國立政治大學資訊科學研究所
Department of Computer Science
National Chengchi University

碩士論文
Master Thesis

一個可降低 Gentry 全同態加密演算法公鑰個數之提案

An Improvement of Gentry's "Fully Homomorphic Encryption Scheme" by Reducing the Number of Public Keys

指導教授：左瑞麟 教授
研 究 生：陳漢光

中華民國一〇一年七月
July 2012

一個可降低Gentry全同態加密演算法公鑰個數之提案

An Improvement of Gentry's "Fully Homomorphic Encryption Scheme" by Reducing the Number of Public Keys

研 究 生：陳漢光　　Student：Han-Kuang, Chen

指導教授：左瑞麟　Advisor：Ray-Lin, Tso

國立政治大學

資訊科學系

碩士論文

A Thesis

submitted to Department of Computer Science

National Chengchi University

in partial fulfillment of the Requirements

for the degree of

Master

in

Computer Science

中華民國一〇一年七月

July 2012

摘要

"全同態加密法"(Fully Homomorphic Encryption (FHE))一詞的介紹以及架構源於西元 2009 年由 Gentry 所提出。它讓加密後的密文執行特定的運算再將其解密即可得出該對應的明文運算結果，除此之外，全同態與同態最大的不同是它允許兩種或是多種以上的運算元進行資料運算，期間必須可以處理大量的資料並且保護其資料隱私性使其無洩漏之虞。也因為上述特點使得它可被廣泛使用在許多資料庫或是資料儲存上的應用，像是 ASP、雲端運算或是雙方相等性驗證上，然而在 Gentry 的全同態加密中，它需要大量的空間來儲存所需要的公鑰，因此在實作上仍有一定的難度。為了解決上述問題，本文提供了一種新的改良方案使其更有效率來達到全同態加密的實作性，除此之外，我們也會在文章中提出安全性分析來證明本改良方案並不會對安全性造成影響，並且提出系統效能測試，說明本方案除了可減少公鑰儲存空間之外，在時間上，更可降低公鑰生成以及系統加密的時間，讓其全同態運算更具效率。

Abstract

C. Gentry in 2009 proposed the first practical scheme which can compute arbitrary functions of encrypted data. This scheme is named “Fully Homomorphic Encryption (FHE)”. FHE allows a worker without the secret decryption key to compute any result of the data on one hand and still keep the data privacy on the other hand. It can be widely used in data storage application or database application, such as ASP, cloud computing and two-party equality testing. However, one drawback of Gentry’s fully homomorphic encryption scheme is that the size of public keys used in this system is extremely large. This means that a lot of space is required in order to store those public keys. This problem causes Gentry’s FHE hard to be implemented. In this thesis, we address the problem above, and give an improvement encryption scheme. Our improvement scheme needs less space to store the public keys which also makes the new scheme more efficient than Gentry’s original scheme. We also give a rigorous security proof to show that our improvement scheme is as secure as Gentry’s original scheme. A system performance test is also provided which shows that our scheme can not only reduce the numbers of public keys, but also reduce the time for public key generation and for encryption. Therefore, our improvement scheme can make fully homomorphic encryption more practical.

致謝

時間匆匆，隨著碩士論文的完成，也為我的學生生涯畫上了一個句點。回首過往，感謝許多老師、家人以及朋友的幫助以及指引，讓我在就讀碩士期間學習了很多也獲得到更多。首先要感謝我的指導教授左瑞麟老師，謝謝老師的指導以及督促讓我在資訊安全領域上獲益良多，更重要的是老師除了授業之外，更帶給我們謙和待人、誠正做人等待人處事的道理。在我的心中，左老師不僅僅是傳達專業知識的教授，更是一位關心學生、了解學生的兄長。感謝口試委員楊中皇教授、王旭正教授以及黃仁俊教授，三位給我的鼓勵以及意見，讓我獲益良多。政大師長們的教導跟指點，更是我碩士兩年自我培育過程的明燈。

感謝同窗同學承峰、欣瑤、凱彬以及圖學實驗室同學這兩年的互相砥礪，也謝謝兩位學長致諺、士峰的照顧，讓我們完全都沒有剛進實驗室該有的陌生，謝謝理學院的融亭、亭君、映磊，在我投稿時間給我的鼓勵以及幫助。也謝謝兩位學弟士豪跟思瑋，在我碩二的期間帶給我許多的歡樂。而我特別感謝承峰及浩宇，感謝你們這一路上以來的包容跟幫助，不管是課業上的討論或是分享彼此的人生，這一路上都謝謝你們的陪伴跟幫助，讓我在人生的道路上並不孤單。在這兩年也很感謝心恆、柏蓁，陳鐸、志翰、峻年、克嘉、閔閑、慈葳、冠甫、家瑄、宏儒、凱群、明慶、宗憲等朋友假日的陪伴，讓我在周末時可以放鬆自己，迎接下一關卡。謝謝上述的各位朋友，在我求學的期間給我的鼓勵、關心、提點，真的謝謝你們。

最後我要感激我的家人，特別是我的父母，在求學這兩年間讓我毫無後顧之憂的往上衝刺，父母給我的加油打氣畫面歷歷在目，我也銘記在心，感謝父母所為孩子做出的犧牲與苦心栽培。最後則是要感激我的外公，外公在我著寫論文期間離開了我們，感謝你與外婆幼時的撫養及教養之恩，辛辛苦苦地把一個孩子養到懂事，這份恩情我永遠記在心裡，最後感謝的是還是這一路上曾幫助過我的師長、家人以及朋友，希望我的努力能讓你們感到欣慰與驕傲。

目錄

圖目錄

1. 緒論:

1.1 雲端運算

隨著科技的日新月異，造就了現今電子數位化的時代，使用者只須要有一台可連接網路的微電腦就可以漫遊在網路之中取得他們所需要的資訊或是傳遞彼此之間的訊息。但隨著數位化的普及造成單一台電腦在運算上或是儲存空間上漸漸開始不堪負荷，也因此進而誕生雲端運算的概念。雲端運算是一種資料處理及儲存的模式，它透過網路連結其他眾多電腦，使其可以進行平行運算，簡單來說利用網路來達到多台電腦運算上的溝通，或是透過網路來使用遠端伺服器提供的服務，皆可被歸類於雲端運算之中。也因為它的便利性以及成本上的考量，雲端運算已經成為近年來熱門的話題。

在現今，不管所在何時及何處，人們都可以透過雲端運算去接收或發送他們所需要的線上資訊。使得資訊的傳遞變得越來越為方便。但在便利性的背後，也逐漸突顯了安全性以及隱私性的重要。雖然我們都知道將資訊加密是最為安全的做法，但當我們需要對這些加密後的資訊做出運算時，必須先將這些加密後的密文進行解密進而進行運算，如此一來不僅造成安全上的風險更使得運算效率降低。假設我們想對兩個數值進行相加，在上述的情況下我們必須先將這兩個加密後的密文解密還原其原本數值進行加法運算，再將其結果加密並且回傳到雲端上，這樣子的過程除了在效率上大幅度降低之外更造成了硬體上的負擔，因此是個不明智的解決辦法。但在另一方面，若資料沒有進行加密可能會造成較大風險的損失，像是醫療隱私這種特定的資料是必須經由加密來確保個資上的隱私以及安全性。也因此產生了一個問題，如何在沒有私鑰的情況下，可以透過伺服器端進行運算而依然保持正確性呢?

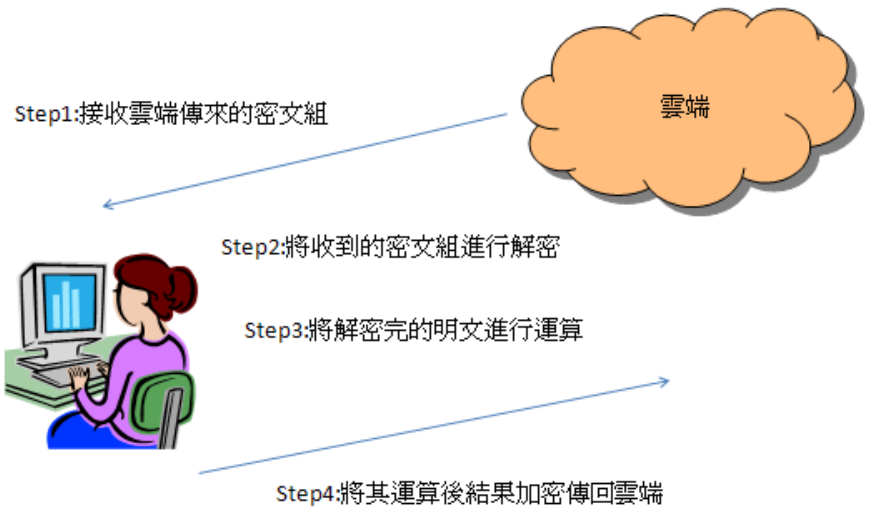

圖 1　利用傳統加密方法保障機密性且需進行雲端資料運算流程圖

1.2解決方法

有鑑於上述問題，為了解決安全性上的疑慮，許多學者紛紛研究如何設計同態加密演算法(Homomorphic Encryption)來確保資料的安全性以及正確性[3][16][17][24]。其中，以 Gentry 的全同態演算法[11][12][13]最為著名，主要原因為其演算法除了沒有限制運算次數之外更可讓兩種以上的運算子進行全同態運算。簡單的說，若想讓儲存在雲端上的資料做整數運算(如:加法、乘法)而又不想執行如圖一這般的繁雜的傳送以及動作，使用同態加密對使用者端而言，是最明智與方便的決定。舉例說明:假設明文為$m_1, m_2, \ldots, m_t$，經由加密過後得出密文為$c_1, c_2, \ldots, c_t$,並且送至雲端，使用同態加密進行運算對使用者端而言，只需要一次的解密動作即可得到結果。所需要的步驟只需選擇函數 f 對雲端上加密的密文進行特定運算後將結果傳送給使用者端進行解密，其結果會等同於其對應明文的運算結果。基於以上特性，在對密文進行運算處理時並不會造成任何資料隱私上的任何問題。舉例來說，c_1, c_2分別代表 $E(m_1)$ 和 $E(m_2)$的加密密文，所選擇的函數 f 為加法運算，因此 $E(m_1) \oplus E(m_2) = E(m_1 + m_2)$，最後我們只需將 $E(m_1 + m_2)$ 解密即可得到$m_1 + m_2$ 的計算結果，並且不會洩漏m_1, m_2 或是其他資訊。

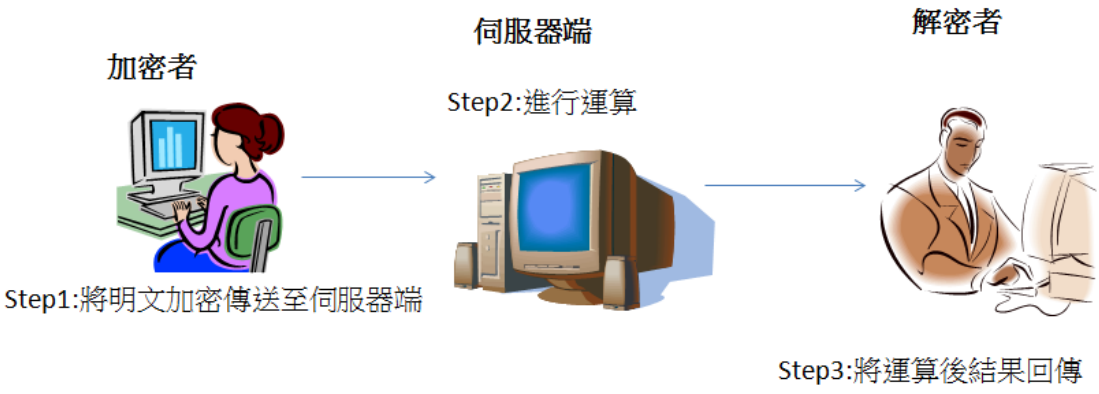

圖 2　利用同態加密進行資料運算流程圖

1.3全同態加密應用

同態以及全同態加密演算法可以廣泛的被應用在許多需要隱私性保護的資料應用上[8][15][25]，像是搜尋引擎、電子郵件、MSN…等。企業或是政府亦可利用全同態加密來達到協同作業上的資料隱私性。舉例來說，一個國家若想要統計該國人民得到 AIDS 的統計數據或是加總資料，但是卻不想這類資料被其他機構或是人員知曉其細部內容，使用全同態加密不論是資料或是運算都在伺服器端儲存或進行，並且伺服器端並沒有解密金鑰，因此並無法破解密文以及最後的運算結果。因此若要對隱私性的資訊進行運算，選擇全同態加密對於保密資料以及計算資料上而言是最好的選擇。更甚是企業可以使用全同態加密來計算員工工時或是員工薪資，這樣一來則少去許多信任上以及相關疑慮，讓公司在人事或是其他相關作業上的運轉增加效率、少去不必要的麻煩。若在雲端上使用全同態加密，由於資料可儲存在雲端上，因此可以省去硬體設備上的成本，更重要的是對於使用者端而言，我們只需將伺服器端傳來的運算結果進行解密即可(如圖二)，因此若在雲端上使用全同態加密可以保證其便利性以及低成本的兩大優點。

除了以上雲端上的應用之外，全同態加密更可以使用在雙方相等性協議的實作上[8][25]。若兩方 A、B 想要比較各自擁有其秘密資訊m_1, m_2是否相等，期間並不得洩漏任何有用的資訊給任何一方，使用全同態加密去實作此項協定可以確保資料的正確性以及安全性。而使用全同態加密的雙方相等性驗證更是可以被應用在許多資料庫服務或是搜尋引擎服務上，例如線上拍賣競標、商業談判、電子投票、線上登入…等。皆可利用上述技術去解決安全以及隱私性上的問題。

簡而言之，Fully Homomorphic Encryption(FHE)是一套強而有力的工具尤其在提供雲端計算或是其他應用所需要的便利性和資料隱私性。

1.4研究動機及貢獻

雖然 Gentry 的全同態加密可以被應用在上述的各種服務中而且保證其正確性以及隱私性。但是在實作上仍有相當的難度。其中最重要的一點是，公鑰生成需要太多的硬體儲存空間。一般來說，公鑰的大小至少佔據 800MB[6]，所需要的容量實在太大，因此導致實作上有相當程度的瓶頸。在假設一個狀況，若是要對 n 方進行多方通訊，那公鑰的總大小至少需要(800*n)MB，如此一來對於系統造成一個沉重的負擔。因此，減少公鑰的數量是本文首要研究課題。

本論文主要是根據 Gentry 的全同態加密進行改良。利用減少公鑰個數讓本研究的全同態加密在實作上是較為可行的，若以上述 800MB 為例利用本研究改良的全同態加密方案可以省去近 1/3(266MB)的空間，在 n 人通訊上更可省去(266*n MB)的空間資源，讓其較 Gentry 的原始方案易於實作並且不影響安全性。

並且由於公鑰使用數量的減少，使得公開金鑰生成以及明文加密的時間也隨之降低。亦即若使用我們改進後的全同態加密提案方式，可以有效率的降低金鑰生成的時間讓其實用性增加。

1.5論文架構

本論文共分為六個章節，首先第一章為緒論，介紹全同態加密演算法的設計緣由以及其相關應用，接著在第二章我們會介紹一些與同態相關的定義以及其相關知識，像是加法同態與乘法同態的基本介紹、公開金鑰的同態加密演算法...等。第三章則是介紹 Gentry 所提出的全同態加密演算法以及他的實作瓶頸為何、第四章則是介紹我們改良過後的 Gentry 的全同態加密方案以及本論文的貢獻，第五章則是提出其安全性分析、第六章則是提出系統效能分析，並且在第七章提出其全同態加密應用並且在最後提出我們的結論未來展望。

2. 預備知識

在第二章我們會介紹一些與本研究相關的密碼學基礎知識。以及說明同態定義及其相關演算法，並且在最後為一些安全性定義做出介紹。

2.1近代密碼學

密碼學最重要的課題就是安全且秘密的傳遞兩方或是多方彼此擁有的私密資訊。首先一個完整的密碼系統(Cryptosystem)必須由明文(plaintext)、金鑰(key)、加密演算法(encryption algorithm)、解密演算法(decryption algorithm)以及密文(ciphertext)等五項元素所組成。明文一般為加密前的原始資料，通常為加密過程中的輸入之一、解密結果的輸出。密文則代表加密過後的資料，將其轉換後成為難以理解的訊息，通常代表加密過程的輸出，解密過程的輸入。而加密演算法則是利用金鑰對明文進行加密的演算法。解密演算法則是剛好相反，則是利用金鑰對密文進行解密的演算法。架構圖如下:

圖 3　加解密基本流程圖

一般而言密碼系統分為兩大類，分別稱為對稱式金鑰密碼系統以及非對稱式金鑰密碼系統。

2.1.1 對稱式金鑰密碼系統 (Symmetric Key Cryptosystem)

在對稱式金鑰密碼系統中，會有兩方角色分別為發送者以及接受者。在確定彼此的

雙方的通訊之前，必須決定一把雙方都擁有的共同金鑰。再確定金鑰選定後，發送者會使用這把雙方都擁有的金鑰對明文進行加密(Encryption)算出密文，($E_k(m) = C$，其中 E 為加密演算法、k 為雙方都擁有的密鑰，C 為密文)並且將其結果傳送至接收者。接收者收到密文訊息後，利用相同的金鑰進行解密(Decryption)得出明文 ($D_k(c) = m$，其中 D 為解密演算法)，簡單的說，對於對稱式金鑰密碼系統而言所使用的加解密金鑰皆為同一把，像 AES[7]、DES[4]都為著名的對稱式加密系統。

對稱式金鑰密碼系統最大的優點在於計算方面較為簡單及快速，但是相對來說其缺點為金鑰的管理以及傳送上的困難，如何成功的設計出一個安全性機制可將金鑰安全的分送到所要進行通訊的另一方或是保證這把雙方都擁有的密鑰不會有洩漏的疑慮，以上兩點都是相當困難去實行及驗證。然而對稱式密碼最讓人詬病的則是金鑰數量上的管理，假設現在有 N 方要進行兩兩之間的通訊，對每一個使用者而言則需要C_2^N把金鑰，這樣的數量對於管理或是儲存都是會對使用者造成莫大的負擔。

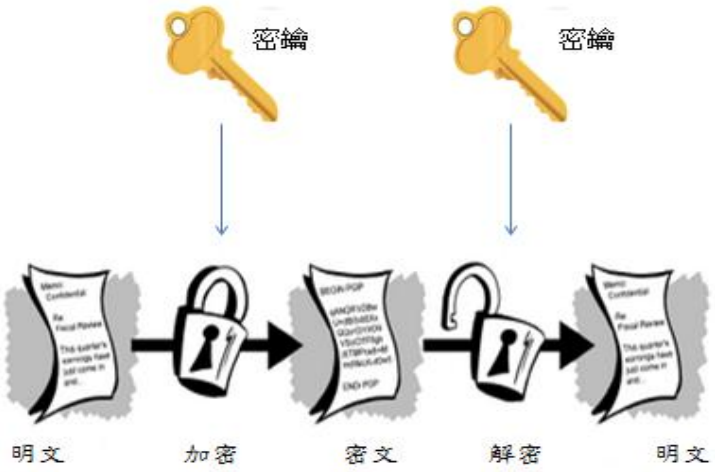

圖 4　對稱式金鑰密碼系統

2.1.2　非對稱式金鑰密碼系統 (Asymmetric Key Cryptosystem)

此密碼系統又稱做公開金鑰密碼系統，顧名思義與對稱式金鑰密碼系統最大的差別為加密和解密要使用不同的金鑰。

對於非對稱式金鑰密碼系統，每個使用者擁有一對公私鑰對(public key and secret

key)，兩把金鑰為數學相關。在加密過程中，訊息由公開金鑰進行加密後，其密文必須以私密金鑰才可解密，其中公開金鑰與加密演算法皆可公開，其中只需對私密金鑰進行隱密的儲存，像 RSA[22]、ElGamal[9]都為著名的非對稱式加密系統。

與對稱式金鑰密碼系統相較，雖然使用非對稱式金鑰加密在加密過程中的計算效率較低，但是最大的優點為公開金鑰是允許被公開的，因此沒有像對稱式系統上安全傳送密鑰的問題。若是進行 N 方兩兩通訊，也並不用C_2^N這麼多把金鑰即可解決彼此通訊的問題。

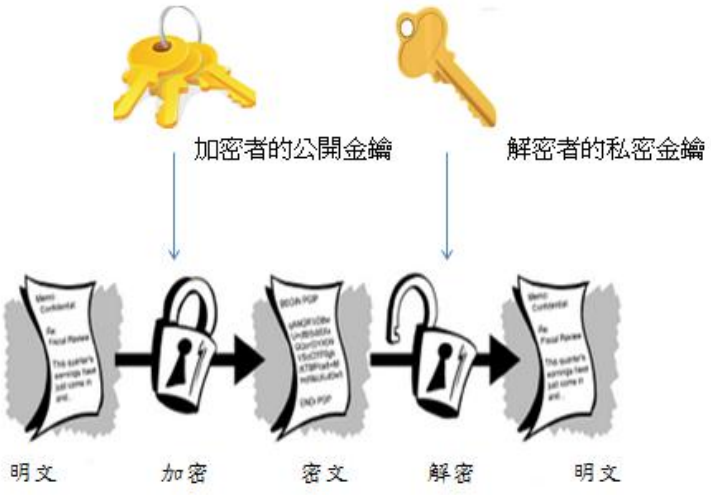

圖 5　非對稱金鑰密碼系統

2.2同態加密

同態是一個函數具有相同代數性質的定義域與對應域的對應關係，在函數運算後仍保有原先的代數結構，意指單位元素(Unit element)、反元素(Inverse element)、二元運算(Binary compute)經過此函數運算後仍然保有原本的特性。

同態加密:若 f(x)為一同態加密函數，f: X → Y，且 a, b ∈ X，則f(a ⊕ b) = f(a)⊗f(b)。此處的⊕、⊗分別為X,Y 代數系統特定的運算元，兩個運算元不一定相同，又因定義域運算元的不同而分為加法同態與乘法同態。

也因此我們可以清楚的瞭解到同態加密是一種特殊的加密方式，如果一套加密系統滿足了同態加密的性質(如:乘法同態、加法同態)表示了這套系統滿足了對密文執行特定的代數運算後在解密的結果會等同於對明文進行運算的結果。

同態加密可以應用在密文之間的運算，如將密文$c_1 = E(m_1)$和密文$c_2 = E(m_2)$做運算得到$c = c_1 \odot c_2$等同於$c = E(c_1 \oplus c_2)$(c 為密文，m 為明文，E 是具同態性質的加密演算法，⊙ 和⊕為特定運算)。而同態加密的最大的好處在於，若一套加密系統符合同態的性質，它可以允許在不經由解密的情況下，仍可讓人對加密後的訊息進行運算、比較…等操作，並且依然保持其正確性。如此一來便可以解決將私密數據委託給第三方做運算，或是自己本身經由繁雜的傳送做運算的問題，像是雲端計算的隱私性保護問題。而不僅僅是雲端運算，全同態更可以保障其他應用上的隱私性問題，像是電子投票系統、私密資訊擷取(PIR)…等。利用同態加密的此種作法不但可以節省運算成本及縮短計算時間(不需多餘的加解密)，還可保障運算過程中的訊息安全。

2.2.1 加法同態

加法同態是指兩個密文做某種運算，會等同於明文做"加法"運算再加密，如下:

m_i為明文, c_i為密文, E 為加密方案

$$c_1 = E(m_1)，c_2 = E(m_2)$$

$$c = c_1 \oplus c_2 = E(m_1) \oplus E(m_2) = E(m_1 + m_2)$$

舉例:若 k 為一個常數，且 E(m)=km，$E(m_1)+E(m_2)=km_1+km_2=k(m_1+m_2)=E(m_1+m_2)$，此運算方式能達到加法同態。

在此以 Paillier 的加密系統[19]來說明，此系統由三個演算法：Key Generation、Encryption 、Decryption 組成。

Key Generation(金鑰生成演算法**)**：假設密文接收者為 Alice，則 Alice 利用 Key Generation 產生自己的公私鑰。其方法如下

1. 選擇兩個大質數 p 和 q

2. 計算 *N=p*q* 和$\lambda=lcm(p-1,q-1)$，而 *lcm*(Least Common Multiple)代表$p-1$ 和$q-1$ 的最小公倍數。

3. 選擇亂數$g \in Z_{N^2}^*$，$Z_{N^2}^*$是比N^2小且與N^2互質的元素且包含單位元素 1 的集合。

4. 函數 L(u)=(u-1)/N

5. 設g的序為l，確保$N|l$ ，亦即 N 可整除 l

6. Alice 的公鑰 $pk=(N,g)$

7. Alice 的私鑰 $sk=(N, \lambda)$

Encryption(加密演算法**)**：假設 Bob 欲利用 Paillier encryption 加密訊息 *m* 給 Alice，則 Bob 利用加密演算法執行以下步驟

1. 欲加密訊息 $m \in Z_n$

2. 選擇亂數 $r \in Z_n^*$

3. 計算密文 C=E(m)=$g^m * r^N \mod N^2$

Decryption(解密演算法**)**：Alice 收到密文 C 後利用解密演算法做以下步驟解密

1. 密文 $C \in Z_{N^2}^*$

2. m=D(C)=($\frac{L(C^{\lambda} \bmod N^2)}{L(g^{\lambda} \bmod N^2)}$) mod N

Paillier加法同態：

假設Bob把m_1和m_2分別加密起來，得到$E(m_1)=g^{m_1} * r_1{}^N$和$E(m_2)= g^{m_2} * r_2{}^N$，把$E(m_1)$ 和$E(m_2)$傳送到資料庫，資料庫在不知道明文的情況下，對兩個密文可做同態運算，如下：

$C=E(m_1)* E(m_2)= (g^{m_1} * r_1{}^N)*(g^{m_2} * r_2{}^N)= g^{m_1+m_2}*(r_1r_2)^N=E(m_1+ m_2)$

Alice可以從資料庫中得到C，但是她無法得到任何有關密文c_1、c_2的明文訊息，但是她可以得到明文$m_1 + m_2$的結果。

2.2.2 乘法同態

乘法同態是指兩個密文做某種運算，會等同於明文做"乘法"運算再加密，如下:

m_i為明文,c_i為密文,E 為加密方案

$$c_1 = E(m_1)，c_2 = E(m_2)$$

$$c = c_1 \otimes c_2 = E(m_1) \otimes E(m_2) = E(m_1 * m_2)$$

舉例:若 e 為某個常數，且$E(m) = m^e$，$E(m_1) * E(m_2) = m_1{}^e * m_2{}^e = (m_1 * m_2)^e = E(m_1 * m_2)$。

我們舉RSA來說明：

Key Generation：假設密文接收者為Alice，則Alice利用Key Generation產生自己的公私鑰。其方法如下

步驟1. 隨意選擇兩個大的質數p和q，$p \neq q$，計算$N=pq$。

步驟2. 根據尤拉函數，不大於N且與N互質的整數個數為$(p-1)(q-1)$

步驟3. 選擇一個整數e與$(p-1)(q-1)$互質，並且e小於$(p-1)(q-1)$

步驟4. 用以下這個公式計算d：$d * e \equiv 1 \ (mod \ (p-1)(q-1))$

步驟5. 將p和q的記錄銷毀。

(N,e)是公鑰，(N,d)是私鑰。將公鑰(N,e)公開，而私鑰(N,d)秘密保存。

Encryption：假設Bob欲利用RSA encryption加密訊息m給Alice，Bob利用Alice的$pk=(N,e)$計算密文C：

$$C=m^e mod\ N$$

Decryption：Alice收到密文C後利用*sk=(N,d)*解密C得

$$m=C^d\ \ mod\ N$$

分析與討論：

$c^d \equiv m^{e*d}(mod\ N)$，ed ≡ 1 (mod p−1)和ed ≡ 1 (mod q−1)。

由費馬小定理可以證明(因為p、q皆為質數)

m^{e*d}≡m (mod p)

m^{e*d}≡m (mod q)

由於p≠q，且p和q皆為質數，因此p和q互質

m^{e*d}≡m (mod p*q)

RSA乘法同態:

Bob產生兩組密文c_1、c_2分別為${m_1}^e(mod\ N)$、${m_2}^e(mod\ N)$並且將其送至資料庫，資料庫在無法得知明文為何的狀況下，進行同態運算。

C=E(m_1)*E(m_2)= ${m_1}^e(mod\ N) * {m_2}^e(mod\ N) = (m_1 * m_2)^e\ mod\ N$

Alice可以從資料庫中得到C，但是她無法得到任何有關密文c_1、c_2的明文訊息，但是她可以得到明文$m_1 * m_2$的結果。

再以一個例子ElGamal來介紹其乘法同態特性

ElGamal 加密演算法有三個部分，分別是金鑰生成(Key generation)、加密演算法(Encryption)以及解密演算法(Decryption)。

金鑰的生成：

1. Alice 取一個生成元g，足以生成一個循環群G，內有q個元素。
2. Alice 任選一個變數 $x \in \{0, \dots, q-1\}$。
3. Alice 計算 $y = g^x \bmod p$。
4. Alice 以 (G, q, g, y)作為她的公鑰，x作為她的私鑰。

加密演算法：

1. Bob 任選一個變數 $r \in \{0, \dots, q-1\}$，並且計算$C_1 = g^r \bmod p$。
2. Bob 加密明文 m，生成 $C_2 = m \times y^r \bmod p$。
3. 傳送密文 (C_1, C_2)給 Alice。

解密演算法：

1. Alice 計算 $C_1^x = g^{rx} = y^r$。
2. 接著計算 $C_2 \times (C_1^x)^{-1} = (m \times y^r) \times (y^r)^{-1} = m$ 即可還原明文。

ElGamal 的乘法同態性質

這小節我們將介紹 ElGamal 加密系統的乘法同態性質。首先利用 ElGamal 加密系統分別以對明文m_1, m_2做加密，產生$(C_1, C_2), (C_1', C_2')$兩組密文，其中：

$$(C_1, C_2) = (g^{r_1}, m_1 \times y^{r_1}) \bmod p$$

$$(C_1', C_2') = (g^{r_2}, m_2 \times y^{r_2}) \bmod p$$

接著對兩組密文(C_1, C_2)與(C_1', C_2')做乘法運算得到$(C_1'', C_2'') = (C_1, C_2) \times (C_1', C_2')$，其中：

$$C_1'' = C_1 \times C_1' = g^{r_1} \times g^{r_2} \bmod p = g^{r_1 + r_2} \bmod p$$

$$C_2'' = C_2 \times C_2' = (m_1 \times y^{r_1}) \times (m_2 \times y^{r_2}) \bmod p = (m_1 \times m_2) \times y^{r_1 + r_2} \bmod p$$

最後做解密：

$$C_2'' \times \left(C_1''^{x}\right)^{-1} = (m_1 \times m_2) \times y^{r_1 + r_2} \times (g^{r_1 + r_2})^{-1} = (m_1 \times m_2) \times y^{r_1 + r_2} \times \left(g^{x(r_1 + r_2)}\right)^{-1}$$

$$= (m_1 \times m_2) \times y^{r_1 + r_2} \times (y^{r_1 + r_2})^{-1} = (m_1 \times m_2)$$

由以上的加解密過程及運算結果，我們可以清楚地看到，對兩組利用ElGamal加密

系統所加密的密文做乘法運算之後，其解密結果會同等於對明文做乘法運算。

2.3Genrty 對稱式全同態加密方案

Gentry 的全同態加密方案有分為兩種，一種是對稱式金鑰的加密方案，另一種則是非對稱式金鑰的加密方案。雖然在對稱式的方案當中，金鑰個數為一以及計算上較為快速使得儲存空間上以及硬體的計算負擔皆為減輕許多，但對於金鑰的管理上以及如何安全的傳送金鑰皆是對稱式金鑰密碼系統的瓶頸或缺點。因此為了安全性以及管理層面上的考量，使用非對稱式的密碼系統是較為正確的選擇，也因此本文所討論以及改進的是目前被使用的非對稱式的加密方案。但是由於 Gentry 所提出的非對稱式的全同態加密方案是根據對稱式的方案演變而成，因此為清楚的了解其脈絡，在本文中我們在此介紹其對稱式加密方案，並在下一章介紹其非對稱式的加密系統。

其對稱式方案之演算法如下:

1. (1)金鑰的生成如下(KeyGen(λ)): (輸入:λ，輸出: 密鑰 p)

說明:產生一個安全參數λ，由此變數可求得密鑰 p、q、r 的長度以及數值，其中 p 的位元數為 $O(\lambda^2)$且 p 為奇數、q 的位元數為$O(\lambda^5)$、r 的位元數為 O(λ)。

(2)加密演算法如下(Encrypt(p, m)): (輸入: p、明文m ∈{0,1}，輸出: 密文 C)

說明:對單一明文m ∈{0,1}進行加密，其加密過後輸出如下:C←m + pq + 2r，其中 q 為一個隨機 $O(\lambda^5)$位元數的亂數。

(3)解密演算法如下(Decrypt(p,C)): (輸入:p、C，輸出: m)

說明:將加密過後的結果 C mod p，再 mod 2 即 m←(C mod p)mod 2，其中 C - C'可被 p 整除。

舉例:假設公私鑰 p 為 101，明文 m 為 1，r 為 5，q 為 9

則加密過程如下: C = pq + 2r +m=11+909=920

而在解密的過程: ((C mod p) mod 2)=11 mod 2=1

因此其方案確保資料正確性。

然而在加密過程當中，C← pq + m +2r，其中 2r + m 被稱為雜訊(noise)，由於 p、q 兩數皆遠大於 2r + m，因此其加密後的結果 C 必定落在 p 的倍數前後少許。也因為其雜訊遠小於 p，因此解密才可得到正確的答案。

在 Gentry 的全同態演算法中，可以執行的同態種類有加法同態、乘法同態、減法同態這三種。

2.4AGCD 問題(Approximate Greatest Common Divisor Problem)

AGCD 問題因為其求解的困難性被利用在密碼學的領域之中，而 Gentry 的全同態加密系統的安全性主要也是基於求解 AGCD 問題。簡單來說若是攻擊者 A 破解了 Gentry 的全同態加密系統則我們可以利用 A 去破解 AGCD 問題。然而何謂 AGCD 問題呢?在下個段落我們會有詳細定義以及介紹。

若有兩組數字x_1以及x_2，其中$x_1 = pq_1, x_2 = pq_2$，若要找到其相同因數 p 並不困難，我們可以利用 GCD 問題來求解得出 p 的結果。但是 A-GCD 問題是指當給定一組數值x_i，且假設x_i=pq_i+2r_i, 其中 i=0,1,…,n 且r_i遠小於 p。在$\forall q_i, r_i$皆是未知的情況下，能否找出其近似的最大公因數p的一個問題。也因此假設 p 若可以被一個演算法有效率的求出的話，Gentry 的全同態加密系統則可被破解。但到至今為止，AGCD 問題都仍被視為是一個相當求解的難問題，也因為如此，Gentry 的全同態加密方案至少以目前而言是安全無虞的。我們也將會在第五章安全性分析對 AGCD 問題以及全同態加密方案作出詳細探討。

2.5語意安全(Semantic Secure)

語意安全是一種加密演算法的安全定義，並且廣泛的被運用在說明加密系統的安全性高度之中，尤其是在非對稱式金鑰加密(Asymmetric Key Encryption, AKE)演算法，是否達到語意安全則是此演算法安全與否的一個課題。那麼對於一個具備語意安全的非對稱式金鑰加密演算法而言，需要達到的條件為何，則是設計加密演算法的設計者所要重視的問題，它必須可以抵抗一個具備有限計算能力(computationally bounded)的攻擊者所提出的攻擊行動。假設攻擊者擁有密文以及加密系統中所提供的加密金鑰，他也無法有效率地從密文當中得到任何有關明文的資訊。簡單來說，若滿足語意安全則必須能夠有效的抵抗選擇明文攻擊(Chosen Plaintext attack, CPA)，也因此一個具備語意安全的加密演算法，攻擊者是無法從密文以及公鑰等現有資訊來破解明文。

語意安全的當初最早是在西元 1982 年由 Goldwasser 和 Micali [14]所提出。其必要條件是必須抵抗選擇明文攻擊，也因此在語意安全的證明上會設計一種類似遊戲的流程，若根據以下流程攻擊者最後可以得到明文資訊，則說明該加密演算法不具備語意安全的能力。另一方面假設到最後隨著流程結束並且參與者贏得該遊戲則說明了該演算法具備語意安全，攻擊者無法從中得到任何明文結果。下列則介紹這套遊戲的流程：

在開始證明是否具備語意安全的遊戲之前，介紹兩方角色參與這項遊戲，分別為攻擊者 A 以及參與者 B。接著步驟如下:

Step 1.攻擊者 A 端可以得知加密演算法以及系統的公開金鑰，因此攻擊者可以在有限的時間內取得任意數量的密文。

Step 2.攻擊者 A 任意產生兩個長度相同的訊息m_0以及m_1，並且將此兩個訊息傳給參與者 B。

Step 3.參與者隨機選擇m_0或m_1進行加密，並且將其密文結果傳送給攻擊者 A。

Step 4.在一般的情況之下，攻擊者 A 猜中明文機率是 1/2+ ε，若 ε 的值微小到可以

忽略不計(negligible)，則代表該演算法具備語意安全。反之，若 ε 的值是無法忽略的話，則代表攻擊者 A 成功的破解該加密演算法，並且不具備語意安全。

以 ElGamal 為例，我們如上流程建立一個遊戲：

(1) 攻擊者 A 知道在這個遊戲中使用 ElGamal 加密演算法以及其公開金鑰(G, q, g, y)，並允許攻擊者在有限的時間內產生任何數量的密文。

(2) 攻擊者 A 任意生成兩個長度相同的訊息m_0, m_1，並且將這兩個訊息傳給參與者 B。

(3) 參與者 B 隨機選擇m_0或 m_1進行加密。由於 ElGamal 加密演算法公開，因此參與者 B 加密時會選取一個變數$r \in \{0, \ldots, q-1\}$，並且計算$C_1 = g^r \bmod p$。接著選擇明文 $m_i, i = \{0,1\}$，生成 $C_2 = m \times y^r \bmod p$。加密完之後將密文$(C_1, C_2)$回傳給攻擊者 A。

此時由於挑戰者在加密過程中，任意選取了一個亂數$r \in \{0, \ldots, q-1\}$，使得攻擊者 A 並沒有辦法從已知條件(G, q, g, y)計算出密文(C_1, C_2)究竟是由m_0, m_1哪一個訊息所加密的，因此攻擊者並無法得知其真正的明文選擇為何，因此了解 ElGamal 具有語意安全，其詳細證明已在[23]被提出。

在第五章安全性證明時將會以我們改良後的全同態加密演算法為例，說明我們提出的全同態加密演算法符合語意安全。

3. Gentry 全同態加密方案回顧

3.1Gentry 全同態加密演算法

在章節 2.2 中我們介紹了 Gentry 最先設計的對稱式全同態加密方案，但是由於密鑰的難管理以及如何安全傳送都是一個相當難以解決的問題，也因此對稱式的全同態加密方案並不實用。在這裡我們將介紹 Gentry 後來提出的公開金鑰密碼系統的全同態加密方案。

全同態加密共分成四個部分，分別是:金鑰生成、加密演算法、同態運算和解密演算法

(1)金鑰的生成如下(KeyGen(λ)): (輸入:λ，輸出: 私鑰 p、公鑰x_i)

(i)選取一個安全參數λ，由此變數可求得η、τ、γ，其中η為私鑰 p 的長度，τ 為公鑰的個數，γ為 q 的長度，其中η=O(λ^2)、γ=O(λ^5)。

(ii)在$[2^{\eta-1}, 2^{\eta})$中，選取一個夠大的奇數 p 當私鑰,亦即$p = (2\mathbb{Z}+1) \cap [2^{\eta-1}, 2^{\eta})$

(iii)選取τ個亂數q_i及τ個亂數r_i，其中$r_i \epsilon (-2^{\rho}, 2^{\rho})$、$q_i \epsilon (2^{\gamma-1}, 2^{\gamma})$，$\rho$為$r_i$的長度，且$\rho$=λ。產生的公鑰為$x_i$=p$q_i$+2$r_i$ ，i = 0, … , ($\tau - 1$)，其中x_0必須是奇數，而且 $x_0 = \max\{x_i | \forall i \in [0, (\tau - 1)]\}$ ，r'為x_0除 p 後的餘數且必須為偶數。

(2)加密演算法如下(Encrypt(pk,m)): (輸入: x_i、明文$m_i \in \{0,1\}$，輸出: 密文 C)

1.假設明文有 t 位元，將明文拆解成單一位元的數字，也就是分成$\{m_1, m_2, \dots, m_t\}$，

∀i, $m_i \in \{0,1\}$

2.對於每個m_i做加密，i = 1, … , t

加密過程如下:

(i)選取子集合 S⊆{1,2,…,($\tau - 1$)}

(ii)計算$\sum_{i \in S} x_i$的數值

(iii)在$(-2^{\rho}, 2^{\rho})$中隨機選取一個亂數r_i

(iv)密文$c_i = (m_i + 2r_i + \sum_{j\in S} x_j) \bmod x_0$ $=(p(\sum_{j\in S} q_j) + 2(r_i + \sum_{j\in S} r_j) + m_i) \bmod x_0$

3.得到密文 C=$\{c_1, c_2, \dots, c_t\}$

(3)同態運算如下(Evaluate(pk,f,$c_1, c_2, \dots c_t$)): (輸入:f、c_i、pk，輸出: c)

(i)將密文傳送到運算端運算:

對任意兩個密文參數c_i, c_j，若想做加法同態則輸出$c_i + c_j$，若想做加法同態則輸出$c_i * c_j$。

(ii)將(i)運算結果彙整成 C'=$c_1 \odot c_2 \circledast \dots \odot c_t$，其中任兩個兩個密文參數$c_i, c_j$中間的運算元不一定相等。

(4)解密演算法如下(Decrypt(sk,c)): (輸入:p、c，輸出: m')

明文:$m' = (\text{c mod p}) \ mod\ 2$。

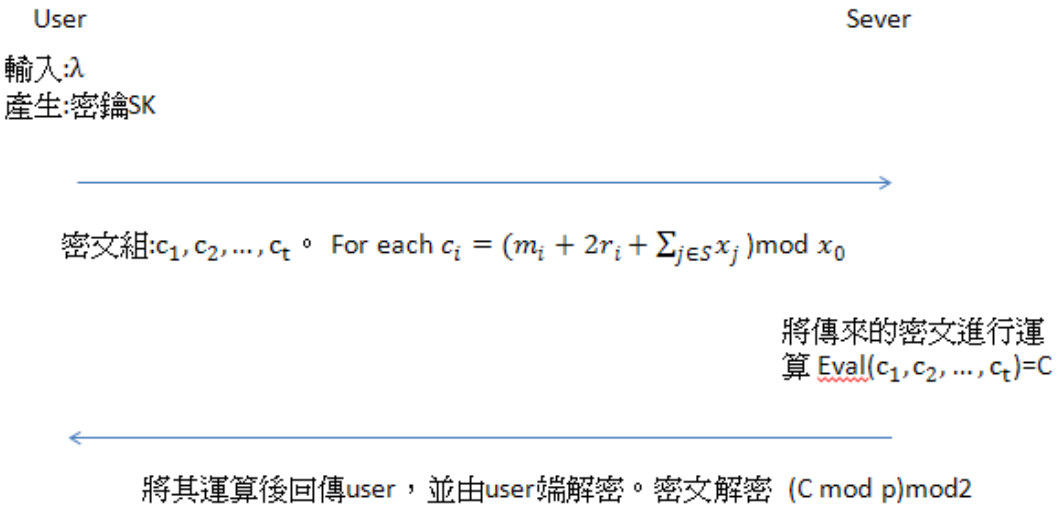

圖 6 全同態加密方案流程

3.2同態加密的正確性

假設明文為兩位元的數字$(m_1||m_2)$，且公鑰為x_i=pq_i+2r_i, i $= 0, \dots, (\tau - 1)$、私鑰為p。

所以密文 C=$\{c_1, c_2\}$=$\{m_1 + 2r_1 + \sum_{j\in S_1} x_j, m_2 + 2r_2 + \sum_{k\in S_2} x_k\}$

同態運算:

(1)(以加法為例)

$c_1 + c_2 = m_1 + 2r_1 + \sum_{j\in S_1} x_j + m_2 + 2r_2 + \sum_{k\in S_2} x_k$=

$p(\sum_{j\in S_1} q_j + \sum_{k\in S_2} q_k)$+2$(r_1$+$\sum_{j\in S_1} r_j + r_2$+$\sum_{k\in S_2} r_k)$+$(m_1 + m_2)$

解密: $((c_1 + c_2)$mod p $)$mod 2=$(m_1 + m_2)$

所以符合加法同態

(2)(以乘法為例)

$c_1 * c_2$ =$(m_1 + 2r_1 + \sum_{j\in S_1} x_j)(m_2 + 2r_2 + \sum_{k\in S_2} x_k)$=$(m_1 * m_2)$+2$(r_2 m_1 + r_1 m_2 + r_2 \sum_{j\in S_1} x_j + r_1 \sum_{k\in S_2} x_k)$+$(m_1 \sum_{k\in S_2} x_k + m_2 \sum_{j\in S_1} x_j + \sum_{j\in S_1} x_j * \sum_{k\in S_2} x_k)$=$(m_1 * m_2)$+2R+pQ，Q,R 為整理之後的結果

所以$((c_1 * c_2)$mod p $)$mod 2=$(m_1 * m_2)$

符合乘法同態

3.3探討 Gentry 方案

雖然 Gentry 的方案可以被用在任意函數的加密演算,但是在實作上有相當的難度。其中最重要的理由是，產生一個公鑰群組需要太多的儲存空間。一般來說，公鑰的大小最小有 800MB[6]，非常消耗設備系統的儲存空間。所以我們就思考是否有其改善的方法，以下是我們改進後的方案。

4. 改進過後的 FHE 方案

4.1我們提出的 FHE 方案

我們提出改進後的 FHE，是以 Gentry 的 FHE 架構下設計出來的。在金鑰生成 KeyGen(λ)、同態運算(Evaluate(pk,f,$c_1, c_2, \dots c_t$))和解密演算法(Decrypt(sk,c))階段都是使用 Gentry 的構想。而此方案的改進最主要是在加密演算法 (Encrypt(pk,$m_i \in\{0,1\}$))階段，我們採取縮減公鑰個數的策略，設計出新的加密階段，以達到效能提升的目的。

(1)金鑰的生成如下(KeyGen(λ)): (輸入: λ，輸出: 私鑰 p、公鑰x_i)

如同之前 Gentry 的方案，但公鑰個數為$\tau' = \lceil \log_3 2^{\tau} \rceil$。私鑰為$p$，且 p 是奇數、公鑰為 x_i=pq_i+2r_i，其中x_0必須是奇數。

其中$x_0 = \max\{x_i | \forall i \in [0, (\tau' - 1)]\}$，$r'$為$x_0$除 p 後的餘數且必須為偶數。

(2)加密演算法如下(Encrypt(pk, m_i)): (輸入: x_i、明文$m_i \in\{0,1\}$，輸出: 密文 C)

1.假設明文有 t 位元，將明文拆解成單一位元的數字，也就是分成$\{m_1, m_2, \dots, m_t\}$，

∀i, $m_i \in\{0,1\}$

2.對於每個m_i做加密，$i = 1, \dots, t$

加密過程如下:

隨機從$S = \{-1,0,1\}$集合選取(τ'-1)個數值a_i。亦即$\forall a_i \in \{-1,0,1\}$，$i = 1, \dots, (\tau'\text{-}1)$

(i)計算$\sum_{i=1}^{(\tau'-1)} a_i x_i$的數值

(ii)在$(-2^{\rho}, 2^{\rho})$中隨機選取一個亂數r_i

(iii)計算 w=$m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j$

(iv)隨機選取$x_k \in \{x_1, x_2, \dots, x_{(\tau'-1)}\}$，

(v)選取A_i

如果 0<w<x_0，則取$A_i \in \{0,1,\dots,6\}$

如果-x_0<w<0，則取$A_i \in \{0,-1,\dots,-6\}$

如果$w_i > x_0$或$w_i <$ -x_0則A_i=0

(vi)密文$c_i = (m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j + A_i x_k) \bmod x_0$。

3.得到密文 C=$\{c_1, c_2, \dots, c_t\}$

(3)同態運算如下(Evaluate(pk,f,$c_1, c_2, \dots c_t$)): (輸入:f、c_i、pk，輸出: c)

如同之前 Gentry 的方案，

(i)將密文傳送到運算端運算:

對任意兩個密文參數c_i, c_j，若想做加法同態則輸出$c_i + c_j$，若想做加法同態則輸出$c_i * c_j$。

(ii)將(i)運算結果彙整成 C'=$c_1 \odot c_2 \circledast \dots \circledcirc c_t$，其中任兩個兩個密文參數$c_i, c_j$中間的運算元不一定相等。

(4)解密演算法如下(Decrypt(sk,c)): (輸入:p、c，輸出: m')

如同之前 Gentry 的方案，

明文:$m' = (\text{c mod p}) mod\ 2$

4.2改進方案的貢獻

我們改進過後的方案，跟 Gentry 的方案相比，只需更少的公鑰即可達成一樣的安全性。因為在 Gentry 的方案中，加密階段對每一把公鑰只有選與不選兩種可能，所以若有 n 把公鑰，以他的加密方案只有2^n種公鑰組合方式。而我們的方案中，增添了負數的概念，在加密階段對每一把公鑰有選、不選和取負值共三種可能。所以若有 n 把公鑰，則改進後的加密方案有3^n種公鑰組合方式。舉例而言:如果在 Gentry 的方案使用了 1000 把公鑰，所以公鑰組合共有2^{1000}種可能性。但以我們改進過後的加密方案，只需 $\log_3(2^{1000})$ ≈640 把公鑰即可達成一樣的安全性，縮減了大約 1/3 的時間去產生公鑰，也減少公鑰的儲存空間，更符合現實的應用層面。

在金鑰生成以及加密時間方面，由於我們所需要的公開金鑰數量低於 Gentry 的金鑰個數，也因此在我們提案中所要的金鑰生成時間與 Gentry 提案中相比，亦只需要將近 67%的時間即可完成。而更重要的是在加密過程中，所需要的時間也只需要 80%的時間即可加密完成。

在此處我們與原提案方式最大的不同之處就是增加了兩個變數$A_i x_k$，其目的為確保 $m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j$ 不能過小，此舉是為了確保資料安全性，詳細說明會在第五章安全性分析中提出。

4.3正確性分析

改進後的方案中，

(1)若對明文 m∈{0,1}做加密，則密文 $c=ap+(2b+m)$，其中 a,b 為特定整數，且 $|2b+m|\leq \tau' 2^{\rho+3}$

證明:

$x_i=pq_i+2r_i, i=0,\dots,(\tau'-1)$

$c\leftarrow (m+2r+\sum_{j=1}^{(\tau'-1)} a_j x_j + Ax_i) \bmod x_0$

所以 $c=(m+2r+\sum_{j=1}^{(\tau'-1)} a_j x_j + Ax_i)+kx_0$

$=(m+2r+\sum_{j=1}^{(\tau'-1)} a_j pq_j + A\,pq_i + kpq_0)$

$=p(Aq_i+\sum_{j=1}^{(\tau'-1)} a_j q_j+kq_0)+2(Ar_i+r+=\sum_{j=1}^{(\tau'-1)} a_j r_j+kr_0)+m$，其中$|k|\leq \tau'$。

因此可化簡 $c=ap+(2b+m)$。

(其中 $a=Aq_i+\sum_{j=1}^{(\tau'-1)} a_j q_j+kq_0$ ，$b=Ar_i+r+\sum_{j=1}^{(\tau'-1)} a_j r_j+kr_0$)

$|2b+m|\leq 2(6|r_i|+|r|+|\sum_{j=1}^{(\tau'-1)} a_j r_j|+|k|r_0)+1$， 且$\forall r_j \leq 2^{\rho}$

所以$|2b+m| \leq 2(7*2^{\rho}+\tau' 2^{\rho}+\tau' 2^{\rho})+1$，

τ'為公鑰的個數，為了維持其安全性，所以$\tau' > 7$

所以$|2b+m|\leq 2\tau'(2^{\rho}+2^{\rho}+2^{\rho})$

$\leq 2\tau' 2^{\rho+2}=\tau' 2^{\rho+3}$

(2) 若對明文$m_i \in\{0,1\}$，$i=1,\dots,\tau'$做加密，$c\leftarrow$(Evaluate(pk,f,$c_1,c_2,\dots c_t$))，則密文 $c=a*p+(2b+m)$，其中 a,b 為特定整數，且$|2b+m|\leq p/8$

根據(1)、(2)可以得知在雜訊較小的情況下執行同態加密、以及同態運算在資訊的正確性來說是正確無虞的。

5. 安全性證明

5.1A-GCD 問題

因為我們的加密方案是改進 Gentry 的方案而來的，所以我們的方案無法被取得私鑰的證明和 5.2 節語意安全的說明，皆跟 Gentry 的原始方案類似[11] [13]。

Gentry 方案的安全性是基於 A-GCD 問題是個難問題。A-GCD 問題是指當給定一組數值x_i，且假設x_i=pq_i+2r_i, i=0,1,…,n。在$\forall q_i, r_i$皆是未知的情況下，能否找出其近似的最大公因數p?

B 是一個企圖攻擊 A-GCD 問題的攻擊者。假設有攻擊者 A 宣稱可以破解我們的 FHE 方案，則 B 可透過 A 去破解 A-GCD 問題。

B 破解 A-GCD 問題步驟如下:

1.問題轉換:

挑戰者 C，希望能破解 A-GCD 問題，所以將數列$(x_0, x_1, \dots, x_\tau)$傳送給 B，

x_i=pq_i+2r_i, i=0,1,…, τ，若x_0不是奇數或有任何一個$x_i > x_0$，其中 i=1,…, τ，則 C 再另外生成一組數列，直到符合上述的條件為止。

2.LSB 預測(目的為求出 z 除 p 之後所得出的商為奇數或是偶數):

B 任意取一個 z∈[0,2^γ)，γ 為公鑰的位元長度，z= $q_p(z) * p + r_p(z)$，$|r_p(z)_z| < 2^\rho$，而$q_p(z)$為 z/p 的商、$r_p(z)$為 z/p 的餘數，此時$q_p(z), r_p(z)$還是未知數，此時 B 去執行 LSB(Last Significant Bit)，並且程序如下

LSB (z, $x_0, x_1, \dots, x_\tau$):(輸入:z、公鑰x_i，輸出: $q_p(z)$的 LSB)

(i)選一個干擾數值$r_j \in (-2^{\rho'}, 2^{\rho'})$，$m_j \in \{0,1\}$，且隨機從$\{-1,0,1\}$集合選取τ個數值$a_i$，i = 1, … , τ，$\forall a_i \in \{-1,0,1\}$

(ii)$c_j \leftarrow \left(z + m_j + 2r_j + Ax_k + \sum_{i=1}^{\tau} a_i r_i\right) \bmod x_0$，A 的取值為{0,1,…,6}

(iii)因為攻擊者A可有效破解本全同態加密系統，所以 B 將c_j傳送給A，此時A將解密c_j後

的數值a'_j傳回給他。($a'_j = \mathrm{m}_j \oplus \lfloor r_p(z) \rfloor \bmod 2$)。

(iv)$b_j \leftarrow a'_j \oplus$parity(z) $\oplus \mathrm{m}_j$，其中 parity(z)代表 z 的最後一個位元值，也就是 z 為奇數或是偶數(如 z = 1001，所以其 parity(z)代表最後一個位元的 1)。其中b_j所代表的是$q_p(z)$的最後一個位元。

(v)Ɛ是 A 猜中明文$\mathrm{m}_j \oplus \lfloor r_p(z) \rfloor \bmod 2$的 advantage,重複上述(i)~(iv)步驟 poly(λ)/Ɛ次，其中 poly(λ)是一個跟λ相關的多項式，最後由出現較多次的$\mathrm{m}_j \oplus \lfloor r_p(z) \rfloor \bmod 2$做為 A 的選擇(如:假設對單一明文執行十次的 LSB 運算，得出七次結果為 1，三次結果為 0，由於 A 可有效的破解加密方案，因此選擇結果七次的 1 做為解密的結果)。

假設 A 可以準確地猜測 p 的數值，所以$a'_j = r_p(z)(mod\ 2) \oplus \mathrm{m}_j$，又 $q_p(z)(mod\ 2) = r_p(z)(mod\ 2) \oplus$parity(z)$= b_j$，因此可以得知$q_p(z) \pmod 2$的奇偶性質，並且可利用以上資訊透過 Binary GCD 來得出私鑰 p。

3.Binary GCD[11][13]:

任意選取兩整數$z_1 = q_p(z_1) * p + r_p(z_1)$，$z_2 = q_p(z_2) * p + r_p(z_2)$。

B-GCD(q_{z_1}, q_{z_2}):(輸入: $q_p(z_1)$、$q_p(z_2)$，輸出: z'=1*p + r)

根據 LSB 預測，我們可以得知$q_p(z_1), q_p(z_2)$的奇偶性質。並且去執行 Binary GCD，其程序如下:

(i)假設$z_2 > z_1$，則交換彼此數值$z_1 \Leftrightarrow z_2$。

(ii)透過 LSB 可以$q_p(z_1), q_p(z_2)$的最後一個位元，並且表示$b_i = q_p(z_i) mod 2$

(iii)假設兩個$q_p(z_1), q_p(z_2)$數字皆為奇數，則$z_1 \leftarrow z_1 - z_2$。並且設定$b_1 \leftarrow 0$

(v)如果對任意一個z_i，其中它的$b_i = 0$，那麼計算$z_i \leftarrow (z_i - parity(z_i))/2$

終止條件:當z_2隨著遞迴成為 0 時，即可終止此流程，並且將z_1設定為 z'。

4.找出私鑰 p:

根據 Binary GCD，B 選取兩整數$z_1 = q_{z_1} * p + r_{z_1}$，$z_2 = q_{z_2} * p + r_{z_2}$，其中$z_1$、$z_2$可以由公開金鑰中選擇，並且執行 Binary GCD 並且得出 z'。最後則是將$z_1 \leftarrow z_1$，$z_2 \leftarrow$z'，再

去執行一次 Binary GCD，此時紀錄每一回合的b_1將其成為一個二進位的數字，此數字即為$q_p(z_1)$，故此我們可得知q_{z_1}的實際數值，所以 B 可得知 p=$\lfloor z_1/q_{z_1} \rceil$。B 破解了 A-GCD 問題。

5.2語意安全

假設攻擊者 A，已經知道使用者 B 所使用的加密方法以及公開金鑰，也知道全部的明文種類。A 攔截了 B 傳送一個加密過後的密文，但是 A 仍然無法根據所有的明文種類和加密方案去推測出 B 傳送的是何種明文，則此種加密方案具有語意安全。

因為 Gentry 方案跟我們的方案都是非決定性的加密方案，除此之外對於相同的明文 m 進行多次不同的加密，所得出來的密文其相同機率是非常低的，應此對攻擊者而言並無法根據所得到的密文訊息來判斷明文為何，並且私鑰 p 無法被攻擊者有效地破解，所以我們的方案也具備了語意安全。

5.3為何需要Ax_k

因為我們的改進方案在加密階段對每一把公鑰有選、不選和取負值共三種可能。則在計算$(m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j)$的數值時，$|(m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j)|$跟 Gentry 的方案比起來有更多的機會讓此數值比 p 值來的小，如果有此種情形，則攻擊者可以將此數值直接 mod 2，即可得出明文m_i。但是如果我們多加一個Ax_k數值，則可保障當$\sum_{j=1}^{(\tau'-1)} a_j x_j$數值是小正數或小負數時，會讓$|(m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j + Ax_k)|$可能>$x_0$。因為 mod x_0的關係，所以改變了原本計算的數值，讓攻擊者無法認定在$|(m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j + Ax_k)|$是一個比 p 小的數值時，可以用 mod 2 的方法得出明文，所以才考慮增加Ax_k的數值。

A 的取值分析:在|w|=$|m_i + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j|$<$x_0$，我們才增加$Ax_k$數值。我們希望加入的$Ax_k$數值只是為了干擾攻擊者的猜測公鑰的方法，所以只要達到干擾目的即可。為了方便說明，因此我們假設 w>0，0≤w+ Ax_k <$3x_0$ ⟹0≤ A<$(3x_0 - w)/\ x_k \leq 3x_0/\ x_k$，又公鑰的長度為γ，所以$\forall i, x_i \in [2^{\gamma-1}, 2^{\gamma})$，所以$x_0/\ x_k$ <2。故 0≤ A ≤3*2=6。所以 A 的取值範圍為{0,1,…,6}。

6. 系統效能分析與比較

在第六章，我們會根據 Gentry 以及我們改良過後的提案方式來設計程式，而這份程式最主要是測試兩種方案下所需要的計算時間為何以及相關比較。

在開始之前，先介紹所使用的程式軟體以及硬體設備為何:

1) 程式軟體: MATLAB

2) 硬體設備: Intel i7-740QM processor (1.73GHz,6MB L3 cache)

4GB DDR3 Memory

3)系統環境: WIN7

程式內容最主要是在測量兩個部分，第一個是測試在不同方案下公鑰生成所需時間，第二個則是測量對單一個位元進行加密所需要的時間為何。而在兩個方案當中，私鑰 p 的位元長度、亂數 p 跟亂數 r 的位元長度都是根據安全參數λ所決定，進而計算出金鑰生成以及加密時間兩者之間的差異。

6.1金鑰生成所需時間

在此部分，我們先回顧一下兩種方案金鑰生成的方式

Gentry 的全同態加密方案:

公開金鑰的生成:

Step1:選取一個安全參數λ，由此變數可求得η、τ、γ，其中η為私鑰 p 的長度，τ 為公鑰的個數，γ為 q 的長度，其中η=O(λ^2)、γ=O(λ^5)。

Step2:在$[2^{\eta-1}, 2^{\eta})$中，選取一個夠大的奇數 p,亦即$p = (2\mathbb{Z}+1) \cap [2^{\eta-1}, 2^{\eta})$

Step3:選取τ個亂數q_i及τ個亂數r_i，其中$r_i \epsilon (-2^{\rho}, 2^{\rho})$、$q_i \epsilon (2^{\gamma-1}, 2^{\gamma})$，$\rho$為$r_i$的長度，且$\rho$=λ。產生的公鑰為$x_i$=p$q_i$+2$r_i$ ，i = 0, … , ($\tau - 1$)。

在此處我們將參數λ設成 5，並將τ設定為 1000，其公鑰大小大約為 10MB。

我們改良過後的全同態加密方案:

公開金鑰的生成:

我們的方案最主要是從 Gentry 的全同態加密方案改良而成,因此在參數方面我們所選擇的跟原方案相同。唯一的不同就是金鑰個數，在此處我們只需要$\log_3(2^{1000}) \approx 640$把金鑰個數去進行測試。

Our scheme's key generation time(單位:秒)	Gentry's key generation time(單位:秒)
90.4274	139.933
90.3806	146.001
88.2574	138.638
91.4634	132.554
82.7818	127.812
86.8458	129.403
81.5026	128.638
80.8598	127.484
81.7693	128.233
85.863	128.092

圖 7 公開金鑰所需時間

在這裡時間的單位是秒(sec)，我們可以發現出我們所改良的方案所能降低的時間較 Gentry 的方案高出許多，而其中所降低的比率大約為 0.36 左右，直觀的來看所降低的時間大致上跟所降低的公鑰個數相同。

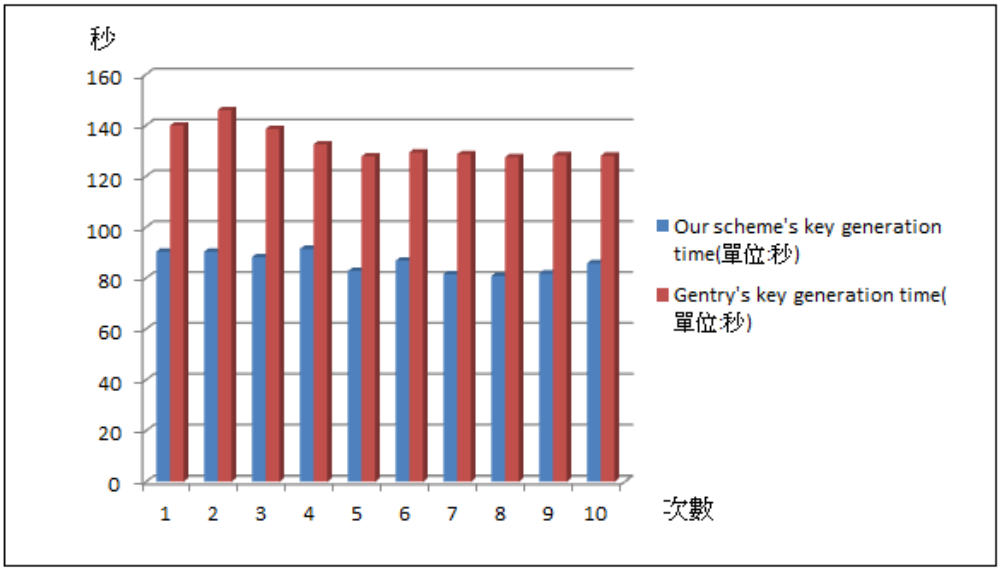

圖 8 金鑰生成時間

6.2加密過程所需時間

相同地在這裡我們先回顧兩個方案所使用的加密方案，接而進行比較。

Gentry 的全同態加密方案

1.假設對單一明文 m 進行加密，且 m∈{0,1}

2.加密過程如下:

(i)選取子集合 S⊆{1,2,…,$(\tau-1)$}

(ii)計算$\sum_{i\in S} x_i$的數值

(iii)在$(-2^{\rho}, 2^{\rho})$中隨機選取一個亂數r_i

(iv)密文$c = (m + 2r_i + \sum_{j\in S} x_j) \bmod x_0$

在此處我們同樣的將安全參數λ設成 5，並將τ設定為 1000。

我們改良過後的全同態加密方案:

1.假設對單一明文 m 進行加密，且 m∈{0,1}

2.加密過程如下:

隨機從$S = \{-1,0,1\}$集合選取$(\tau'\text{-}1)$個數值a_i。亦即$\forall a_i \in \{-1,0,1\}$，$i = 1, \dots, (\tau'\text{-}1)$

(i)計算$\sum_{i=1}^{(\tau'-1)} a_i x_i$的數值

(ii)在$(-2^{\rho}, 2^{\rho})$中隨機選取一個亂數r_i

(iii)計算 w=$m + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j$

(iv)隨機選取$x_k \in \{x_1, x_2, \dots, x_{(\tau'-1)}\}$，

(v)選取A_i

如果 0<w<x_0，則取$A_i \in \{0,1, \dots,6\}$

如果-x_0<w<0，則取$A_i \in \{0, -1, \dots, -6\}$

如果w_i>x_0或w_i<-x_0則A_i=0

(vi)密文$c_i = (m + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j + A_i x_k) \bmod x_0$。

其中安全參數λ為 5，τ為 640。

Our scheme's running time(單位:秒)	Gentry's running time(單位:秒)
0.824004	0.920406
1.058006	0.858006
0.824004	0.795605
0.870804	1.18561
0.608404	1.01401
0.855204	1.04521
0.724004	1.07641
0.639604	0.842405
0.855204	1.17001
0.780005	1.23241

圖 9 加密時間

由圖九我們可以發現大部分的狀況下，我們改良後的提案方式所需要的加密時間都比原方案的加密時間低，少部分的情況會高於 Gentry 的原方案原因有二:

1. 選擇較多的金鑰進行運算。
2. 加密過後的 w=$m + 2r_i + \sum_{j=1}^{(\tau'-1)} a_j x_j$<p，因而多出加$A_i x_k$的動作。

而根據測試十次的結果，可以計算本方案加密的時間與 Gentry 的方案相比大約可減少 20%的時間，因此再加密過程中，我們的加密方案在大部分的情況之下優於 Gentry 所提出的全同態方案。

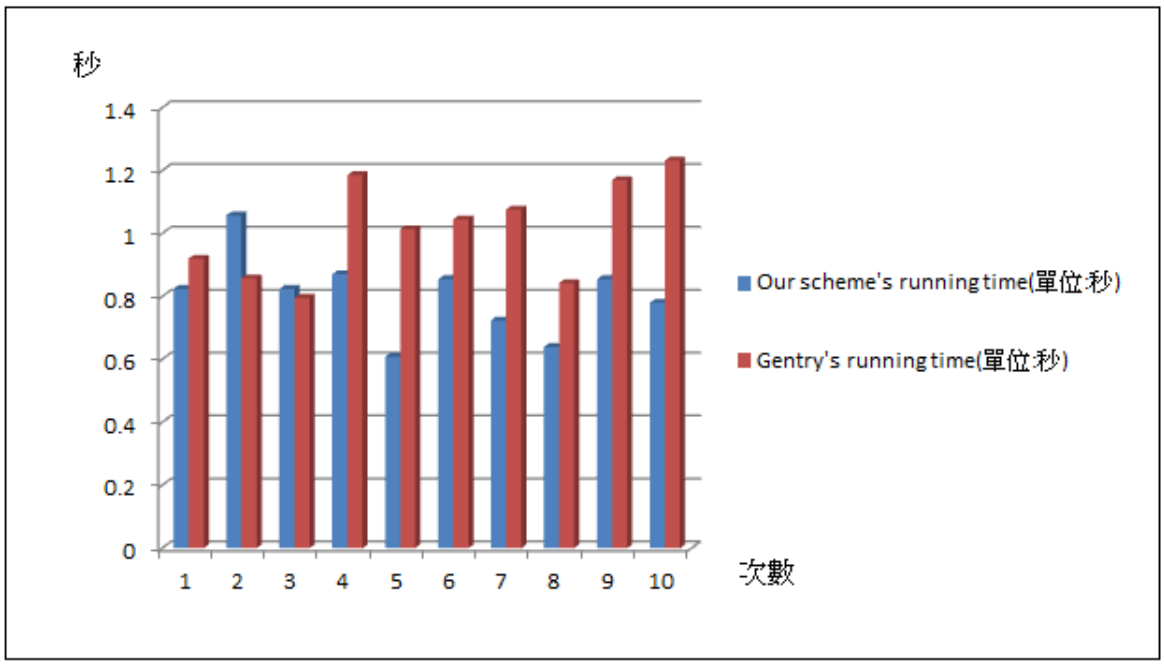

圖 10 加密時間

7. 全同態加密之應用

我們在 1.5 節介紹了許多與全同態加密的相關應用，像是對雲端上的資訊進行加密儲存以確保隱私性之外、更可利用全同態加密的性質對加密過後的訊息做運算。除此之外全同態加密更廣泛被運用在許多協定上，像是在 1.5 節也提出過的雙方相等性驗證。因此在本章節將介紹兩篇文獻利用全同態加密演算法來達到雙方相等性驗證之應用，其協定可讓雙方在不洩漏任何自身資訊的情況下，進行秘密計算來了解彼此的資訊是否相等。

7.1邱姓學者提出的雙方相等性驗證

邱姓學者在 2012 年提出了具隱私保護功能之兩方相等性驗證機制之提案[8]，所提出的協定可以讓雙方都能夠驗證彼此的私密資訊是否相同，其協定如下:

Alice 個人資訊 a，Bob 為 b，a,b∈ Z_N，E()為同時具備 commutative encryption 以及 fully homomorphic 性質的加密演算法，A 為 Alice 的公鑰，所以E_A為 Alice 的同態公開金鑰，同理，E_B為 Bob 的同態公開金鑰

1. Alice 生成亂數$r = r_x + r_y, r_x, r_y, r \in Z^*$，接著計算$a \times r_x$並且用$PK_A$加密得到$E_A(a \times r_X)$，再將$r_y$和$E_A(a \times r_X)$傳送給 Bob。
2. Bob 在收到r_y和$E_A(a \times r_X)$之後，計算$b \times r_y$並且用PK_A加密得到$E_A(b \times r_y)$，並挑選一個隨機的亂數$q \in Z^*$加密後得到$E_A(b \times r_y)$。接著利用全同態性質運算得到$E_A\left(q(ar_X + br_y)\right)$，最後 Bob 以自己的公鑰$PK_B$加密得到$E_B E_A\left(q(ar_X + br_y)\right)$傳送給 Alice。
3. Alice 收到 Bob 傳來的資訊後，利用可交換式加密的特性，先用自己的私鑰SK_A對$E_B E_A\left(q(ar_X + br_y)\right)$解密，$D_A E_B E_A\left(q(ar_X + br_y)\right) = E_B\left(q(ar_X + br_y)\right)$。接著計算$E_B(r^{-1})$，利用乘法同態加密性質運算$E_B(r^{-1}) \times E_B\left(q(ar_X + br_y)\right) = E_B\left(\frac{q(ar_X + br_y)}{r}\right)$

並將運算結果傳給 Bob。

4. Bob 在收到 $E_B\left(\frac{q(ar_X+br_y)}{r}\right)$ 之後用自己的私鑰 SK_B 解密並且 $mod\, p'$ 得到 $D_B E_B\left(\frac{q(ar_X+br_y)}{r}\right) mod\, p' = \frac{q(ar_X+br_y)}{r}$，接著將$q$除掉。假如$\begin{cases}\frac{(ar_X+br_y)}{r} = \text{b} \Rightarrow \text{a} = \text{b} \\ \frac{(ar_X+br_y)}{r} \neq \text{b} \Rightarrow \text{a} \neq \text{b}\end{cases}$。

Bob 將判斷結果連同$\frac{q(ar_X+br_y)}{r}\ mod\, p'$與$E_A\left(q(ar_X+br_y)\right)$傳送給 Alice。

Alice 在收到 Bob 傳遞來的資訊後，以自己的密鑰SK_A將$E_A\left(q(ar_X+br_y)\right)$解密得到 $M = D_A E_A\left(q(ar_X+br_y)\right) = q(ar_X+br_y)$，並計算$N = \frac{q(ar_X+br_y)}{r}\ mod\, p' \times r$。此時 Alice 便能夠比較$q(ar_X+br_y)$是否等於$\frac{q(ar_X+br_y)}{r}\ mod\, p' \times r$，再進一步的判斷若 $\begin{cases} a \mid \frac{q(ar_X+br_y)}{r} \Rightarrow a = b \\ otherwise \Rightarrow a \neq b \end{cases}$。

由以上協定可以讓雙方彼此擁有的秘密資訊在不需要公開的情況下進行比較，判斷其是否相等，使其並無洩漏之虞。而在第二個步驟，Bob 在接收 Alice 的密文之後，開始對自己擁有的訊息進行加密並進行同態運算，因此本方案所提出的全同態加密演算法可適用於邱姓學者的研究之中，並讓其可實行。

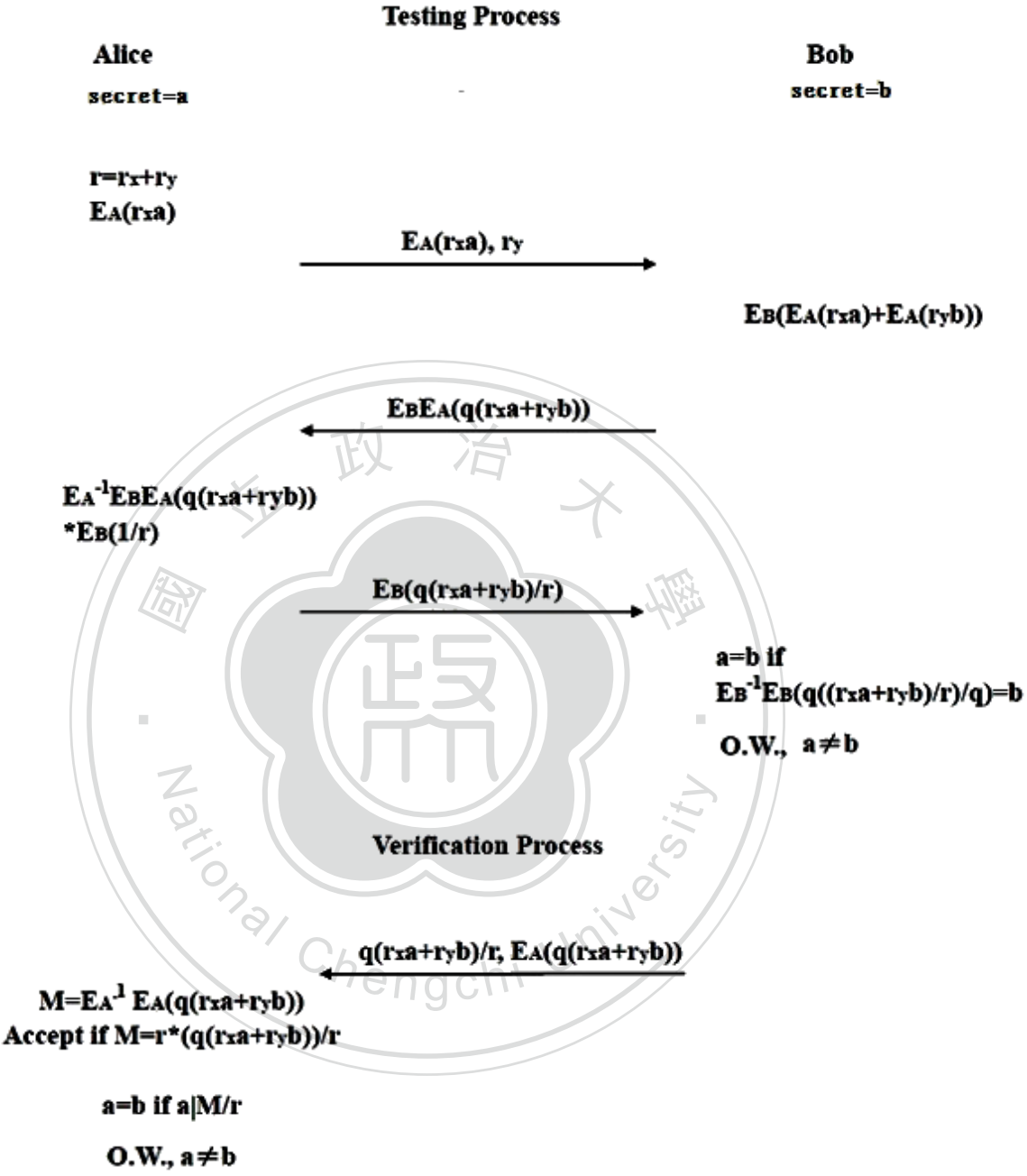
Testing Process
Alice
secret=a
Bob
secret=b
r=rx+ry
EA(rxa)
EA(rxa), ry
EB(EA(rxa)+EA(ryb))
EBEA(q(rxa+ryb))
EA-1EBEA(q(rxa+ryb))
*EB(1/r)
EB(q(rxa+ryb)/r)
a=b if
EB-1EB(q((rxa+ryb)/r)/q)=b
O.W., a≠b
Verification Process
q(rxa+ryb)/r, EA(q(rxa+ryb))
M=EA-1 EA(q(rxa+ryb))
Accept if M=r*(q(rxa+ryb))/r
a=b if a|M/r
O.W., a≠b

圖 11　協定與驗證流程

7.2FNP 隱私性交集相等性驗證

假設 A,B 兩方擁有其私密集合，其裡面元素需大於或等於一項，其元素代表著 A,B 兩方擁有的多筆秘密資訊，若想比較其各自集合內元素相等的個數可透過此協定來完成，如圖十二。

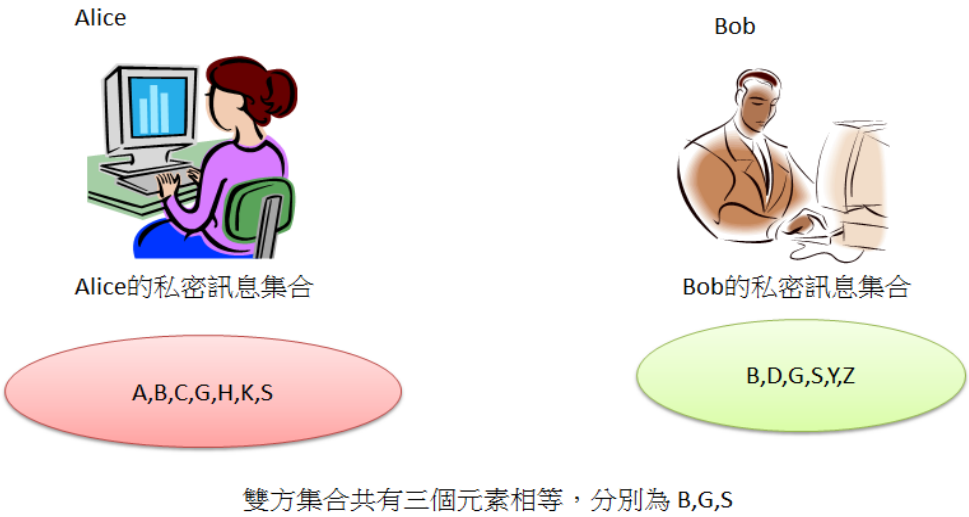

圖 12 比較交集個數

假設:Alice 的私密集合為 T=$\{t_1, t_{2,\dots} t_m\}$，Bob 的私密集合為 Y=$\{y_1, y_2, \dots y_n\}$，我們想得知 T，Y 集合內相等元素個數，也就是|T∩Y|。利用以下協定可以得知其交集個數並且不會洩漏其自身秘密訊息。

Step1:Alice 選擇一個全同態的加密方案，生成其公私鑰，並且傳送一個數值 h 給 Bob

Step2:Alice 建立一個多項式T(x)=$\prod(x - t_i)$，接著算出此多項式的各個係數a_k，即$\mathrm{T}(\mathrm{x}) = \sum_{k=0}^{k=m} a_k x^k$，而此多項式的 m 個根分別為$t_1, t_{2,\dots} t_m$

Step3:Alice 把多項式的係數a_k加密並且傳送給 Bob $(\{\mathrm{E}(a_0), \mathrm{E}(a_1), \dots, \mathrm{E}(a_k)\})$

Step4:Bob 在接收到多項式a_k係數後，不需要解密並且將加密後的係數還原其多項式成為$\mathrm{E}(a_0) + \mathrm{E}(a_1)x + \cdots + \mathrm{E}(a_m)x^m$

Step5:Bob 隨機選擇一個亂數 c，並且將自己集合內的元素代入多項式求得

$E(T(y_j))$，因為滿足全同態定義所以$E(a_0) + E(a_1)E(x) + \cdots E(a_m)E(x^m)$= $E(a_0 + a_1x + \cdots + a_mx^m)$=E(T(x))，接著計算$E(c * (T(y_j)) + h)$並且將多個結果回傳給 Alice

Step6:Alice 在收到 Bob 的結果後，將其解密，若 Y 集合內元素$y_j \in |T \cap Y|$，則 Alice 得到 h，相反，則得到一個無意義的亂數

Step7:計算 h 的個數為何，並且傳送給 Bob

由上面協定之中，我們可以計算雙方集合元素相等個數比較，然而在協定之中 Alice 或是 Bob 只知道其相等個數，其他的秘密訊息都不會被洩漏，而選擇全同態加密演算法的原因是在這套協定當中 Step5 Bob 要對自己的訊息進行運算$E(c * (T(y_j)) + h)$，而對於多種運算元中，只有全同態的加密方案才符合此協定之需求。

圖 13 協定流程

7.3相等性驗證之應用

相等性驗證所能推廣出來的應用相當廣泛，例如線上拍賣競標，線上商業談判，電子投票，雲端資料庫計算，線上登入…等，都是可以利用此類技術加以解決，尤其是現在資料庫系統的普及，使得儲存資料以及對資料進行分析已經是商業上面不可或缺的分析工具時，我們應該對這些重要資料的安全風險更加謹慎，例如就商業面而言，我們可以得知該項產品最終的總銷售數量，但卻無法得知該產品各月份的銷售量，這些對企業來說，或許就是他們想要隱藏的秘密資訊，或是企業併購時，我們只可得知雙方對於此併購金額上的共識相不相同，但是卻無法得知雙方希望買下或是賣出的金額。而我們在 7.1 以及 7.2 所提出的相等性驗證皆可以運用在以上我們所提供的應用上，並且讓其具可行性且易於實行。

8. 結論及未來展望

本研究最大的貢獻，是在於我們改良 Gentry 的全同態加密方案。讓加密所需要的公鑰並不需要佔據太多的硬體空間或是資源，利用減少公鑰個數來達到降低所需空間，並再資料隱私的情況下，同樣的進行對資料加密以及同態運算的正確性，並且與 Gentry 的全同態相比，我們降低公鑰所需空間使其易於實作，並且在安全性以及正確性上並不會造成任何影響使其依舊保有其隱私性、正確性、安全性三項特點。

在未來，我們會針對所選取的集合 S 大小對應公鑰的數量來進行進一步的研究。雖然增加集合 S 會使降低的公鑰數量更為明顯，但是在正確性以及安全性來說，可能會造成疑慮。因此如何在兩端之間求出最佳的平衡數量是我們未來所要進一步研究的課題。

9. 參考資料

[1]B.Applebaum, D.Cash, C.Peikert, and A.Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In CRYPTO2009, vol.5677 of LNCS, pp 595–618. Springer,2009.

[2]D.Boneh, E-J.Goh, and K.Nissim. Evaluating 2-DNF formulas on ciphertexts. In CRYPTO 2005, vol. 3378 of LNCS, pp 325–342, Springer,2005.

[3]D.Boneh, G. D Crescenzo,R.Ostrovsky, G.Persiano: Public key encryption with keyword search. In EUROCRYPT 2004., vol. 3027 of LNCS,pp. 506–522.Springer,2004.

[4]E. Biham, A.Shamir: Differential cryptanalysis of DES-like cryptosystems.In CRYPTO1990 : vol.4 of LNCS, pp. 2-21. Springer,1991.

[5]Z.Brakerski and V.Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In CRYPTO 2011, vol. 5677of LNCS,pp 505-524.Springer 2011.

[6]J.Coron, A.Mandal, D.Naccache, and M.Tibouchi. Fully-homomorphic encryption over the integers with shorter public-keys. In CRYPTO 2011, vol.6841 of LNCS, pp. 487-504. Springer,2011

[7]N.Courtois, J.Pieprzyk, Cryptanalysis of block ciphers with overdefined systems of equations. In ASIACRYPT 2002,vol. 2501of LNCS, pp267–287.Springer 2002.

[8]S.Ciou and R.Tso A privacy preserved two-party equality testing protocol,In ICGEC 2011,pp.220-223,2011.

[9]T.ElGamal.A Public key cryptosystem and a signature scheme based on discrete logarithm. In IEEEE Trans.Inform.Theory,vol.31,pp469-472,1985.

[10]M.van Dijk, C.Gentry, S.Halevi, and V.Vaikuntanathan. Fully homomorphic encryption

over the integers. In EUROCRYPT'10, vol.6110 of LNCS, pp.24–43. Springer 2010.

[11]C.Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford University, 2009.crypto.stanford.edu/craig. available at: https://docs.google.com/viewer?url=http%3A%2F%2Fcrypto.stanford.edu%2Fcraig%2Fcraig-thesis.pdf

[12]C.Gentry. Fully homomorphic encryption using ideal lattices. In STOC'09, pp 169–178. ACM, 2009.

[13]C.Gentry and S.Halevi. Implementing gentry's fully-homomorphic encryption scheme. In EUROCRYPT, vol.6632 of LNCS, pp 129–148. Springer, 2011.

[14]S.Goldwasser and S.Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information.In STOC'82,pp365-377.ACM1982.

[15]Y.Ishai and A.Paskin. Evaluating branching programs on encrypted data. In TCC, vol.4392 of LNCS, pp 575–594. Springer, 2007.

[16]V.Lyubashevsky, C.Peikert, and O.Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT, vol.6110 of LNCS, pp 1–23.Springer,2010.

[17]C.A.Melchor, P.Gaborit, and J.Herranz. Additively homomorphic encryption with -operand multiplications. In CRYPTO, vol.6223 of LNCS, pp 138–154. Springer, 2010.

[18]C.Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In STOC'09, pp 333–342. ACM, 2009.

[19]P.Paillier. Public-key cryptosystem based on composite degree residuocity classes.In EUROCRYPT1999.vol.1592 of LNCS,pp223-238. Springer,1999.

[20]O.Regev. On lattices, learning with errors, random linear codes, and cryptography. In STOC'05, pp 84–93. ACM, 2005.

[21]O.Regev. The learning with errors problem. In IEEE Conference on Computational

Complexity, pp 191–204. IEEE Computer Society, 2010.

[22]R.Rivest,A.Shamir and L.Adleman.A method for obtaining digital signatures and public-key cryptosystem.In Commun'77,vol.21,pp120-126.ACM1977.

[23]Y.Tsiounis and M.Yung. On the security of ElGamal based encryption. In Public Key Cryptography 1998.vol.1431 of LNCS.pp117-134. Springer,1998.

[24]B.Waters: Efficient identity-based encryption without random oracles. In EUROCRYPT 2005. vol. 3494 of LNCS, pp. 114–127. Springer,2005.

[25]吳承鋒、陳漢光、左瑞麟 利用ElGamal加密的雙方相等性驗證協議 全國計算機會議,pp183-191,2011.