國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

以智慧型手機為使用者端裝置之安全分散式儲存系統實作

End users of smart phone as secure distributed storage system implementation

研 究 生: 顏豪緯

指導教授: 曾文貴 教授

中 華 民 國 九 十 九 年 六 月

以智慧型手機為使用者端裝置之安全分散式儲存系統實作

End users of smart phone as secure distributed storage system implementation

研 究 生：顏豪緯　　Student：Hao-wei Yen

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

以智慧型手機為使用者端裝置之安全分散式儲存系統實作

學生:顏豪緯　　指導教授:曾文貴 博士

國立交通大學資訊科學與工程研究所　　碩士班

摘要

雲端計算將有可能被視為下一代網路及電腦架構，它具有移動式應用軟體，以及集中的大型數據中心，其管理的數據和服務可能不完全可信，但它卻可以幫我們解決需要大規模運算的成本，讓大家都可以從事需要大規模運算的研究與相關應用活動。然而雲端的好處確實讓我們未來的生活有了大大的改變，不過不可忽視的是它同時也帶來了許多新的安全性議題，要如何確保資料儲存在雲端裝置上的完整性，是我們研究的主要目標，另外我們的行動式智慧裝置系統使用的是android，主要是它因為採用開放原始碼的策略，讓軟體開發者能盡情的開發符合使用者需求的軟體。我們採用 public key system，並利用 Erasure Code 來確保我們的資料完整性，至於加密的部分我們採用 paring 來讓資料達到保密性。我們的系統能讓儲存的成本降低，且提高資料的可靠度及可信度。

致謝

本篇碩士論文能夠順利完成，在此要感謝我的指導老師曾文貴教授，老師不僅在學術上對我細心的指導，在生活和做人處事上更是不時的提點以及教導我，皆令我感念在心，並由衷地感謝我的老師。另外，我要同時感謝蔡錫鈞教授、謝續平教授以及中央研究院資訊科學研究所的呂及人教授擔任我的口試委員，並在口試時給了我許多意見以及對論文上的修正，使得本篇論文能更加的完整。

感謝實驗室的林孝盈學姐、沈宣佐學長以及陳毅叡學長的細心指導，另外還要感謝同學羅日宏、官振傑和學弟妹們在研究上的討論外，在生活上的經驗分享，也都給了我莫大的幫助，謝謝你們這段期間的付出以及一切的協助。

最後要感謝我的家人，感謝父母的栽培，感謝姊姊以及哥哥的關心，讓我在精神或物質上都無後顧之憂，因為你的的支持才讓我能順利的完成學業。

感謝所有在我求學生涯中曾幫助過我的人，謹以此文獻給我摯愛的家人以及所有關心我的師長、朋友們。

目錄

中文摘要 I
致謝 II
目錄 III
圖目錄 IV
表格目錄 V
第一章 引言 1
1.1 介紹 1
1.2 雲端計算的發展 2
1.3 雲端所面臨的問題 2
1.4 Android 的發展 3
1.5 問題的陳述 4
1.5.1 加入 Erasure code 機制 4
1.5.2 管理機制 5
第二章 背景知識 6
2.1 儲存系統 6
2.2 橢圓曲線密碼學 7
2.2.1 Pairing 的背景基礎 11
2.2.2 PBC 上的 pairing 介紹 12
2.3 Secure Decentralized Erasure Code 13
第三章 系統架構 19
3.1 系統建置 19
3.2 使用者操作 23
3.3 服務型伺服器的資料儲存 26
第四章 系統實作 27
4.1 開發環境建置 27
4.1.1 選擇 JPBC 的因素 28
4.2 實作上面臨的問題 29
4.2.1 JPBC library 29
4.3 Android 上的效能 33
4.4 安全度分析 35
第五章 結語 36
5.1 研究成果 36
5.2 未來發展 37
參考文獻 38

圖目錄

圖 1.1:雲端現有架構..... 2
圖 2.1:橢圓曲線加法運算..... 8
圖 2.2:橢圓曲線加法運算..... 8
圖 2.3:橢圓曲線 Type A 的參數..... 13
圖 2.4:Secure Decentralized Erasure Code 的架構 15
圖 3.1:系統設定流程..... 20
圖 3.2:使用者清單..... 20
圖 3.3:金鑰伺服器清單..... 22
圖 3.4:儲存伺服器清單..... 22
圖 3.5:系統上傳流程..... 23
圖 3.6:檔案清單..... 24
圖 3.7:系統下載流程..... 25
圖 4.1:手機資源列表..... 28
圖 4.2:JPBC 原本 toBytes 的函式..... 30
圖 4.3:JPBC 修改過 toBytes 的函式..... 30
圖 4.4:JPBC 原本 Jacobi 的函式..... 31
圖 4.5:JPBC 修改過 Jacobi 的函式..... 32
圖 4.6:系統下載實際操作圖..... 32
圖 4.7:系統上傳實際操作圖..... 32
圖 4.8:手機 CPU 資源比例分配圖..... 33
圖 4.9:手機記憶體資源比例分配圖..... 34

表目錄

表 4.1:橢圓曲線運算分析..... 35
表 4.2:橢圓曲線跟 RSA 的安全強度分析..... 35

第一章、 引言

1.1 介紹

雲端計算近年來成為大家討論的話題，而且逐漸變成下一代人們使用電腦的架構，雲端計算基本上含有一個以上大型的數據中心，裡面包含了大規模的軟體以及大量的資料庫，用戶們只需要一個可以連到網路的可攜式硬體，即可以透過網路享有高品質的服務(如圖 1.1 所示)，而這種獨特模式打破傳統的儲存系統均存放在自己的硬碟裡，隨著網路速度的增加以及現代人需求愈來愈多樣化的情況下，儲存技術廠商紛紛推出雲端儲存的想法，加上現在的智慧型行動裝置愈來愈普及的狀況下，許多行動裝置業者紛紛加入雲端服務的功能，讓使用者能隨意地欣賞到最熱門的音樂、視訊、數位課程、新聞、遊戲、電子書等服務，不再像是以往的儲存系統只能在限定的機器上才能享受到應有的服務。

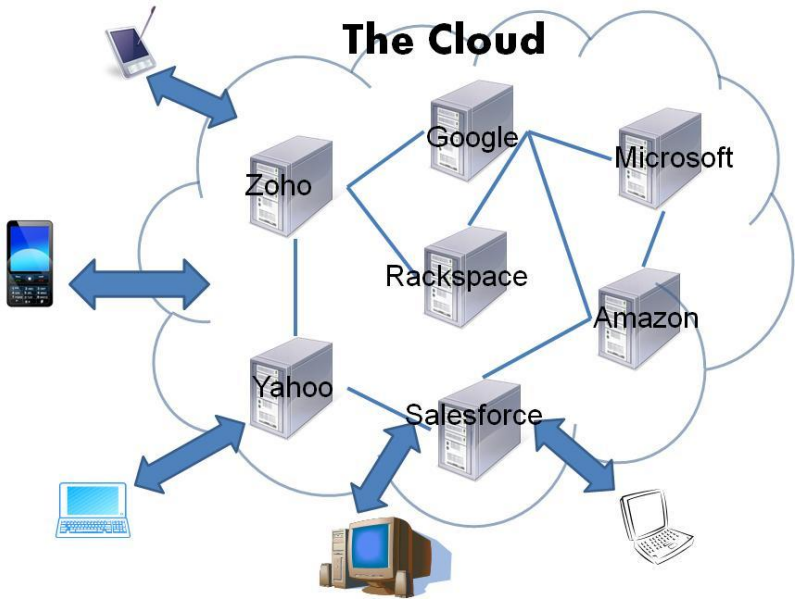

圖 1.1

1.2 雲端計算的發展

雲端這個概念在 2007 年 10 月 IBM 與 Google 在美國校園裡展開，之所以會推廣雲端計畫，最主要的是希望能降低分散式運算在學術研究計畫裡所花費的巨大資本，所以 IBM 及 Google 提供了相關的軟硬體支援，讓每個學生都可以透過網路進行大規模運算為基礎的學術研究。此外 2008 年 1 月 Google 也宣布在台灣的雲端計畫，並且跟台灣的台大、交大等學校合作，把這個先進的雲端概念迅速的吹進了校園裡頭。

1.3 雲端所面臨的問題

但是雲端的概念充滿了許多問題，包括安全、管理以及效率等方

面逐漸浮出檯面，尤其最近幾年環保意識高漲，許多電器都需加上節能標章，所以在雲端的概念裡不斷加入了節能的方法，讓每台機器都能充分的運用到。不過在節省能源的同時也突顯了一些管理上以及安全上的問題，例如:當一台沒有用到機器如何關掉其電源，而又或者忽然間大量服務湧現時，如何讓電源迅速開啟，陸陸續續許多人在管理這方面的研究，此外管理方面還會出現安全上的問題，例如當雲端計算的使用者逐漸減少，為了要節省能源將某些沒用到的電腦逐一關閉，然而在關閉的電腦裡或許還有一些使用者正在運算，那如何讓正在運算的電腦移到其另一台電腦裡，或許是管理上的問題，但是在移後不會讓目前的使用者不被移到這台的使用者所存取或是相反過來，都是我們不願意看到的結果，所以近年來雲端發展計畫在安全性這方面逐漸受到重視。

1.4 Android 的發展

隨著智慧型手機的發展以及市場需求，Google 在 2007 年推出 Android 手機作業系統，並結合各家手機廠商的技術，讓 android 手機系統能更適合 Google 的次世代網路服務以及開放原始碼的策略，讓開發者可以自由地修改以符合使用者的需求。在 Android 的發展，Google 除了提供一個 Open Source 的環境讓廠商可以節省平台授權費，

不必被層層剝削，且開發者也能在開發應用程式上獲取更多的自由，另外還有一個重要的因素：使開發應用程式可以在不同平台上通用，不必再花費時間及精力將應用程式作跨平台的改寫。Android 系統底層以 Open Source 中最著名的 Linux Kernel 為基礎,以提供基本的運作功能。最有附加價值的應用軟體部份則開發者可以利用 Java 作為編寫程式來進行開發。因此我們選擇了 android 及跨平台的 java 當我們架構的主要軟體。

1.5 問題陳述

近年來智慧型手機愈來愈普及且在服務上更是推陳出新，許多手機現在都可以支援 3G 及 Wi-Fi 上網，使得任何人都可以在這開放的 internet 中隨意的把資訊儲存在雲端系統裡，但卻沒有對資料做任何加密隱私性的問題，為雲端安全問題一大主因。

在這種雲端儲存的架構下如何確保個人資料不被人修改，即使在修改幅度微小的情況下，我們也有辦法將資料還原回來，最簡單的方法即是備份至每台 storage 上，不過這樣不單是會浪費儲存空間，並且是很沒效率的一個做法。

1.5.1 加入 Erasure code 機制

我們在設計的架構裡加入了 erasure code 的概念，目的是為了減

少每台伺服器上的儲存成本。特別是我們的做法讓即便是全部的 storage server 都被 compromise，也無法在有限時間內將我們的資料解密出來，且我們提出了 key server 概念來存放我們的 secret key，但我們假設 key server 比 storage server 更不容易被 compromise，因為如此重要的訊息，我們會用更加安全的方式加以保護，而且我們需要收集到一定數量 key server 的資訊，我們才有辦法將訊息解密回來，換句話說，攻擊者需要同時 compromise 一定數量的 key server 會具有相當困難的難度。

1.5.2 管理機制

除此之外我們還加入 control server 來管理使用者存取的權限以及儲存伺服器的數量，另外還幫忙使用者將一份資料傳給數個 storage server，因為我們假設 client 端的網路頻寬是相當有限的，不像是 server 端的網路那麼好，因此我們 control server 將幫我們決定傳送哪幾個 storage server，我們這樣的方法不僅僅只有節省了用戶傳輸的流量更是節省遠端儲存伺服器的儲存成本，另外安全方面我們更是確保了資料的完整性及隱私的問題。

第二章、背景知識

這章主要介紹本篇如何做加密及解密的方法，主要參考了 Hsiao-Ying Lin and Wen-Guey Tzeng[9]的 Protocol。另外一方面，因為 Protocol 會利用到橢圓曲線密碼學的相關理論，因此接下來我們會在 2.2 節及 2.3 節做逐一的介紹。

2.1 儲存系統

OceanStore 在 2000 年就提出了分散式儲存系統的構想，目的是希望提供行動用戶們可以隨時隨地透過行動裝置去存取 OceanStore 中的資料。不過它不像是傳統的系統一樣將資料存放在固定的伺服器中，且每台伺服器都有防火牆保護。雲端的概念使得使用者能透過網路在任意時間任意地點，以及任意設備上獲取資料，而且會依伺服器不同的需要在任意時間點將資料搬動到其他伺服器上。假設資料是以明文儲存的話，這樣安全性上會有很大問題，另外如果伺服器上的資料有發生損毀的問題，那麼使用者下載完成的檔案也會發生問題。所以目前的儲存伺服器都必須要具有檔案的完整性及保密性的功能。

儲存系統的目的主要是為了長期保存資料，會有系統資料損毀或是遭人惡意串改的情況，儲存上如有這些情況都是長期保存資料時所不允許的，早期的方式是將一份資料複製好幾份存到每台 storage server，或是一台 storage server ，會有一台相對應的 mirror storage server(簡

稱 MSS)，當資料毀損或被串改時，就可以從 MSS 上還原回來，蛋是以上種種方法的 overhead 都太高，所以 Erasure code 是較早提出的方法中具體達到容錯以及低 overhead 的方法。在 2006 年 Decentralized Erasure code [12]被提出，並用在分散式網路儲存系統的架構中，緊接著在 2010 年 Secure Decentralized Erasure code [9]被提出，用以加強系統中資料的保密性。

2.2 橢圓曲線密碼學

橢圓曲線密碼系統近年來逐漸被廣泛的使用，主要是系統可以使用較短的金鑰長度，達到與 RSA 密碼系統相同等級的安全強度，讓擁有相同安全強度的橢圓曲線密碼系統較 RSA 密碼系統其計算複雜度相對來的小。橢圓曲線的相關研究已經有一段時間，在西元 1985 年，由 Victor Miller 和 Neal Koblitz 兩人提出橢圓曲線的公開金鑰密碼系統，而橢圓曲線要使用在密碼系統上，須利用基礎於有限場(finite field)之上，其通用格是為 $y^2+axy+by=x^3+cx^2+dx+e$，也稱之為 Weierstrass 方程式(generalized Weierstrass equation)，橢圓曲線之所以可以用在密碼系統上，主要是基於橢圓曲線密碼上離散對數問題(Elliptic Curve Discrete Logarithm Problem,ECDLP)，因此我們將收集滿足此方程式的所有點及一個無窮遠點(point at infinity)的集合所形成的一個加法群(additive group)。

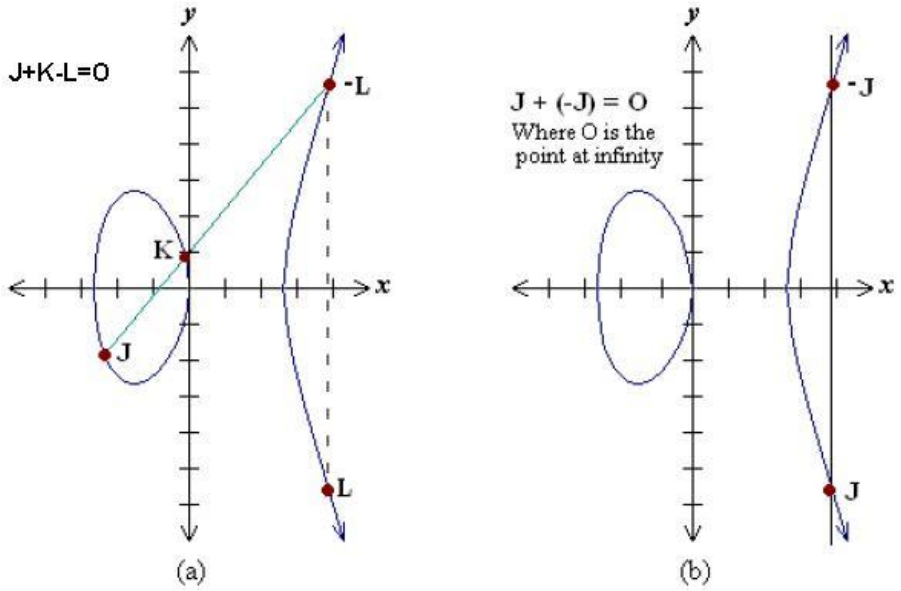

圖 2.1 [11]

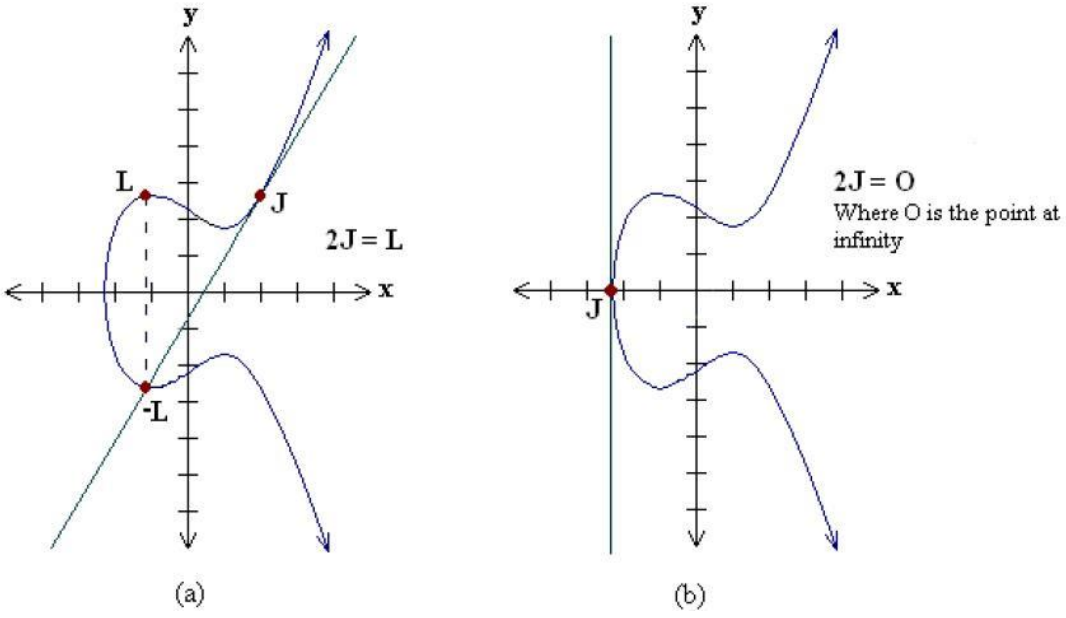

圖 2.2 [11]

橢圓曲線上的點能夠進行點跟點之間的加法，以幾何觀點來看，一條直線交於橢圓曲線上至多三點，所以有以下四種情況:

1. 如果三點都位於橢圓曲線跟直線的交點上，則這三點的總合為 O，其中 O 我們定義為無窮遠點(point at infinity)。

2. 當橢圓曲線上的一個點 J，要找出-J 的點，我們需將此點 J 對 x 軸做鏡射，即可得到我們所要的-J 點，而其中我們的 J + -J = O。

3. 當一條直線交於橢圓曲線上的兩個點分別為 J、-L 兩點，則 J + J = 2J = L

4. 如果一條直線跟橢圓曲線交於一個點 J，則我們定義 J + J = O，而 O 為無窮遠點(point at infinity)。

假設有一個質數 p>3，在有限體大小為 p 之橢圓曲線上我們可以記為 GF(p)，其方程式為 $y^2 \equiv x^3 + ax + b \pmod{p}$ ，其中 $a,b \in GF(p)$且$4a^3 + 27b^2 \neq 0$，則滿足此方程式的解，存在一群 P = (x_1 , y_1)的點且 $x, y \in GF(p)$ 且都滿足此曲線方程式，加上無窮遠點(point at infinity)，可形成交換群(abelian group)。根據橢圓曲線上的點運算，可以定義出點運算的公式如下:

首先有兩點 P、Q 欲相加，其中 $P = (x_1, y_1)$且$Q = (x_1, y_1)$，我們會先算出通過 P、Q 兩點直線的斜率 λ ，則 $P + Q = (x_3, y_3)$ 為:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$ ，其中: $$\begin{cases} P \neq Q，\lambda = \dfrac{y_2 - y_1}{x_2 - x_1} \\ P = Q, \lambda = \dfrac{2x_1^2 + a}{2y_p} \end{cases}$$

另外在群中的任意點 P 也有一個唯一反元素-P，使得 P + (-P) =O，而反元素記為 $P = (x_1, -y_1)$，當 k 為一個正整數且 $P \in GF(p)$，則

$kP = P + P + \cdots + P$，其中有 k-1 次的加法。

另外一種橢圓曲線是基礎於大小為 2^m 的有限體上，我們可以記做為 GF(2^m)，其方程式寫做 $y^2 + xy = x^3 + ax^2 + b$，其中 $a, b \in GF(2^m)$且$b \neq 0$。則存在一群點 $P = (x_1, y_1)$ 且 $x, y \in GF(2^m)$ 且都滿足此曲線方程式，加上無窮遠點(point at infinity)，可形成交換群(abelian group)。根據橢圓曲線上的點運算，可以定義出點運算的公式如下:

首先有兩點 P、Q 欲相加，其中 $P = (x_1, y_1)$且$Q = (x_1, y_1)$，我們會先算出通過 P、Q 兩點直線的斜率 λ，則 $P + Q = (x_3, y_3)$ 為

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$

，其中:

$$\begin{cases} P \neq Q, \lambda = \dfrac{y_2 + y_1}{x_2 + x_1} \\ P = Q, \lambda = x_1 + \dfrac{y_1}{x_1} \end{cases}$$

在 GF(2^m)群中的任意點 P 也有一個唯一反元素-P，使得 P + (-P) =O，而反元素記為 $P = (x_1, x_1 + y_1)$，當 k 為一個正整數且 $P \in GF(p)$，則 $kP = P + P + \cdots + P$，其中 P 自己加 k 次。

不管是基於哪種任何的有限體，在運算時(反元素、加法、減法、乘法以及除法計算)，都需要在相關的有限體下進行運算。另外在橢圓曲線中只有加法的運算並不存在乘法的運算，所以在乘法群中的指數次方運算相對應到橢圓曲線加法群中的純量乘法(scalar multiplication)

運算，當 k 為一個正整數且 $P \in GF(p)$，若 $kP = O$，則 k 稱之為 P 的級數(order)，通常我們會找一個級數大於 2^{160} 的基點，因為安全強度上相對於 RSA 密碼系統的 1024 位元金鑰，所以我們可以利用純量乘法(scalar multiplication)的運算來架構橢圓曲密碼系統的基礎。

2.2.1 Pairing 的背景基礎[13]

Pairing 為一線性映射函數，通常都是由一個群對應到另一個群，而定義域上的群 G_1 通常為定義在一個質數很大的加法群上，而群上的所有點都符合橢圓曲線方程式，另外對應域 G_2 通常是定義在一個質數很大的乘法群上，我們假設在這 G_1 和 G_2 的離散對數問題是非常困難的。至於 pairing 是如何運算以及演算法在此我們不多加以介紹，有興趣可以參考 Elliptic Curves Number Theory and Cryptography [13]，而我們的實作只需要利用 pairing 的特性，來達到我們所要的加解密系統，所以首先我們先定義 pairing 的運算為 $\tilde{e}: G_1 \times G_1 \to G_2$，假設我們生成三個元素 P、Q 和 R，且 $P, Q, R \in G_1$，而這個 pairing 需要滿足以下三種特性:

1. 可計算性(computability) :對所有屬於 G_1 上的兩點 P、Q，其 $\tilde{e}(P^a, Q^b) = \tilde{e}(P, Q)^{ab}$ 計算是可以在多項式(Polynomial Time)時間內完成。
2. 雙線性(Bilinearity):對於所有屬於 G_1 上的三點 P、Q、R，使得

$\tilde{e}(P+R,Q)=\tilde{e}(P,Q)\tilde{e}(R,Q)$且$\tilde{e}(P,Q+R)=\tilde{e}(P,Q)\tilde{e}(P,R)$

3. 不退化性(Non-degeneracy): 對於所有屬於 G_1 上的點存在 P、Q 兩點，使得$\tilde{e}(P,Q)\neq 1$。

因為在乘法群中的指數次方運算可以相對應到橢圓曲線加法群中的純量乘法(scalar multiplication)運算，所以我們也可以將 Diffie-Hellman 上的問題對應到橢圓曲線上，我們稱這個問題為雙線性 Diffie-Hellman 問題，簡稱為 BDHP，給定 P,aP,bP,cP，任意從 Z_p^* 中挑選出 a、b和c，對任何一個機率的多項式時間演算法(probabilistic polynomial time algorithm) A，要計算出 $d\in G_2$，使得 $d=\tilde{e}(P,P)^{abc}\in G_2$ 的機率是可忽略的(negligible)， 所以我們可以將式子寫成:

$$\Pr[A(P,aP,bP,cP)=\tilde{e}(P,P)^{abc}:x,y,z\in_R Z_P^*]$$

2.2.2 PBC 上的 pairing 介紹

PBC (Pairing-Based Cryptography)的函式庫是一個免費 C 的函式庫(released under the GNU Lesser General Public License)[7]，由史丹佛大學所開發，它提供了七種不同型態的 pairing 供大家使用。其中包括有 Type A、B、C …、G。而我們的系統是則是 Type A，因此以下我們將介紹 Type A 的 pairing 型態。

Type A:

Type A 的型態建構在橢圓曲線方程式為 $y^2=x^3+x$ 的場(field)

上，而G_1和G_2的兩個加法群上的點都在$E(F_q)$，所以我們稱這種 pairing 具有對稱性(symmetric)。因此$\#E(F_q)=q+1$且 $\#E(F_{q^2})=(q+1)^2$。Embedding degree k 是 2，因此G_T是F_{q^2}的一個子群。且級數(order) r 為一個質數也是$q+1$的一個因數。

設定$q+1=r\cdot h$，且為了提高效率，刻意讓 r 是一個 Solinas 質數，r 必須符合2^a+-2^b+-1，其中$a>b>0$，且這裡的 r 就是我們G_1、G_2以及G_T以的級數(order)。[7]

以下是實作上系統參數設定的一個範例：

```
type a
q 3505822541250298138174949230025263325159
r 576601489791778817
h 6080148253720804294 80
expl 47
exp2 59
sign0 1
sign1 1
```

圖 2.3

2.3 Secure Decentralized Erasure Code

本節介紹我們實作基礎的主要協定，出至 2010 年的 Secure Decentralized Erasure code[9]，接下來我們會用簡短地描述如何用 Erasure Code 來進行資料的儲存和取回。首先我們會對資料進行切割，切割的大小取至於我們所選擇橢圓曲線中級數(order)的 bits 長度，因為我們實作中選擇的橢圓曲線級數為 106 bits，因此我們每一個區塊

(block)都為 106 bits，假設我們現在的資料切割成 K 個區塊，而每一個區塊我們都會先將自己所選擇檔案的名字對應到橢圓曲線上的一個點 h，然後由我們自己所選擇的一個數 r 做純量乘法的運算，其結果為 h'，然後再跟系統所選擇的公鑰(public key)一起做 paring 的運算，將運算過後的結果分散傳給 v 個儲存伺服器(storage server)，當一個儲存伺服器收到兩個同樣檔名而不同區塊的加密，將可以再進行線性組合(linear combination)的加密程序，讓多個區塊組合成一個區塊。當使用者要取回檔案時，對 m 個金鑰伺服器(key server)發出下載請求，而每一個金鑰伺服器又會對 u 個儲存伺服器發出下載檔案的請求，而金鑰伺服器首先會跟其他的金鑰伺服器做金鑰的交換，當收集到一定數量的金鑰後，就可以解密出共同的私鑰(secret key)，當金鑰伺服器從儲存伺服器下載的檔案可以透過私鑰進行部分解密，而每一個部分解密的資料都是一個線性組合的方程式，所以當使用者收到 k 個線性獨立(Linear independence)的方程式，將可以把我們的資料解密出來。而以上所敘述的架構(見圖)，大致上分為四大部分，詳述如下:

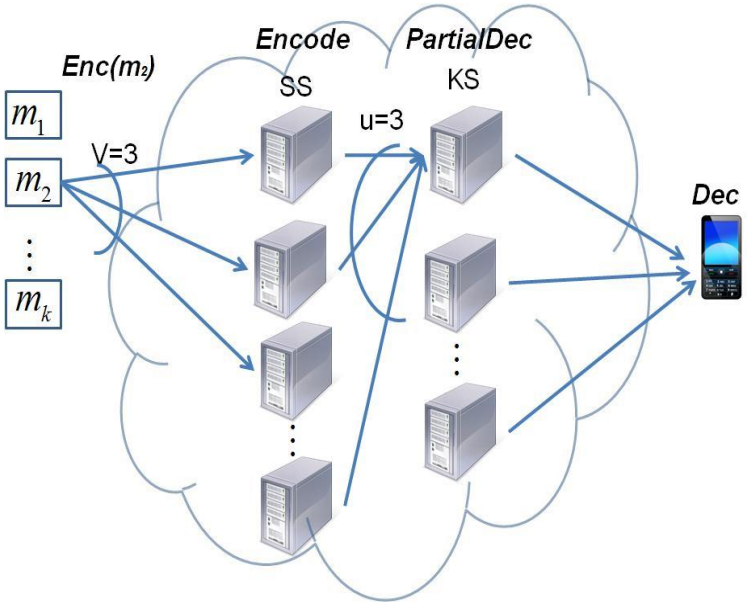

圖 2.4[9]

系統設定:

在還沒進入加解密的過程中，整個系統在運行時，就會先執行一些系統參數，而這些系統參數是這環境中所有角色都知道的，首先會挑一種合適類型的橢圓曲線方程式，且橢圓曲線方程式中的一些參數設定也是公開的，因此可以知道的是整個橢圓曲線的加法群 G_1 以及是基礎於大小為 q 的有限體之橢圓曲線 GF(q)，且會挑一個較為合適的 pairing 運算以及系統所產生出的生成元素(generator) g。因此整個系統輸出的系統參數為 $g, \tilde{e}, G_1$。

使用者設定:

當使用者擁有了系統參數 $g, \tilde{e}, G_1$ 後，可以產生一對金鑰，分別為公鑰(Public key)及密鑰(Secret key)，而密鑰裡包含了一個使用者所選

的參數 x，則公鑰裡包含了$(\tilde{e}, g, g^x, q)$，接著使用者選定一個門檻(Threshold)做密鑰共享(Secret key share)，其中我們的多項式$f(i) = sk_i$是當門檻選定完後所產生的，至於多項式的最高次方是取決於使用者選定的門檻大小所決定的，因此我們可以將多項式寫做為$f(z) = x + a_1 z + a_2 z^2 + \cdots + a_{t-1} z^{t-1} (\mathrm{mod}\ p)$且$a_1, a_2 \cdots, a_{t-1} \in_R Z_P$，其中 t 為門檻的大小，一旦超過這個門檻就可以將 x 取回，而 p 為橢圓曲線中加法群的級數(order)。所以將透過多項式所產生出來的部分密鑰，然後再將檔名所對應的點做部分密鑰的指數次方，接著逐一的分配給金鑰伺服器。

2.3.1 Enc(m_i)

在加密過程中，主要分成兩大部分，第一部分首先會將一個檔案切割成 k 個大小相等的區塊，而區塊大小在於G_T乘法群的級數(order)大小，其中G_T為$\tilde{e}: G_1 \times G_1 \to G_T$，接著將每個區塊的資訊對應到$G_T$乘法群上的一個生成元素 m'。第二部分的第一個步驟，先將要加密檔案名字對應到橢圓曲線上加法群G_1中的一個點 h，然後使用者隨機挑一個參數 r 對 h 做指數次方的運算產生 h^r。接著第二步驟將 h^r跟公鑰中g^x的做 pairing 運算，而所得到的結果 e，然後將第一部分的 m'以及第二部分的 e 做乘法群中乘法運算得到的結果為$C_i, i \in k$。最後

再將系統的生成元素 g 以及使用者所挑的參數 r 做 g^r 運算，將運算出來的 g^r 、h、C_i 都傳至遠端儲存伺服器上。

2.3.2 Encode

我們的系統架構符合 Homomorphism 的特性，使得 $E\ n(m_1)^{a_1} \cdot E\ n(m_2)^{a_2}$ 等於 $Enc(m_1^{a_1} \cdot m_2^{a_2})$，儲存伺服器先將收到的密文跟伺服器自己選的係數做指數次方的運算，當同一位使用者把檔案分割成 k 個區塊時，我們的伺服器也選取 k 個係數，以對應每一個區塊所組成的方程式。

2.3.3 PartialDec

主要是透過 key server 將資料做 Partial Decode 的動作，讓有心人即使破解了少部分的 key server 也無法將資料萃取出來，因為這邊我們用了 Threshold 的 SK 以及線性組合這兩道關卡保護資料。

首先先介紹如何將 Threshold 的 SK (secret key)解密回來，Client 會先對所有的 key server 發出 download 的 request，接著每個 key server 會對 u 個 storage server 下載檔案，當我們收集到足夠 key server 我們就可以將 SK_i 解密回 SK，其中 SK_i 為 f(i)= SK +b_{1i}+b_{2i}^2+...+b_{ni}^{t-1}，透過 Lagrange interpolation 運算解回 SK 。

如何拿到一個線性組合的方程式，當我們拿到了真正的 SK 時，我們可以利用 paring 的特性 $e(g^x, h^r)$ 會等於 $e(g^r, h^x)$，其中 h^r 為 SK、g^x 為 PK，而 g^r 是我們之前存在 storage server 的一個值，所以我們可以利用 $\frac{M \cdot e(g^x, h^r)}{e(g^r, h^x)}$ 將 M 順利解回，M 為 $m_1^{a_1} \cdot m_2^{a_2}$ 的線性組合。

2.3.4 Dec

當我們拿到了 M_1、M_2、…、M_k 時，我們可以利用線性方程組的關係來解聯立方程解，因為我們有係數跟每個方程式的值，反推回去就可以得到我們要的 m_1、m_2、…、m_k。換句話說也就是我們要的檔案內容。

第三章、系統架構

本章節將描述系統的整個架構及角色，以下將會簡短的描述各小節的大綱，而此章節相較於[9]的角色中，多增加了一個控制伺服器角色，原因之一是因為假設手機的網路頻寬不是那麼的充裕，因此藉由控制伺服器代為傳送資料給多個儲存伺服器，這樣只需傳送一份資料的大小到伺服器端，減少了頻寬的使用，前提假設是控制伺服器的網路頻寬遠遠大於手機網路的頻寬，另外一方面是因為多增加了一個控制伺服器較容易方便管理伺服器和使用者。首先 3.1 小節將會詳細的描述在系統建置過程中需要的步驟及流程。3.2 小節描述使用者在進行下載、上傳操作時會遇到哪些步驟。3.3 小節則是在說明服務型伺服器端(儲存伺服器及金鑰伺服器)如何儲存資料。

3.1 系統建置

在系統還沒運行之前，所有的系統都必須在相同的橢圓曲線方程式下做相同型態的 pairing 運算，因此所有系統的橢圓曲線的參數是事先寫好在程式裡頭且不能任意得更改，在系統建置時會產生出在系統建置過程中主要分為三大部分:

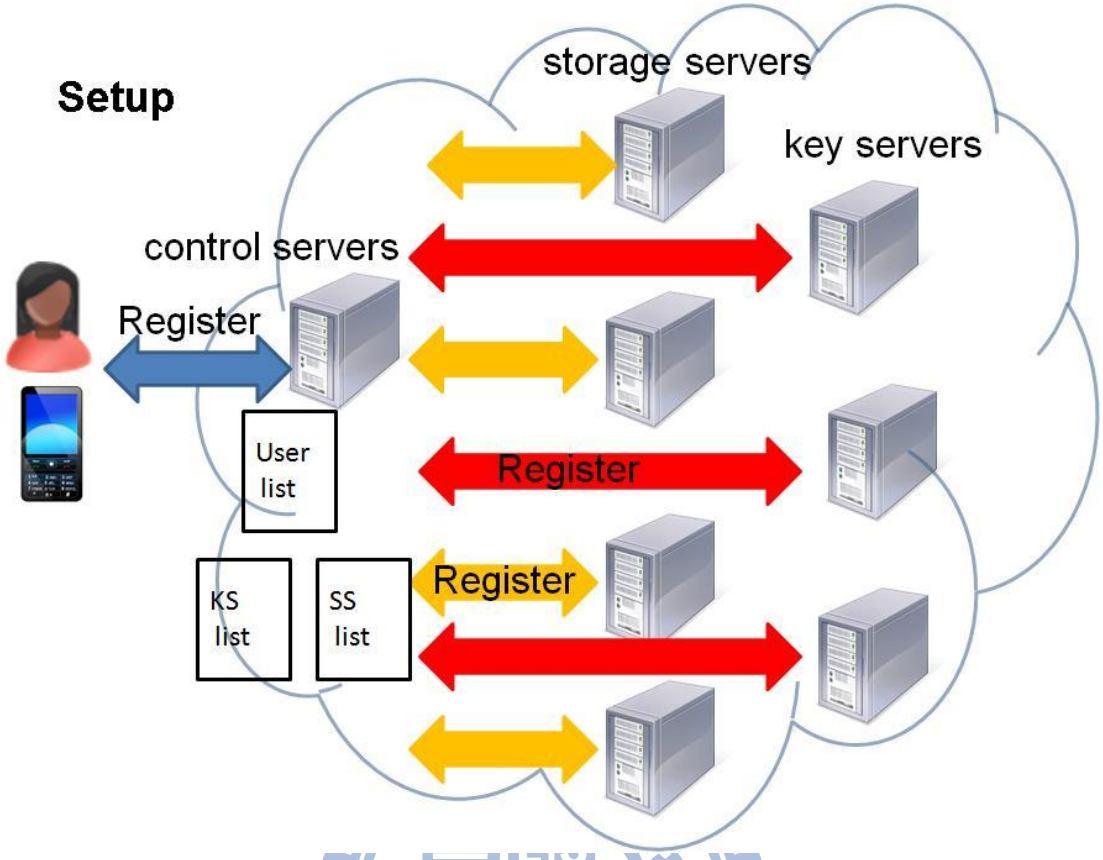

圖 3.1

第一部分主要為客戶(client)端，例如: android 智慧型手機，首先使用者必須跟我們的控制伺服器註冊使用者姓名及密碼，如果註冊成功的話，則控制伺服器上的使用者清單會有我們剛註冊的使用者姓名及密碼，如圖 3.2。

	name	password
	woei	*1A9E6D33611D2A9D632D1B983AC95C7B65EFB4D0
	ccis	*CF0231CFDD492D4EFF4FF988510F9A4CE7621FFE
	Kobe Bryant	*7EE94F15CF66B748370A65103767A4920A66232F

使用者清單(圖 3.2)

第二部分為控制端伺服器，其目的是為了監看使用者和伺服器數量並代替客戶端(智慧手機)將加密後的資料分別傳送給不同的儲存伺

服器(storage server)，所以必須要先啟動控制端伺服器讓其他服務型伺服器啟動後可以向控制端伺服器登記，如此一來才有辦法快速的掌握伺服器的數量。

第三部分是服務型伺服器端，其中包含儲存伺服器及金鑰伺服器(key server)，當一個儲存伺服器被架起來的時候，會發出一個訊息給控制伺服器進行註冊，如果註冊成功的話，可以透過瀏覽器查詢目前有哪些儲存伺服器提供服務的，儲存伺服器清單裡包含的內容有伺服器的名字、IP 地址以及 Port，如圖 3.4。當儲存伺服器關閉時會向控制伺服器發出一個關閉的訊號，使得控制伺服器能從儲存伺服器清單裡做修改，萬一儲存伺服器因為某些不確定的因素而導致不正常關閉，則當控制伺服器在送資料時給儲存伺服器時，會發現對方無回應或者有例外產生，則控制伺服器會將沒回應或者發生 socket 上例外的儲存伺服器從儲存伺服器清單中清除。而當一個金鑰伺服器啟動完成的時候，會發出一個註冊訊息給控制伺服器登記在金鑰伺服器清單上，然後接著在瀏覽網頁時除了會跟儲存伺服器清單看得到 name、port、IP 外，還提供一項 information 讓使用者能多點資訊知道伺服器有哪些訊息，如圖 3.3。最後在系統建置完畢後會輸出的系統參數為 $g, \tilde{e}, G_1$。

←T→	name	IP	port	info
	woei-server	140.113.214.159	6000	
	right-server	140.113.214.166	6001	
	ccis-hadoop	140.113.207.134	6002	core i7
	ccis-server	140.113.207.136	6003	
	ccis-desktop	140.113.207.139	6004	

金鑰伺服器清單(圖 3.3)

←T→	name	IP	port
	ccis-desktop	140.113.207.139	3004
	ccis-server	140.113.207.136	3003
	ccis-hadoop	140.113.207.134	3002
	right-server	140.113.214.166	3001
	woei-server	140.113.214.159	3000

儲存伺服器(圖 3.4)

a. 使用者登錄

因為每個使用者都是個別獨立的個體，所以兩兩使用者之間不能互相存取他人的檔案，因此必須妥善的管理每一位使用者權限，讓使用者不能任意的下載以及修改他人檔案及資料。第一次進入系統的每位使用者都必須先跟控制伺服器註冊一個帳號及密碼，註冊成功後，可以利用瀏覽器來登入個人帳號及密碼，而系統會依據帳號的權限而給予不同檔案及資料。另外一方面可以透過瀏覽器來查詢目前使用者有哪些檔案已經在遠端的儲存伺服器上並可監看有幾台服務型伺服器已經在提供服務。

3.2 使用者操作

使用者在進行操作時有上傳及下載兩項功能。

a.上傳

在這主要分成兩部分來探討，第一部分首先 android 手機會產生一對金鑰 PK, SK 跟一個生成元素 g (generator)以及系統在橢圓曲線中所選擇的級數 p ，跟一個系統相同型態且有效的 pairing 運算 $\tilde{e}$ ，緊接著將密鑰(secret key)利用 secret sharing 方式來產生多把不同的部分密鑰，然後將這些部份密鑰依照系統編號陸續的存放在相對應的金鑰伺服器上。

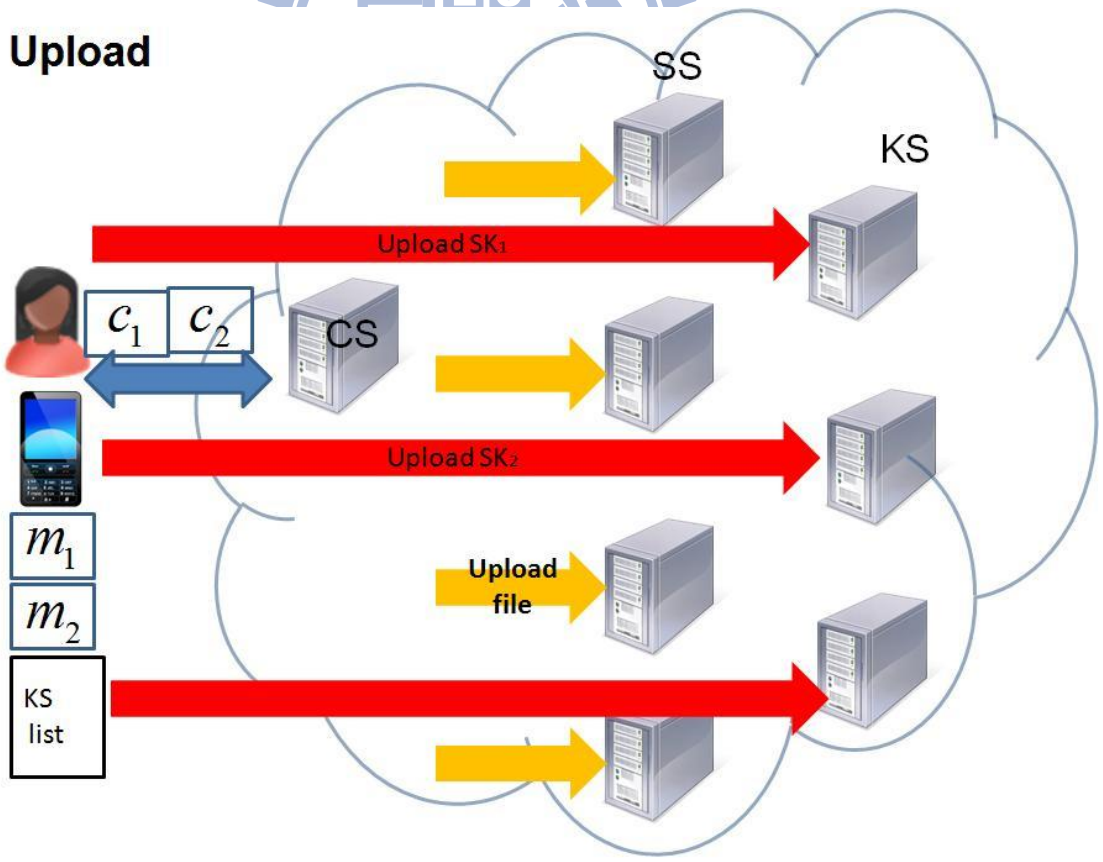

圖 3.5

第二部分，在上傳之前 android 手機會將一個檔案切割成 k 個大小相等的區塊，而區塊大小在於 G_T 乘法群的級數(order)大小，其中 G_T 為 $\tilde{e}: G_1 \times G_1 \to G_T$，緊接著將每個區塊的資訊對應到 G_T 乘法群上的一個生成元素 m'。另外一方面將要加密檔案名字透過一個哈希函數(hash function)對應到橢圓曲線上加法群 G_1 中的一個點 h，然後使用者隨機挑一個參數 r 對 h 做指數次方的運算產生 h^r。接著將 h^r 跟公鑰中 g^x 做 pairing 運算，而所得到的結果 e 跟 m'做乘法群中乘法運算得到的結果為 $C_i, i \in k$。最後再將系統的生成元素 g 以及使用者所挑的參數 r 做 g^r 運算。

在上傳時先用使用者的密碼對(檔名||h)做加密得到 F'，然後將使用者帳號及 F' 上傳控制伺服器，給當控制伺服器收到使用者帳號後，先從使用者清單中將其對應的密碼對 F' 做解密，得到的檔名透過哈希函數算出來的值是否等於 h，如果是的話就會將檔名、h 及使用者都加入檔案清單中，如圖 3.6，接著再將運算出來的 g^r、(h||使用者帳號)、C_i 都傳至遠端儲存伺服器上。第一部分，如果金鑰伺服器上已經有了使用者的部分密鑰後，則不需要再做第一部分的步驟。

	filename	hashvalue	usrname
	LKK.txt	{x=231950660266562954505839984280699274454,y=90692...	woei
	curve.properties	{x=260502799075947749839941441859584260885,y=23057...	woei

檔案清單(圖 3.6)

c.下載

在下載之前，android 手機會先用使用者的密碼對(檔名||h)做加密得到 F'，然後將使用者帳號及 F' 上傳至控制伺服器，給當控制伺服器收到使用者帳號後，先從使用者清單中將其對應的密碼對 F' 做解密，得到的檔名透過哈希函數算出來的值是否等於 h，如果是的話就會將 h、使用者跟儲存伺服器清單傳送給金鑰伺服器，當金鑰伺服器收到儲存伺服器清單後，對部分儲存伺服器送出(h||使用者帳號)，然後儲存伺服器收到(h||使用者帳號)，會從儲存目錄下比對(h||使用者帳號)的檔名，然後接著將其內容傳送給金鑰伺服器，然後金鑰伺服器對其內容做部分解密，將其結果傳送給客戶端，接著客戶端再做解密，得到原本所要的資料。

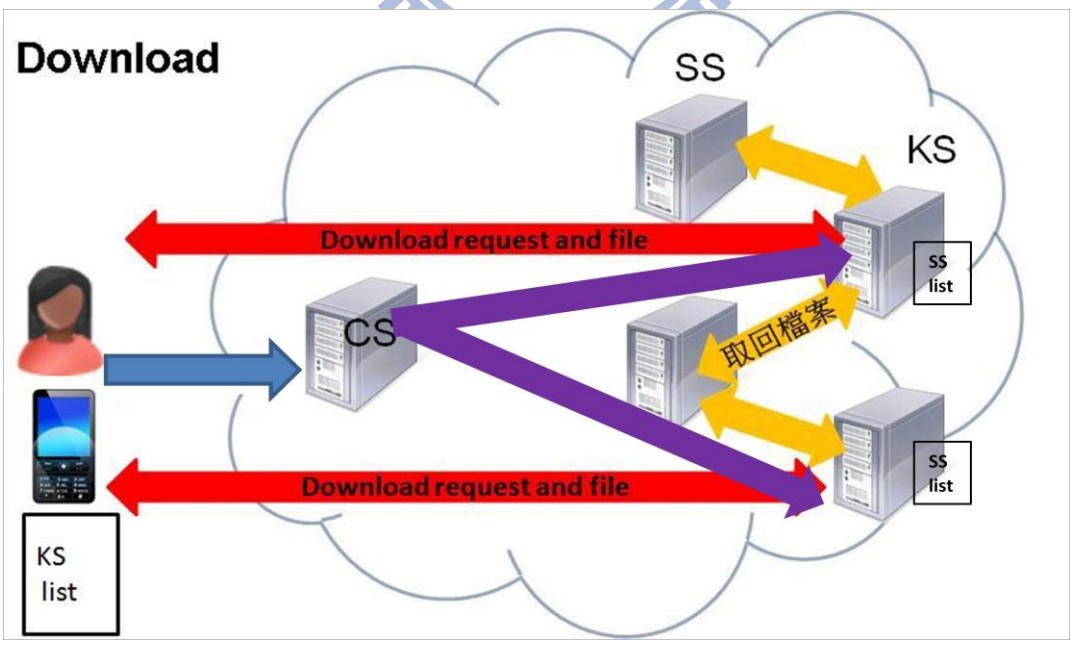

圖 3.7

3.3 服務型伺服器的資料儲存

本節將會描述金鑰伺服器是如何儲存其部分密鑰以及儲存伺服器是如何儲存資料的。

a. 金鑰伺服器的儲存

當客戶端傳送部分密鑰以及使用者帳號時，金鑰伺服器會將使用者帳號當作檔名，然後將部分密鑰的內容寫入檔案中。另外一方面，為了避免有人刻意竊聽客戶端以及金鑰伺服端之間的通訊，來達到收及部分密鑰，因此我們採用了以橢圓曲線為基礎的 Diffie-Hellman 密鑰交換的方法來加密我們的部分密鑰。

b. 儲存伺服器的資料儲存

收到來自於控制伺服器所送的(h||使用者帳號)，我們會將其(h||使用者帳號)當作檔名，然後將儲存伺服器所做的 combine 的結果，還有伺服器自己所挑的係數，都一一的儲存到檔案裡。

第四章、系統實作

在系統實作上，首先需要大量的伺服器，所以我們先將大量的伺服器都實作在同一台機器上面，依據 port 的不同區分不同的伺服器，另外我們會依據不同資料夾來區分不同儲存伺服器的儲存空間，所以即使在同一個機器上，我們仍然有區分出不同的伺服器。首先 4.1 章節我們將介紹系統有哪些設備工具及環境，緊接著 4.2 我們會介紹我們所遇到的困難點。4.3 會接著介紹我們程式在 android 上跑的效能為何。4.4 還會有一個安全強度的分析。

4.1 開發環境建置

在安全的分散式儲存系統的實作，目前我們所有的伺服器包含了控制伺服器、 儲存伺服器及金鑰伺服器都跑在同一台機器上，所使用的 CPU 為 core i7，硬碟容量為 500GB，Operating system(OS)為 Window vista，計畫會將機器逐步的分散出去，客戶端我們使用的手機為 HTC Magic，針對 HTC Magic 其 CPU 為 ARMv6-compatible processor rev 2 ，而頻率為 526.25MHZ，內部記憶體為 192.37MB，如圖 4.1。android SDK 的版本為 1.5，另外我們所使用的開發環境為 java 的 JDK 1.6，開發軟體為 eclipse 3.4.0，並使用套件 Java Pairing-Based Cryptography Library ，簡稱 JPBC，版本為 1.0.0。

圖 4.1[8]

4.1.1 選擇 JPBC 的因素

雖然在 java JDK 1.6 以及 android SDK 1.6 以上的版本都已經有橢圓曲線密碼的 library 可以使用，不過上述的 library 並還沒有實作出 pairing 這個運算，因此我們需要找一個套件是有實作出 pairing 這運算的 library，但在眾多的 pairing base 的 library 裡，例如:JPBC、PBC 以及 Tiny-ECC…等可以挑，而且 JPBC 的 pairing 運算時間並不是裡面中最好的，主要原因是因為首先它是由 java 語言所寫成的，所以實作

上不需要牽扯到底層的 kernel，實作上會相較容易多，另一方面是因為 JPBC 在被包成.apk 檔(android 的安裝執行檔)後，直接安裝.apk 就能執行 pairing 的運算，所以在廣泛發布上相對較好。

4.2 實作上面臨的問題

在實作中所面臨的困難，最主要是 JPBC library 要如何 porting 到我們的 android 手機上，因為 HTC magic 所提供的 CPU 及 RAM 都是相較於慢且不足，實作上需要盡可能的節省手機不必要的資源浪費。

4.2.1 JPBC library

Java Pairing-Based Cryptography Library 發表於 2009 年 10 月是由 Ben Lynn 所寫的 Pairing-Based Cryptography Library(PBC)改成 java 的 porting[6]，它支援 symmetric 和 asymmetric pairing，不過這邊遇到了兩個問題。

因為 JPBC 的 library 是支援 java JDK 1.6，而在 android 上的 java 是由 android SDK 所提供，因為 android SDK 上的 java 是精簡版的 JDK，因此並不是 java 上每個 function call android SDK 都有提供，因此需要先將 JPBC 上使用的 function，而 Android 裡並沒有提供的 function call 加以修改，然後再加以編譯成.class 檔，最後再匯出成.jar 檔。以下是

以 JPBC 中 NaiveElement.java 為例:

jpbc-plaf\src\main\java\it\unisa\dia\gas\plaf\jpbc\field\naive\NaiveElement.java

bytes = Arrays.copyOfRange(bytes,1,length); (Android 並沒有提供)

```
@Override
public byte[] toBytes() {
    byte[] bytes = value.toByteArray();
    if (bytes.length > field.getLengthInBytes()) {
        if (bytes[0] == 0 && bytes.length == field.getLengthInBytes() + 1) {
            // Remove it
            bytes = Arrays.copyOfRange(bytes, 1, bytes.length);
        } else
            throw new IllegalStateException("result has more than FixedLengthInBytes.");
    } else if (bytes.length < field.getLengthInBytes()) {
        byte[] result = new byte[field.getLengthInBytes()];
        System.arraycopy(bytes, 0, result, field.getLengthInBytes() - bytes.length, bytes.length);
        return result;
    }
    return bytes;
}
```

圖 4.2

首先要先將 code 改成呼叫我們自己所寫的 function，另外還要確認我們所寫的函式是否有符合原來函式所要的結果。

```
public static byte[] copyOfRange(byte[] original, int from, int to) {
    int newLength = to - from;
    if (newLength < 0)
        throw new IllegalArgumentException(from + " > " + to);
    byte[] copy = new byte[newLength];
    System.arraycopy(original, from, copy, 0,
                     Math.min(original.length - from, newLength));
    return copy;
}
@Override
public byte[] toBytes() {
    byte[] bytes = value.toByteArray();
    if (bytes.length > field.getLengthInBytes()) {
        if (bytes[0] == 0 && bytes.length == field.getLengthInBytes() + 1) {
            // Remove it
            bytes=copyOfRange(bytes,1,bytes.length);
        } else
            throw new IllegalStateException("result has more than FixedLengthInBytes.");
    } else if (bytes.length < field.getLengthInBytes()) {
        byte[] result = new byte[field.getLengthInBytes()];
        System.arraycopy(bytes, 0, result, field.getLengthInBytes() - bytes.length, bytes.length);
        return result;
    }
    return bytes;
}
```

圖 4.3

另外一個問題是 android 上的 memory 不像個人電腦那樣可以提

供較大的 memory 可以給程式使用，所以有些程式需要用到遞回的 function call 就有可能會發生 stack overflow 的 exception，因為這個的例外並不是一定會發生的，要發現這個現象需要用到 android 的專屬 debug 工具 Dalvik Debug Monitor Server(簡稱 DDMS)，它能告訴你程式在哪個地方出現不尋常的關閉，所以我們要將發生不尋常的地方例如: stack overflow，改成非遞回的方式。下面以 JPBC 中 BigIntegerUtils.java 為例:

jpbc-plaf\src\main\java\it\unisa\dia\gas\plaf\jpbc\util\BigIntegerUtils.java

如圖將遞回的 Jacobi 函式改成非遞回的 Jacobi 函式。

```
public static int jacobi(BigInteger a, BigInteger n) {
    /* Precondition: a, n >= 0; n is odd */
    int ans = 0;

    if (ZERO.equals(a))
        ans = (ONE.equals(n)) ? 1 : 0;
    else if (TWO.equals(a)) {
        BigInteger mod = n.mod(EIGHT);
        if (ONE.equals(mod) || SEVEN.equals(mod))
            ans = 1;
        else if (THREE.equals(mod) || FIVE.equals(mod))
            ans = -1;
    } else if (a.compareTo(n) >= 0)
        ans = jacobi(a.mod(n), n);
    else if (ZERO.equals(a.mod(TWO)))
        ans = jacobi(TWO, n) * jacobi(a.divide(TWO), n);
    else
        ans = (THREE.equals(a.mod(FOUR)) && THREE.equals(n.mod(FOUR))) ? -jacobi(n, a) : jacobi(n, a);

    return ans;
}
```

圖 4.4

```
public static int jacobi(BigInteger a, BigInteger n) {
    BigInteger temp = ZERO;
    BigInteger j = ONE;
    a = a.mod(n);
    if (a.compareTo(ZERO)<0)
       a = a.add(n);
      while (!a.equals(ZERO)) {
        while (a.mod(TWO).equals(ZERO)){
          a = a.divide(TWO);
          if (n.mod(EIGHT).equals(THREE)||n.mod(EIGHT).equals(FIVE))
              j = j.negate();
        }
        temp = a;
        a = n;
        n = temp;
        if (a.mod(FOUR).equals(THREE)&&n.mod(FOUR).equals(THREE)) j = j.negate();
        a = a.mod(n);
      }
      if (n.equals(ONE))
          return (int)j.longValue();
      else
          return 0;

}
```

圖 4.5

最後再把 JPBC 整個 library 用 eclipse 增加 extenal JARS，將 JPBC library link 起來，接著再跟所寫的程式一起包成.apk 檔，有了 APK 檔案後就可以順利的安裝到 android 上。如圖 4.6、4.7

圖 4.6

圖 4.7

4.3 Android 上的效能

將 JPBC 成功的安裝到 Android 上後，我們測試其在跑我們上傳時所做的加密，以及下在時所做的解密，是否會占用 Android 上大部分的資源，由圖可知道我們的程式 HTC magic 上並不會占用大多數 CPU 及 Memory 的資源。如圖 4.8、4.9

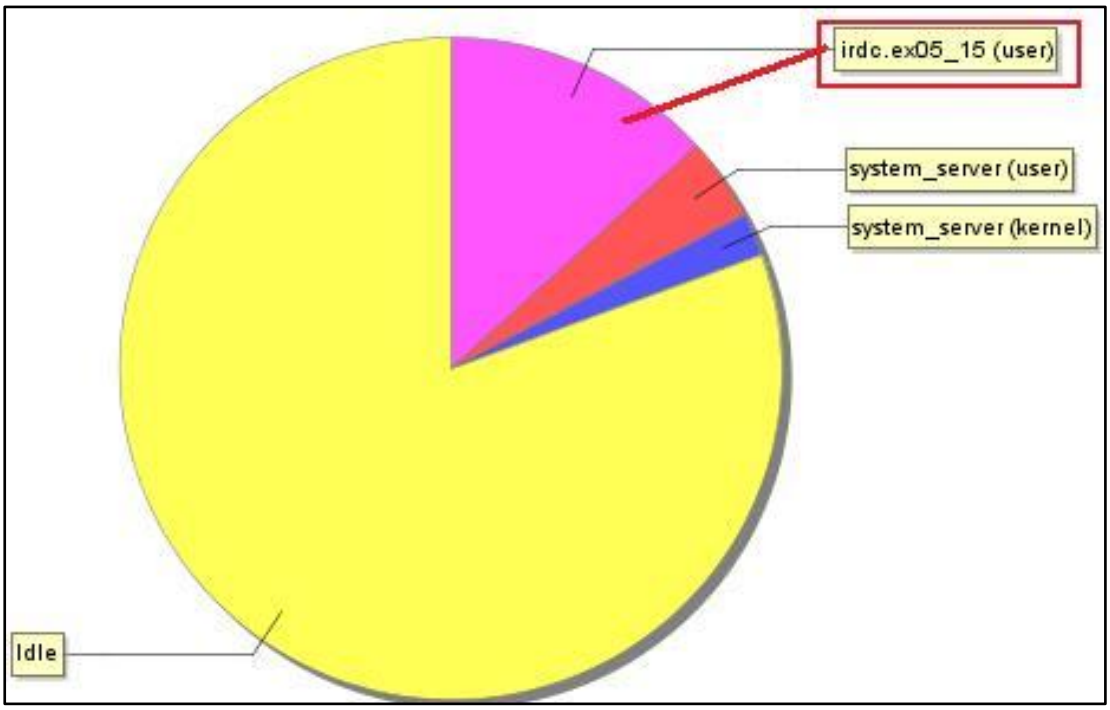

圖 4.8

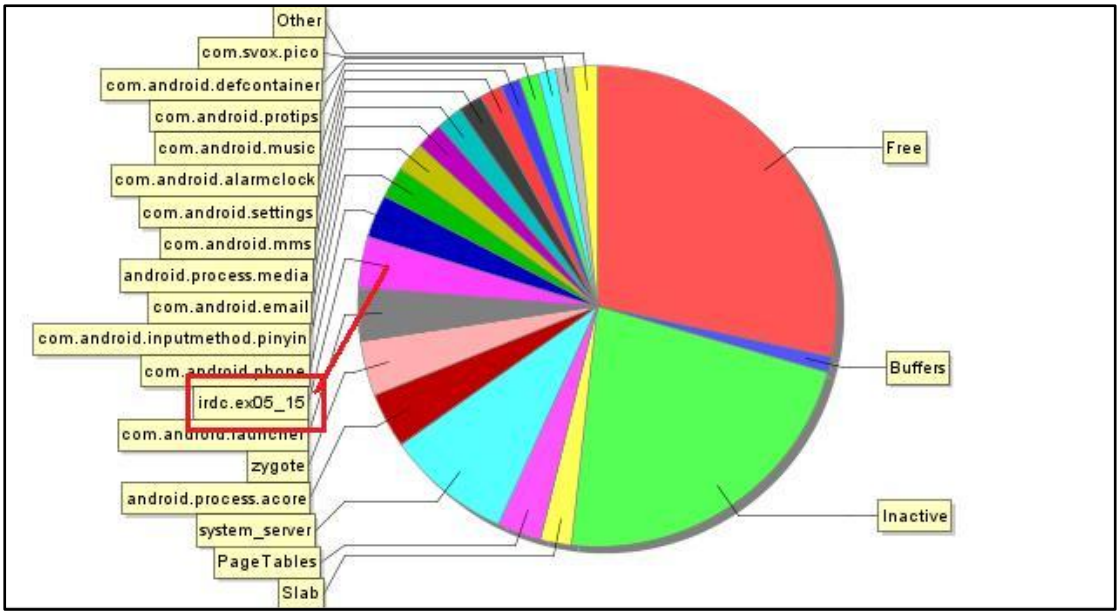

圖 4.9

另外我們針對了橢圓曲線上的加法、純量乘法(MulZn)以及乘法群中的指數次方運算跟乘法(Mul)運算做時間上的分析，如表 4.1，其中我們以級數長度做為比較，而 table 上的值都是以秒為單位，由此可知做 pairing 的運算時間要比指數次方運算時間來的長，所以我們利用 pairing 雙線性的特性，使得 $\tilde{e}(g^x, h^r) = \tilde{e}(g^x, h)^r$ ，因此我們只需要做一次 pairing 的運算，其餘的都做指數次方的運算。假設以 106 bits 為例，有 k 個區塊，則沒利用特性的需要 $1.4*k$ 秒，而有用雙線性的特性，則花費 $1.4+0.5*k$ 秒，所以可以節省 $0.9*k-1.4$ 秒，所以當 k 愈大節省的愈多，不過相對要加密的時間也就會愈高，但是 android 執行的程式可以讓它變成背景執行，而且因為我們的程式占 android 的 CPU 及 Memory 不會太大，所以在上傳時可以將程式 swap 到背景執行，等到程式上傳結束。

Operator/Order bit	60 bits	106 bits
Mulzn	0.3	0.5
Mul	0.01	0.01
Pow	0.3	0.5
Add	0.002	0.002
Pairing	0.8	1.4

表 4.1

4.4 安全度分析

因為 android 軟硬體上的限制，所以我們只能做到級數長度為 106 bits 的安全度，不過以 106 bits 的安全強度等同於 RSA 512bits 的安全強度(參照 www.rsasecurity.com 內的說明)，所以我們的安全還是有一定的強度在。如表 4.2

RSA	512	1024	2048	3072	7680	15360
ECC	106	160	224	256	384	521

表 4.2

第五章、結語

在本篇文章中最主要的重點是在於我們推出了一套既安全且成本低的 Erasure code 在我們的分佈式網路儲存系統。而我們的系統設計，同時提供的儲存服務和金鑰管理服務。

5.1 研究成果

我們的系統即使是在不受信任且分佈網路架構下的儲存伺服器，我們的分佈式網路儲存系統即便是所有的儲存伺服器都受到影響，但我們的系統依舊可以保障個人的隱私，另外我們的儲存系統仍然可以幫助私人資料的長期保存，我們設計一個分佈式網路儲存系統，主要目的是為了行動智慧型裝置的用戶們。它需要進一步減少計算量和傳輸所需的流量。

在實作的過程中，因為我們的整個伺服器系統是用 java 語言所寫出來的，所以我們的伺服器的系統是可以跨平台來使用的，另外方面的貢獻是我們的程式不僅僅可以在 android 手機上做 pairing 運算，甚至考慮到推廣性，因為使用者不需要知道 android 系統內部是怎樣做的，只需要將我們的 APK 檔下載下來就可以直接安裝，而且 APK 檔的好處是我們可以發布在 Google market 或是中華電信的 Hami Apps 上，供大家下載。另外實驗顯示我們的程式不會消耗大部分 CPU 及記憶體的資源，所以我們的程式是可以適用於平常生活使用中。在安

全強度上雖然法達到標準的 RSA 1024 的強度，不過也有達到 RSA 512 的強度。

5.2 未來發展

雖然受限於 HTC-magic 的軟硬體限制，只能做到 106 bits 的長度，可是我們利用模擬器模擬在 android SDK 2.0 的環境下，我們可以將級數(order)長度做的更大，或許 android SDK 2.0 以上的版本可以做到 RSA 1024 的強度。

因為現在我們系統之間的溝通都是透過我們自己訂的協定，所以之後我們可以將伺服器的系統做在 Hadoop 上，透過 Hadoop 的 map-reduce 來計算我們的加解密，或是利用 Hadoop 的檔案系統來管理檔案的儲存。

目前儲存伺服器必須大於或等於我們資料切割的數量，才有辦法將資料解密回來，所以現有的辦法是當儲存伺服器收到一定數量的區塊就會做一次的儲存，就有辦法突破儲存伺服器必須大於或等於我們資料切割的數量。

為了要突破以上的一些限制，所以將系統改成階層式的做法，在我們的儲存系統上用 Session Key 來對我們的文件作加密，用我們的公開金鑰系統的公要來對 Session Key 作加密，如此一來可有效的增強安全強度，也可以突破伺服器數量的限制。

參考文獻

[1] Android Developers ,Available 2010年3月at http://developer.android.com/index.html.

[2] Java.sun.com , Available 2010年3月at http://java.sun.com/.

[3] Eclipse.org home , Available 2010年3月at http://www.eclipse.org/.

[4] Erasure code, Available 2010年3月at http://en.wikipedia.org/wiki/Erasure_code.

[5] Cloud computing , Available 2010年3月at http://en.wikipedia.org/wiki/Cloud_computing

[6] Java Pairing-Based Cryptography Library , Available 2010年3月at http://gas.dia.unisa.it/projects/jpbc/index.html

[7] PBC library manual, Available 2010年3月at http://crypto.stanford.edu/pbc/manual/

[8] 智能監控Lite 1.1.0 , Available 2010年3月at http://www.kiumiu.com

[9] Hsiao-Ying Lin, Wen-Guey Tzeng, "A Secure Decentralized Erasure Code for Distributed Networked Storage," IEEE Transactions on Parallel and Distributed Systems, 26 Jan. 2010. IEEE computer Society Digital Library.

[10] 佘志龍、陳昱勛、鄭名傑、陳小鳳、郭秩均，Google Android SDK 開發範例大全，第一版，2009年4月出版。

[11] Anoop MS, "Elliptic Curve Cryptography An Implementation Guide".

[12] Alexandros G. Dimakis, Vinod Prabhakaran, Kannan Ramchandran, " Decentralized Erasure Codes for Distributed Networked Storage," IEEE Transactions on Information Theory, 12 Jun *2006*.

[13] Lawrence C. Washington, "Elliptic Curves Number Theory and Cryptography", Second Edition,2008年出版。