huang, Ching-Yin Chen, and I-Hui Pan.

(Chung-Cheng Institute of Technology, National Defense University, ROC)

本次赴阿拉伯聯合大公國杜拜參加國際研討會，因考量該國地處東亞，鄰國又有依斯蘭國恐怖局勢，為安全起見及考量當地風土民情，故延請旅行社代定訂團體機票及相關住宿事宜，以節省開支。研討會舉行時間為 104 年 1 月 28 日至 30 日，因台灣至杜拜直航班機已預訂一空，故由旅行社代訂 1 月 28 日經香港轉機阿布達比之機位，當日中午 12 時 50 分搭乘中華航空班機赴香港轉機，下午 18 時 15 分再搭阿提哈德航空與塞什爾航空聯營航班至阿拉伯聯合大公國阿布達比，抵達阿布達比時已逾當地時間晚間 11 時 50 分(台灣時間 29 日凌晨 3 時 50 分)，辦好出關手續及下榻抵達飯店後，已逾當地 29 日凌晨 2 時 30 分，由於發表之場次為 30 日上午，在加上時差問題，29 日當天上午在飯店稍作休息及準備次日發表之投影片等資料，下午則蒞臨會場旁聽其他專家學者發表之論文數篇。1 月 30 日當日早上 08:30 到達會場 Islamic Azad University 並順利完成報到手續，會議一開始，首先大會主席 Dr. Mohammas V. Malakooti 即頒發論文發表證明給每位發表者，接著再由會議主持人(Session Chair) Prof. Ping Sheng Huang 主持議程並由發表人逐一開始報告，本會議共發表七篇論文，分別由銘傳大學及萬能科技大學等教授及學者輪流發表，發表人均詳細報告其研究成果，報告完後，主持人亦提供時間給在場與會學者、專家提問，由於所研究之領域具相關性，因此台下與台上學者討論相當熱絡，導致會議主持人為控管時間而打斷討論，也因而會議延遲結束。本人發表之論文為第 6 順位，論文發表完後，主持人及台下學者亦提出見解及寶貴建議，對於本次參加國際研討會，使自身眼界更開闊及瞭解國際研究趨勢與脈動，因此對於未來研究與方向將產生更大的動力。

本人所發表之論文為 A Strategy Speeds Up the Triple Modular Exponentiation，報告內容摘要如下：在計算 $M^E \bmod N$時，最重要的部分就是模指數，此處的 M 意指明文，N 為模數，E 為大的指數。模指數演算法的執行效率是根據指數的模平方與模乘法的個數而定。本文提出的方法，它可以加速三重模指數演算法，並將時間複雜度降低為 1.875j的乘法量，此處的 j 為指數的位元長度。相關收錄論文如附件。

1 月 29 日，旁聽其他學者、專家發表議題，摘錄如下：

(1) An Offline Capable Communication Framework for Multinational Disaster Operations based on Self-aligning Wireless Gateways：

本文對初期應變人員在大規模緊急救難事故時，提出一種新通聯系統。該系統能夠使用現有的通訊網路基礎設備，並能夠將其向下結合成無線區域網路。此外，該系統也提供了網路設備離線功能。而本研究已利用通信繼電器（COFR）和無線通道（WGW）設計發展出系統雛型，救難人員於災難現場將可依據現況彼此通訊，減少干擾及斷訊發生。

(2) Prediction Metrics in Indoor Wireless Networks Considering Multimedia Applications: A Bayesian Approach：

高傳輸容量的需求，例如視訊，將影響提供給使用者的經驗質量(QoE）和服務質量（QoS）。該研究提出了一個混合預測和評估 QoE 和 QoS 通過量測指標的方法，此方法利用 Bayesian 網路和模擬來協助室內網路環境規劃，並考慮訊號接收強度指示器、峰值信噪比和結構相似性指標，進行室內測量及網路環境規劃。最後，該研究也論證該方法之可行性及未來發展。

(3) RFID Promulgation to Ameliorate the Efficiency of Health Care IT Systems through a Conceptual Model and Chronological Study：

醫療保健企業之專業管理（HCE），項目包括定義、交付最終報告和培育高專業技術人員。本文藉由 RFID 和 RTLS 功能，使開發人員能夠健全及發展 HCE 的 IT 系統。利用 RFID 技術彙整與綜合資訊，以實現預定目標。本文提出一建立使用 RFID 技術開發概念模型以發展 HIC 專業管理，並由實驗證實該模型性能將超越舊有 HIC。

1 月 30 日，當日會議其他學者、專家發表議題，摘錄如下：

(1) Smartphone-Based Assistant for Walking Rehabilitation of Patients with Parkinson’s Disease：

帕金森氏病是一種持續性的退化性病症。除了透過藥物治療達到控制病症外，仍需要保持身體相關機能、肌肉力量和協調性。而日常生活中走路這項運動是最基本的，它涉及直線和轉彎運動。然而，帕金森氏病患者對於直線和轉彎行走是有相當難度，主要因為腿部僵硬或行動遲緩。再者，患者對於跌倒的風險亦會增加許多。物理治療師會評估病患利用走路復建的姿勢，並提供口頭建議與糾正患者的動作。然而，在大多數情況下，患者幾乎在家中戶住家附近實行走路復建；因此，他們本身無法評估康復之效果，進而降低他們復建的意圖。而此論文研究利用智慧型手機內部慣性系統（如加速計和導航），以協助患者獨立評估他們的復建品質。慣性系統也被用來計數步數，並記錄患者走直線及轉彎的時間。最後，研究人員提供就由智慧型手機接收資料，並可對患者復健之品質提供及時建議。

(2) A Robust HDWT-DCT Based Watermarking Scheme Against Cropping Attacks：

本文主要說明彩色影像頻率域藏密技術其強健性與數位浮水印結合之難處，並利用 2DLDA 則可以直接利用二維影像的樣本計算出共變異矩陣，以改善執行速度和減少空間複雜度。最後本文針對實驗數據與世界各學者提出比較。

(3) Performance Improvement for Indoor Positioning Systems Using Xtion Depth Sensors and Smartphone Orientation Sensors：

現今，智慧型手機和平板電腦隨處可見，這些移動設備提供了大量的應用，如行動定位服務（LBS）。行動定位服務不僅可以提供用戶位置，同時也擴展了許多服務應用，如導航和交通資訊服務等。目前，主要的定位技術是全球定位系統（GPS），但是它並不適用於室內。而最適合室內定位技術則是利用射頻信號模式匹配的方法，而這將需要花費大量時間等缺點，來打造射頻信號模式。因此，為了減少時間花費，本研究提出了一種離線式系統，以建立感官探測器（例如，Xtion 或超高動力學）射頻信號模式。通過擷取物體影像及深度訊號，從環境周遭訊息資料可以用更少的時間和人力成本建立射頻信號模式預測模型。隨著智慧型手機內置感測器（加速度計和陀螺儀），可以估算更好的定位精度。最後，本文減少定位時間約 78%，在實驗結果，定位誤差約 2.73 米。

(4) A Safe Walking App for Pedestrians：

由於智慧型手機的便利性及普及性，人們生活中與智慧型手機已無法分開。人們時常利用零碎時間使用智慧型手機，例如坐或站著。甚至，還有人邊走路邊發簡訊。在這種情況下，使用智慧型手機將導致一些交通事故。雖然邊走路邊發簡訊是相當危險的行為，但這卻是無法有效克服避免的。因此，本文提出了利用智慧型手機的 G-傳感器和鏡頭偵測不安全的道路環境，來幫助人們避免上述交通事故。藉由實驗結果，該應用程序的檢測率達到 95%以上。

(5) A Robust Lane Detection Method Using Adaptive Road Mask：

在本文中，作者提出採用適應性路面及邊緣技術的強健性車道檢測法。此方法係利用透過施加快速及消失偵測法則產生車道路面狀況。作者提出之方法具有高效率計算能力，車道之強健性偵測，並可適應於截取道路狀況之場景。由實驗結果證明，該方法的有效及強健性能。

(6) A Kinect-Based Somatosensory Game by Integrated Unity and Motion Builder：

在本研究中，利用 Kinect，Unity 和 Motion Builder 的體感遊戲系統被提出來。Kinect 是我們與設備互動的界面，其可以體感操作遊戲系統，並可隨插即用。而 Unity 是一個簡單易懂的軟體，當作遊戲開發平台時非常有用。作者利用這個平台來建構虛擬遊戲人物，武器及場景。而 Motion Builder 則應用於記錄的遊戲人物基本技巧和怪物之攻擊。本研究的目的是建立一個體感運動與怪物戰鬥的遊戲系統。作者使用動力學來偵測人體四肢運動，以控制遊戲中虛擬角色來進行格鬥。另外，作者也提出利用 Kinect 偵測玩家手部位置並結合遊戲中虛擬武器，以利攻擊怪物的方法。Kinect 也提供了各種的武器選擇及特效，使其增加玩家攻擊怪物的多樣性與可行性。因此，玩家利用體感方式採取相對應的行動以利遊戲中角色之生存。本文利用 Kinect，Unity 和 Motion Builder 的體感遊戲系統的結果證明所採用之方式具可行性和有效性。

參、心得報告：

本次赴阿拉伯聯合大公國杜拜參加國際研討會，對本人而言算是寶貴的經驗，不僅是在學術方面，在國際觀也增加了許多知識也開了眼界，學術上藉由此次研討會，可以思考其他專家學者所專研之領域是否可以與自己的領域相結合，進而產生新的火花，創造出新的想法，另外，本研討會全程均以英文來做報告，而與會者有來自不同國家，因此要聽懂各國不同口音也是一件不簡單的事。在國際觀上，因本研討會結束後參加了主辦單位的市區導覽，期間介紹了杜拜從一片荒蕪的沙漠到發現石油，搖身一變成為現今全球最奢華的代表，市區高聳建築物林立，知名飯店用高達數十噸黃金打造，林林種種只能嘆為觀止、令人瞠目結舌，阿拉伯聯合大公國因為黑金造成如經繁華的現象，也造就許多不可思議的現象產生，例如，當日到了用餐時間，接待工作人員因與我們一同併桌吃飯，期間看見他們拿了四大盤食材，卻只吃了幾口就不吃了，請服務生收回，想一想或許是民族性不同，但也期許他們能想想早期阿拉伯聯合大公國這些酋長國們是如何的辛苦，只吃椰棗跟捕魚過活，生活是如何艱辛。最後，參加國際研討會確實對我們是一項很有幫助的學術活動，也非常感謝科技部研的經費提供，校院部各級長官的協助，使得此研討會能順利成行。

肆、參考資料：

圖片為會議舉辦地點、大會主席頒發證明、研討會場內場外、論文報告及市區導覽等

研討會地點伊斯蘭大學杜拜分校

於研討會海報前留影

大會主席頒發證明

會議主席合影

研討會會議進行(1)

研討會會議進行(2)

與會及工作人員合影

伍、建議事項：

參加國際研討會，除可以增進學術交流亦可開闊國際觀，尤其是身處台灣的我們，在國際機構組織不被接受的我們，利用參加國際學術研討會，讓各國瞭解我們的學術成就。所謂學術無國界，藉由各國的專家學者互相研討，彼此激勵出火花，對於學術而言也是一寶貴的收穫，因此，期許在經費有限的情況下，能鼓勵更多的老師能多出去走走，瞭解目前各國學者專家學術研究方向，增加學校能見度，以提升自己本職學能。最後，感謝科技部研的經費提供，校院部各級長官的協助，使得此研討會能順利成行。

陸、會議資料：

收錄論文光碟片(論文電子檔 31 篇)

A Strategy Speeds Up the Triple Modular Exponentiation

Te-Jen Chang [1], Kuang-Hsiung Tan [1], Ping-Sheng Huang [2], Ching-Yin Chen [3], and I-Hui Pan [3,*]
[1] Department of Electrical and Electronic Engineering, Chung-Cheng Institute of Technology, National Defense University, Taiwan, R.O.C.
[2] Department of Electronic Engineering, Ming Chuan University, Taiwan, R.O.C.
[3] School of Defense Science, Chung Cheng Institute of Technology, National Defense University, Taiwan, R.O.C.
[1,3] No.75, Shiyuan Rd., Daxi Township Tauyuan County 33551, Taiwan, R.O.C.
[2] No.5 De Ming Rd., Gui Shan District,Tauyuan County 333, Taiwan, R.O.C.
[1] karl591218@gmail.com, s913115@gmail.com, [2] pshuang@mail.mcu.edu.tw, [3] ccy101@gmail.com, panchefukui@gmail.com*

ABSTRACT

One of the important parts for computing M^E mod N is the modular exponentiation, where M is a plaintext, N is a modulus, and E is a large exponent. The performance of the modular exponentiation algorithm depends on the numbers of modular square and modular multiplication for the exponent. The computational complexity for the strategy which is provided to speed up the triple modular exponentiation can be reduced to $1.875j$ multiplications, where j is the bit length of the exponent.

KEYWORDS

Modular multi-exponentiation, Complement, Complex arithmetic, Hamming weight, Cryptography.

1 INTRODUCTION

The arithmetic computation of the modular multi-exponentiation plays an important role in modern cryptography. The modular multi-exponentiation is written as AXBY mod N and consists of five variables, where A and B are bases, X and Y are exponents, and a modulus N is a positive integer. The computational operation for the modular exponentiation is greatly time-consuming. They are based on the numbers of modular square and modular multiplication, which are repeatedly computed in the modular exponentiation. The modular multi-exponentiation is universally used in a lot of domains such as DSA (Digital Signature Algorithm) proposed in 1991 [1]. The kernel of most digital signature algorithms is also multi-exponentiation.

Generally, if we want the receiver to understand the secret information, which we have known, the simplest method is to tell the receiver the secret information directly. But, not only does the receiver know the secret information, but also the hacker knows the secret information, and then the secret information is no longer confidential. How do we let the receiver know the secret information which we have understood? But hackers don't know the secret information during the transmission processes.

For the above problem, Feige, Fiat, and Shamir proposed "Zero-Knowledge Proof" in 1988 [1, 2]. In "Zero-Knowledge Proof", a transmitter will not tell any secret information to a receiver. How does a transmitter let a receiver know that a transmitter himself has already understood this secret information and the hacker doesn't know the secret information? Only the receiver knows whether a transmitter understands the secret information or not. "Zero-Knowledge Proof" can be exploited in any place, which needs the authentication in order to avoid any personal information being hacked by malicious programs or hackers. We don't also let another people in government know your secret information. Now, many scholars are devoted to this subject [3, 4]. Feige-Fiat-Shamir's algorithm is described as follows. Three vectors are used

1. The private key vectors are defined as $[s_1, s_2, \ldots, s_k]$. The private key s_k and a modulus n are relatively prime.
2. The public key vectors are defined as $[v_1, v_2, \ldots, v_i]$ and the public key v_i is equal to $[(s_i)^2]^{-1}$ mod n.
3. The authentication key vectors are defined as $[c_1, c_2, \ldots, c_k]$ and they are randomly selected by receivers. c_k is 0 or 1.

The diagram for Feige-Fiat-Shamir algorithm is shown in Figure 1.

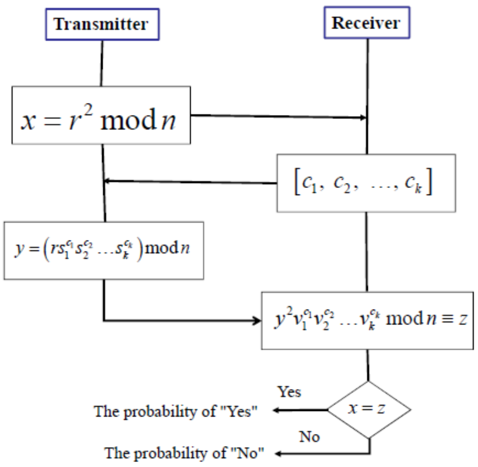
Transmitter
Receiver
$x = r^2 \bmod n$
$[c_1, c_2, \ldots, c_k]$
$y = (r s_1^{c_1} s_2^{c_2} \ldots s_k^{c_k}) \bmod n$
$y^2 v_1^{c_1} v_2^{c_2} \ldots v_k^{c_k} \bmod n \equiv z$
$x = z$
Yes
No
The probability of "Yes"
The probability of "No"

Figure 1. Feige-Fiat-Shamir algorithm diagram.

The authentication steps are shown as follows.

1. A transmitter computes $r^2 \equiv x$ mod n.
2. A transmitter transmits x to a receiver.
3. The authentication key vectors $[c_1, c_2, \ldots, c_k]$ are feedbacked to a transmitter by a receiver.
4. A transmitter computes $(r\, s_1^{c_1}\, s_2^{c_2} \ldots s_k^{c_k}) \equiv y$ mod n, where r is a random number.
5. A transmitter transmits y to a receiver.
6. A receiver computes $(y^2\, v_1^{c_1}\, v_2^{c_2} \ldots v_k^{c_k}) \equiv z$ mod n.
7. A receiver checks whether the result z is equal to x or not.

Based on "Zero-Knowledge Proof" and "Improved Common-Multiplicand Multiplication and Fast Exponentiation by Exponent Decomposition" [1, 6], triple modular exponentiation is proposed. The computational procedures will be described in the following sections.

2 MATHEMATICAL PRELIMINARY

2.1 Binary Method

The binary method is that the exponent E in decimal form is transformed into a binary form $(e_k\, e_{k-1} e_{k-2} \ldots e_1)_2$ and it is also represented $E = \sum_{i=1}^{k} e_i * 2^i$, $e_i \in \{0, 1\}$, where k is the bit length of the exponent E. There are two methods to calculate the number of modular multiplications for the modular exponentiation in binary method. One method is "scanning" from the right (the least important) bit to the left (the most important) bit. It is also called "Least Significant Bit (LSB)" method as shown in Figure 2 [5]. For example, $C = M^{57}$ is calculated by using the LSB algorithm. Now, 57 is represented $(111001)_2$ in binary form, where $k = 6$ and the procedures are shown as follows.

$$\begin{aligned} C &= M * M^8 * M^{16} * M^{32} \\ &= M^{(1+8+16+32)} \\ &= M^{57} \end{aligned}$$

```
Input: $M$, $E = (e_k e_{k-1} e_{k-2} \ldots e_1)_2$;
Output: $C = M^E$;
$C = 1$; $S = M$;
for $i = 1$ to $k$ do
    {
      if ($e_i == 1$) $C = C * S$;
      $S = S * S$;
    }
```

Figure 2 The LSB algorithm [5].

The other method is “scanning” from the left (the most important) bit to the right (the least important) bit. It is also called “Most Significant Bit (MSB)” method as shown in Figure 3. For example, $C = M^{57}$ is calculated by using the MSB algorithm. The procedures are shown as follows.

$$\begin{aligned} C &= ((((M^2 * M)^2 * M)^2)^2)^2 * M \\ &= ((((M^3)^2 * M)^2)^2)^2 * M \\ &= (((M^7)^2)^2)^2 * M \\ &= (M^{56}) * M \\ &= M^{57} \end{aligned}$$

There are only “square” and “multiplication” operations in the above procedures. So, the binary method is also called the square and multiplication algorithm.

```
Input: $M$, $E = (e_k e_{k-1} e_{k-2} \ldots e_1)_2$;
Output: $C = M^E$;
    $C = 1$;
    for $i = k$ to 1 do
    {
        $C = C * C$;
        if ($e_i == 1$) $C = C * M$;
    }
```

Figure 3 The MSB algorithm [5].

2.2 ICMM Algorithm

In 1993, professors Yen and Laih proposed a common multiplicand multiplication algorithm [6]. In 1997, professor Yen exploited a 3-part division technique to improve the common multiplicand multiplication algorithm which was proposed in 1993 to develop ICMM (Improved Common Multiplicand Multiplication) algorithm. This algorithm [2] has effectively reduced the number of multiplications.

$X*Y_1$, $X*Y_2$, and $X*Y_3$ are computed in this algorithm. The fundamental concept is that the common part of all multipliers is computed one time and then the common part of all multiplicands can be avoided computing repeatedly. So, the numbers of multiplications can be computed in binary form. Variables are defined in Equations (1) - (7) as follows.

$$Y_{common} = \text{AND}_{i=1}^{3} Y_i = Y_1 \text{ AND } Y_2 \text{ AND } Y_3, \quad (1)$$
$$Y_{1,2} = (Y_1 \text{ AND } Y_2) \text{ XOR } Y_{common}, \quad (2)$$
$$Y_{1,3} = (Y_1 \text{ AND } Y_3) \text{ XOR } Y_{common}, \quad (3)$$
$$Y_{2,3} = (Y_2 \text{ AND } Y_3) \text{ XOR } Y_{common}, \quad (4)$$
$$Y_{1,c} = Y_1 \text{ XOR } Y_{1,2} \text{ XOR } Y_{1,3} \text{ XOR } Y_{common}, \quad (5)$$
$$Y_{2,c} = Y_1 \text{ XOR } Y_{1,2} \text{ XOR } Y_{2,3} \text{ XOR } Y_{common}, \quad (6)$$
$$Y_{3,c} = Y_1 \text{ XOR } Y_{1,3} \text{ XOR } Y_{2,3} \text{ XOR } Y_{common}. \quad (7)$$

In Equations (1) - (7), “AND” and “XOR” mean “AND gate” and “XOR gate” in logical operation. All common bits of Y_i are recorded in the Y_{common} through “AND gate” operation as shown in Equation (1), where Y_i means Y_1, Y_2, and Y_3. Y_1 means the first multiplier, Y_2 means the second multiplier, and Y_3 means the third multiplier. Y_{common} means the same bits for Y_1, Y_2, and Y_3. X means the multiplicand. Some of the same bits are recorded in the $Y_{1,2}$, $Y_{1,3}$, and $Y_{2,3}$ through “AND gate” and “XOR gate” operations as shown in Equations (2) - (4). These records are different from Y_{common}. But some of these records are still the same as Y_{common}. At last, the total different bits are recorded in the $Y_{1,c}$, $Y_{2,c}$, and $Y_{3,c}$ through “XOR gate” operation as shown in Equations (5) - (7). Three multipliers are defined, respectively as follows.

$$Y_1 = Y_{1,c} + Y_{1,2} + Y_{1,3} + Y_{common},$$
$$Y_2 = Y_{2,c} + Y_{1,2} + Y_{2,3} + Y_{common},$$
$$Y_3 = Y_{3,c} + Y_{1,3} + Y_{2,3} + Y_{common}.$$

According to the above definitions, the multiplications for one common multiplicand “X” can be expanded as follows.

$$X * Y_1 =$$
$$X * Y_{1,c} + X * Y_{1,2} + X * Y_{1,3} + X * Y_{common},$$
$$X * Y_2 =$$
$$X * Y_{2,c} + X * Y_{1,2} + X * Y_{2,3} + X * Y_{common},$$
$$X * Y_3 =$$
$$X * Y_{3,c} + X * Y_{1,3} + X * Y_{2,3} + X * Y_{common}.$$

Obviously, when $X * Y_1$, $X * Y_2$, and $X * Y_3$ are computed, Y_{common} is only computed one time for the common part and $Y_{1,2}$, $Y_{1,3}$, and $Y_{2,3}$ are also computed one time. The probabilities for the computational complexity of each state are shown in Table 1. The probability of non-zero bits which are appeared in Y_i is $\frac{m}{2}$, where m is the bit length of Y_i. The probabilities for the computational complexity of each state are described as follows.

State 1: The probability of non-zero bits which are appeared in Y_{common} is $\frac{1}{2}*\frac{1}{2}*\frac{1}{2}=\frac{1}{8}$.

State 2: When $Y_{1,2} = 1$, the bit vector $[Y_1, Y_2, Y_3]$ is equal to [1, 1, 0]. Because the probability which a single bit is 1 or 0 is $\frac{1}{2}$, the probability of State 2 is $\frac{1}{2}*\frac{1}{2}*\frac{1}{2}=\frac{1}{8}$.

State 3: When $Y_{1,3} = 1$, the bit vector $[Y_1, Y_2, Y_3]$ is equal to [1, 0, 1]. Because the probability which a single bit is 1 or 0 is $\frac{1}{2}$, the probability of State 3 is $\frac{1}{2}*\frac{1}{2}*\frac{1}{2}=\frac{1}{8}$.

State 4: When $Y_{2,3} = 1$, the bit vector $[Y_1, Y_2, Y_3]$ is equal to [0, 1, 1]. Because the probability which a single bit is 1 or 0 is $\frac{1}{2}$, the probability of State 4 is $\frac{1}{2}*\frac{1}{2}*\frac{1}{2}=\frac{1}{8}$.

State 5: When $Y_{1,c} = 1$, the bit vector $[Y_1, Y_2, Y_3]$ is equal to [1, 0, 0]. Because the probability which a single bit is 1 or 0 is $\frac{1}{2}$, the probability of State 5 is $\frac{1}{2}*\frac{1}{2}*\frac{1}{2}=\frac{1}{8}$.

State 6: When $Y_{2,c} = 1$, the bit vector $[Y_1, Y_2, Y_3]$ is equal to [0, 1, 0]. Because the probability which a single bit is 1 or 0 is $\frac{1}{2}$, the probability of State 6 is $\frac{1}{2}*\frac{1}{2}*\frac{1}{2}=\frac{1}{8}$.

State 7: When $Y_{3,c} = 1$, the bit vector $[Y_1, Y_2, Y_3]$ is equal to [0, 0, 1]. Because the probability which a single bit is 1 or 0 is $\frac{1}{2}$, the probability of State 7 is $\frac{1}{2}*\frac{1}{2}*\frac{1}{2}=\frac{1}{8}$.

The probability of the ICMM (Improved Common Multiplicand Multiplication) algorithm is $\frac{7m}{8}$ and the probability of a general multiplication algorithm is $t * \frac{m}{2}$, where m means the bit length of the exponent and t means division parts for the bit length "m" of the exponent. So, the efficiency for the ICMM algorithm is $\frac{t*\frac{m}{2}}{\frac{7m}{8}}$. When $t = 3$ in the ICMM algorithm, the optimal efficiency is $\frac{12}{7}$ = 1.71. It means the number of multiplications can be reduced to 1.71 for computing $X * Y_1$, $X * Y_2$, and $X * Y_3$ by using the ICMM algorithm.

Table 1. The probabilities of each state by which uses ICMM algorithm.

States		Probabilities
1	$Y_{common} = 1$	$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$
2	$Y_{1,2} = 1$	$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$
3	$Y_{1,3} = 1$	$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$
4	$Y_{2,3} = 1$	$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$
5	$Y_{1,c} = 1$	$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$
6	$Y_{2,c} = 1$	$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$
7	$Y_{3,c} = 1$	$\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$

In a general application, we can compare CMM and ICMM algorithms for many multiplication architectures which have one common multiplicand. We list the comparison figure for the original multiplication algorithm, CMM algorithm, and ICMM algorithm as shown in Figure 4.

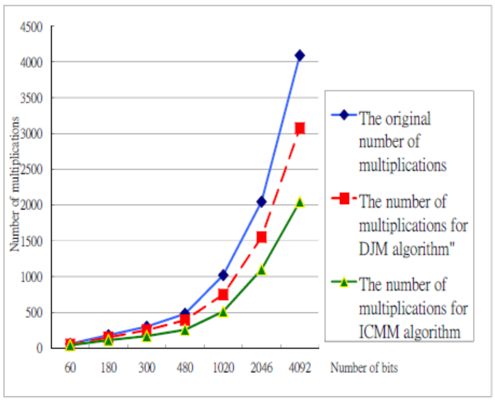

Figure 4. A comparison figure for different numbers of multiplications algorithms.

3 THE PROPOSED ALGORITHM

According to the concept of ICMM (Improved Common multiplicand Multiplication) algorithm, we think the similar architecture of multiplications can be adopted in the triple modular exponentiation algorithm to reduce the computational complexity of the modular exponentiation effectively. We set T, M_1, M_2, M_3, E_1, E_2, E_3, and N positive integers. The variables of the proposed algorithm are defined in Equation (8).

$$T = {}_{i=1}^{3} M_i^{E_i} \bmod N. \quad (8)$$

For the triple modular exponentiation, that is the proposed algorithm, it can be regarded as three multiplication groups of no common multiplicand. We firstly record the common bits of E_1, E_2, and E_3, and then parts of the same bits are recorded. The similar architecture of an improved common multiplicand multiplication can be obtained. At the same situation, the common part of all multipliers is only computed one time, and then the thing of computing the common part repeatedly can be avoided. The number of modular multiplication can be simplified. The definitions of variables in the proposed algorithm are shown in Equation (9) - (15).

$$E_{common} = \text{AND}_{i=1}^{3} E_i = E_1 \text{ AND } E_2 \text{ AND } E_3, \quad (9)$$
$$E_{1,2} = (E_1 \text{ AND } E_2) \text{ XOR } E_{common}, \quad (10)$$
$$E_{1,3} = (E_1 \text{ AND } E_3) \text{ XOR } E_{common}, \quad (11)$$
$$E_{2,3} = (E_2 \text{ AND } E_3) \text{ XOR } E_{common}, \quad (12)$$
$$E_{1,c} = E_1 \text{ XOR } E_{1,2} \text{ XOR } E_{1,3} \text{ XOR } E_{common}, \quad (13)$$
$$E_{2,c} = E_2 \text{ XOR } E_{1,2} \text{ XOR } E_{2,3} \text{ XOR } E_{common}, \quad (14)$$
$$E_{3,c} = E_3 \text{ XOR } E_{1,3} \text{ XOR } E_{2,3} \text{ XOR } E_{common}. \quad (15)$$

"AND" and "XOR" mentioned in Equations (9) - (15) are "AND gate" and "XOR gate" in the logical operation. The common bits of E_i are recorded in E_{common} through "AND gate" operation as shown in Equation (9). The different bits are recorded in $E_{1,2}$, $E_{1,3}$, $E_{2,3}$, $E_{1,c}$, $E_{2,c}$, and $E_{3,c}$ through "AND gate" and "XOR gate" operations as shown in Equations (10) - (15). The common bits of E_1 and E_2 are recorded in Equation (10). The common bits of

E_1 and E_3 are recorded in Equation (11). The common bits of E_2 and E_3 are recorded in Equation (12). The different bits of E_1, $E_{1,2}$, $E_{1,3}$, and E_{common} are recorded in Equation (13). The different bits of E_2, $E_{1,2}$, $E_{2,3}$, and E_{common} are recorded in Equation (14). The different bits of E_3, $E_{1,3}$, $E_{2,3}$, and E_{common} are recorded in Equation (15).Attention, please! After logical operations, the seven variables (E_{common}, $E_{1,2}$, $E_{1,3}$, $E_{2,3}$, $E_{1,c}$, $E_{2,c}$, and $E_{3,c}$) are "exclusive" shown in Equations (9) - (15). That is, the result which 2 more variables are "1" at the same time is not existed. "1" is existed only for one variable. So, three exponents E_1, E_2, and E_3 are defined as follows.

$$E_1 = E_{1,c} + E_{1,2} + E_{1,3} + E_{common},$$
$$E_2 = E_{2,c} + E_{1,2} + E_{2,3} + E_{common},$$
$$E_3 = E_{3,c} + E_{1,3} + E_{2,3} + E_{common}.$$

According to the above definitions and an associative law of the exponents, the computations of the modular exponentiation are shown as follows.

$$M_1^{E_1} = M_1^{(E_{1,c}+E_{1,2}+E_{1,3}+E_{common})} = M_1^{E_{1,c}} * M_1^{E_{1,2}} * M_1^{E_{1,3}} * M_1^{E_{common}}, \tag{16}$$

$$M_2^{E_2} = M_2^{(E_{2,c}+E_{1,2}+E_{2,3}+E_{common})} = M_2^{E_{2,c}} * M_2^{E_{1,2}} * M_2^{E_{2,3}} * M_2^{E_{common}}, \tag{17}$$

$$M_3^{E_3} = M_3^{(E_{3,c}+E_{1,3}+E_{2,3}+E_{common})} = M_3^{E_{3,c}} * M_3^{E_{1,3}} * M_3^{E_{2,3}} * M_3^{E_{common}}, \tag{18}$$

From Equations (16) - (18), we know if the original modular exponentiation computations are used, three modular exponents ($M_1^{E_1}$, $M_2^{E_2}$, and $M_3^{E_3}$) need four modular exponentiation computations. But the concepts of ICMM (Improved Common Multiplicand Multiplication) and CMM (Common Multiplicand Multiplication) algorithms are used; bases of the exponents can be adjusted and combined again. The seven variables can be computed again as follows.

$$M_1^{E_{1,c}},$$
$$M_2^{E_{2,c}},$$
$$M_3^{E_{3,c}},$$
$$(M_1M_2)^{E_{1,2}} = M_1^{E_{1,2}} * M_2^{E_{1,2}},$$
$$(M_1M_3)^{E_{1,3}} = M_1^{E_{1,3}} * M_2^{E_{1,3}},$$
$$(M_2M_3)^{E_{2,3}} = M_2^{E_{2,3}} * M_3^{E_{2,3}},$$
$$(M_1M_2M_3)^{E_{common}} = M_1^{E_{common}} * M_2^{E_{common}} * M_3^{E_{common}} \cdot M_1^{E_1} * M_2^{E_2} * M_3^{E_3} = M_1^{E_{1,c}} * M_2^{E_{2,c}} * (M_1M_2)^{E_{1,2}} * (M_1M_3)^{E_{1,3}} * (M_2M_3)^{E_{2,3}} * (M_1M_2M_3)^{E_{common}} \tag{19}$$

From Equation (19), the proposed algorithm (triple modular exponentiation) is described as shown in Figure 5.

```
Input: $m$, $n$, $x$, $y$, and $p$: positive integers, $p$-prime;
Output: $T \equiv (M_1^{E_1} * M_2^{E_2} * M_3^{E_3}) \bmod n$

    Step 1: Precompute $(M_1M_2) \equiv a_1 \bmod P$;
            $(M_1M_3) \equiv a_2 \bmod P$; $(M_2M_3) \equiv a_3 \bmod P$; $(M_1M_2M_3) \equiv a_4 \bmod P$;
    Step 2: $s = 1$; $j = \max(M_1, M_2, M_3)$
    Step 3: if ($E_{common} == 1$), then $(s * a_4) \equiv s \bmod N$;
        if ($E_{2,3} == 1$), then $(s * a_3) \equiv s \bmod N$;
        if ($E_{1,3} == 1$), then $(s * a_2) \equiv s \bmod N$;
        if ($E_{1,2} == 1$), then $(s * a_1) \equiv s \bmod N$;
        if ($E_{1,c} == 1$), then $(s * M_1) \equiv s \bmod N$;
        if ($E_{2,c} == 1$), then $(s * M_2) \equiv s \bmod N$;
        if ($E_{3,c} == 1$), then $(s * M_3) \equiv s \bmod N$;
    Step 4: if $k >= 1$, then $(s * s) \equiv s \bmod p$;
    Step 5: $j = j - 1$;
    Step 6: if $j >= 0$, then goto Step 5 else
            output $s$.
```

Figure 5. The proposed algorithm.

4 COMPUTATIONAL COMPLEXITY ANALYSES

The probability of each state is shown in Table 2. In the proposed algorithm, j is one of the largest bit length for E_1, E_2, and E_3 i.e. $j = \max(E_1, E_2, E_3)$. If the bit lengths for E_1, E_2, and E_3 are not the same, "0" can be padded into the two short exponents until the bit lengths of the two short exponents are the same as the bit length j. Because the probability of non-zero bits in E_i is $\frac{j}{2}$, the probability of each state is described as follows.

Table 2. The probability of each state in the proposed algorithm.

States	Probabilities
$E_{common} = 1$	$\frac{1}{8}$
$E_{1,2} = 1$	$\frac{1}{8}$
$E_{1,3} = 1$	$\frac{1}{8}$
$E_{2,3} = 1$	$\frac{1}{8}$
$E_{1,c} = 1$	$\frac{1}{8}$
$E_{2,c} = 1$	$\frac{1}{8}$
$E_{3,c} = 1$	$\frac{1}{8}$

State 1:The probability of non-zero bits in E_{common} is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$.

State 2: When $E_{1,2} = 1$, the bit vector $[E_1, E_2, E_3]$ is equal to $[1, 1, 0]$. Because the probability of a single bit (1 or 0) is $\frac{1}{2}$, the probability of State 2 is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$.

State 3: When $E_{1,3} = 1$, the bit vector $[E_1, E_2, E_3]$ is equal to $[1, 0, 1]$. Because the probability of a single bit (1 or 0) is $\frac{1}{2}$, the probability of State 3 is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$.

State 4: When $E_{2,3} = 1$, the bit vector $[E_1, E_2, E_3]$ is equal to $[0, 1, 1]$. Because the probability of a single bit (1 or 0) is $\frac{1}{2}$, the probability of State 4 is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$.

State 5: When $E_{1,c} = 1$, the bit vector $[E_1, E_2, E_3]$ is equal to $[1, 0, 0]$. Because the probability of a single bit (1 or 0) is $\frac{1}{2}$, the probability of State 5 is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$.

State 6: When $E_{2,c} = 1$, the bit vector $[E_1, E_2, E_3]$ is equal to $[0, 1, 0]$. Because the probability of a single bit (1 or 0) is $\frac{1}{2}$, the probability of State 6 is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$.

State 7: When $E_{3,c} = 1$, the bit vector $[E_1, E_2, E_3]$ is equal to $[0, 0, 1]$. Because the probability of a single bit (1 or 0) is $\frac{1}{2}$, the probability of State 7 is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$.

The computational complexity of the triple modular exponentiation algorithm is related to the numbers of "square" and "multiplication" for the exponents. In the proposed algorithm, $E_{common} = 0$ is regarded as one multiplication and each of the others ($M_1^{E_{1,c}}$, $M_2^{E_{2,c}}$, $M_3^{E_{3,c}}$, $(M_1M_2)^{E_{1,2}}$, $(M_1M_3)^{E_{1,3}}$, $(M_2M_3)^{E_{2,3}}$, and $(M_1M_2M_3)^{E_{common}}$) is regarded as two multiplications. According to the probabilistic analyses in Table 4 and for "$(M_1^{E_1} * M_2^{E_2} * M_3^{E_3})$ mod N" in the triple modular exponentiation type, when the probability is $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}$ for "$E_1 = E_2 = E_3 = 0$" at the same time. The number of multiplications for the computational complexity is 1.875j, where j is the largest bit length of exponents E_1, E_2, and E_3 shown in the following equation:

$$(2 * \frac{1}{8} * 7 + 1 * \frac{1}{8})j = 1.875j$$

For analytic result, comparing the CMM algorithm and the triple modular exponentiation algorithm, we list the comparison diagram which describes the numbers of the multiplications for the original multi-exponent algorithm, the CMM algorithm, and the proposed algorithm as shown in Figure 6.

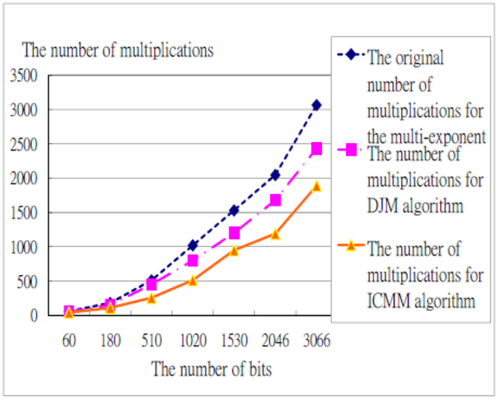

Figure 6. The comparison diagram for the numbers of the multiplications.

5 CONCLUSIONS

The study of the triple modular exponentiation is proposed in this paper. Nobody has not yet not discussed this topic until now. More mathematicians only discuss the double modular exponentiation. According to the improved common multiplicand algorithm, we think the common bits of associated exponents can be recorded separately for the triple modular exponentiation algorithm, and then a similar architecture of the improved common multiplicand multiplication can be obtained. It can effectively reduce the computational complexity of the triple modular exponentiation to simplify the number of the modular multiplication. Finally, the average computational complexity of the proposed algorithm is 1.875j, where j is the bit length of the exponent. Comparing with the double modular exponent, the triple modular exponentiation is more efficient.

REFERENCES

[1] U. Feige, A. Fiat and A. Shamir, "Zero-Knowledge Proofs of Identity," *Journal of Cryptology*, vol. 1, no. 2, 1988, pp.77-94.

[2] B. A. Forouzan, Introduction to Cryptography and Network Security, McGraw-Hill, New York, , 2008, pp. 429-430.

[3] H. Liu and H. Ning, "Zero-Knowledge Authentication Protocol Based on Alternative Mode in RFID Systems," *IEEE Sensors Journal*, vol. 11, no. 12, Dec. 2011, pp. 3235-3245.

[4] J. M. Kizza, "Feige-Fiat-Shamir ZKP Scheme Revisited," *International Journal of Computing and ICT Research*, vol. 4, no. 1, , June 2010, pp. 9-19.

[5] S.-M. Yen and C.-S. Laih, "Common-Multiplicand Multiplication and its Applications to Public Key Cryptography," *IEE Electronics Letters*, vol. 29, no. 17, Aug. 1993, pp. 1583-1584.

[6] Yen, S. M. "Improved Common-Multiplicand Multiplication and Fast Exponentiation by Exponent Decomposition," *IEICE Transactions on Fundamentals of electronics, communications and computer sciences*, vol. E80-A, no. 6, June 1997, pp. 1160-1163.