一個有效率的免憑證密碼系統之環簽章機制

李宗諺
中興大學資訊科學與工程學系
s9856048@cs.nchu.edu.tw

陳昱圻
中興大學資訊科學與工程學系
phd9809@cs.nchu.edu.tw

洪國寶
中興大學資訊科學與工程學系
gbhorng@cs.nchu.edu.tw

摘要

為了解決在傳統公開金鑰系統下公鑰憑證，以及在基於身份密碼系統下金鑰託管的問題。免憑證公開金鑰密碼系統於 2003 年首次由 Al-Riyami 和 Paterson 提出來處理這些問題。在 2009 年 Chang 等學者提出免憑證門檻型環簽章。由群組內的一個人，產生代表所有群組成員的環簽章，而驗證者可以驗證此簽章來自此群組，但是卻不知道由群組內哪個成員所簽屬。本文我們提出一個，有效率的免憑證密碼系統之環簽章機制。我們提出的方法不使用複雜的 pairing 計算，其安全性建構在解離散對數問題下。

關鍵詞：環簽章、基於身份密碼系統、免憑證密碼系統、免憑證環簽章。

Abstract

For solving the key escrow problem in identity-based public key cryptography, certificateless public key cryptography was first invented in 2003 by Al-Riyami and Paterson. In 2009, Chang et al. presented a certificateless threshold ring signature scheme. Any member can generate a ring signature to represent n members in this group, and the verifier only knows this signature is from the group. In this paper, we propose an efficient certificateless threshold ring signature scheme without pairing. In addition, it is provably secure based on the discrete logarithm problem.

Keywords: Ring signature, ID-based cryptography, Certificateless cryptography, Certificateless ring signature

1. 前言

在傳統的公開金鑰密碼系統，一個使用者的公鑰要透過可信賴的授權憑證 (Certificate Authority, CA) 中心，才能取得公鑰憑證。顯而易見的，就需要花更多的時間去管理這些憑證。為了簡化憑證過程，Shamir[7]首先提出了基於身分加密系統(ID-PKC)，不需要原本公鑰憑證，取而代之的是公開的、已知的資訊來當作使用者的公鑰，例如 e-mail 或是 ID 等。但是卻產生了一個內在的問題，因為 ID-PKC 必須經由金鑰產生中心(Key Generation Center, KGC)的 master key 去產生每個使用者的密鑰。然而一個惡意的 KGC 就有能力去偽造任何使用者的合法簽章。這就是 ID-PKC 的公鑰託管問題。在 2003 年 Al-Riyami 和 Paterson[1]提出了一個免憑證公鑰加密法(CL-PKC)去取代原先的為 ID-PKC 中的公鑰憑證的使用，並解決了金鑰託管的問題。CL-PKC 基本的概念是使用者選取的一個秘密值以及 KGC 所給的部分私鑰去決定公密鑰對。

環簽章可以分為一般型的環簽章以及門檻型的環簽章：一般型的環簽章，由一個群組內的成員可以產生代表所有成有成員簽署的簽章，而驗證者可以驗證此簽章來自此群組，但卻無法得知由哪個成員所執行的。將一般型的環簽章進而改進成門檻型的環簽章，使用類似私密分享的方式，讓門檻值 t 個的成員共同來產生環簽章，其驗證如同一般型環簽章。已經有不少研究以深入討論了環簽章系統[4][8]。我們提出免憑證環簽章機制(CLRS)，可以更實際地來應用。

本文第二章描述一些基礎知識以及 CLRS

的定義與安全訴求；第三章我們提出一個有效率的免憑證環簽章機制以及分析與討論，最後為本文結論。

2. 相關研究

2.1 免憑證簽章及其安全性

免憑證環簽章方法包含六個 probabilistic polynomial-time(PPT)演算法：

- SetUp：輸入 1^k 其中 k 秘密參數，此演算法產生系統參數，以符號 *param* 表示。
- MasterKeyGen：輸入 *param*，此演算法產生一對 master public/secret key(*mpk, msk*)。
- PartialKeyGen：輸入 (*param, mpk, ID*)，$ID \in \{0,1\}^*$ 為使用者的身分，此演算法產生使用者部分私密金鑰 psk_{ID}。
- UserKeyGen：輸入(*param, ID*)，此階段產生使用者公私鑰對(upk_{ID}, usk_{ID})。
- Sign：輸入(*param, mpk, R, S, m*)，其中 $R = L \bigcup \{upk_i : i \in L\}$，$L$ 為環成員中身分的集合，$S = \{psk_{ID_\pi}, usk_{ID_\pi}\}$，$ID_\pi \in L$ 是實際的簽章者，$m \in \{0,1\}^*$ 則為欲簽署的訊息，最後產生一個免憑證的環簽章 σ。
- Verify：輸入(*param, mpk, R,* 1*, m,* σ)，此演算法最後輸出 1 代表驗證者接受，否則輸出 0 表示不接受。

首先介紹在證明會用到的 oracle 及其簡單的描述：

1. **CreateUser**：輸入使用者 *ID*，如果 $ID \notin L_0$，則執行 PartialKeyGen 以及 UserKeyGen，並且設置 $L_0 = L_0 \bigcup \{ID\}$。最後回傳 upk_{ID}。
2. **RevealPartialKey**：輸入 *ID*，如果 $ID \in L_0$，則回傳 psk_{ID} 以及設置 $L_1 = L_1 \bigcup \{ID\}$；否則回傳 $\perp$。
3. **ReplacePartialKey**：輸入 *ID* 和使用者部分私鑰 psk'_{ID}，如果 $ID \in L_0$，將 psk'_{ID} 取代 psk_{ID}，以及設置 $L_2 = L_2 \bigcup \{ID\}$。
4. **RevealSecretKey**：輸入 *ID*，如果 $ID \in L_0$，則回傳 usk_{ID} 以及設置 $L_3 = L_3 \bigcup \{ID\}$；否則回傳 $\perp$。
5. **ReplacePublicKey**：輸入 *ID* 和 (upk'_{ID}, usk'_{ID})，如果，將 upk'_{ID} 取代 upk_{ID}，將 usk'_{ID} 取代 usk_{ID}，設置 $L_4 = L_4 \bigcup \{ID\}$。
6. **Osign**：輸入訊息 m，兩個身分集合 T和 L，$T \subseteq L \subseteq L_0$ 最後回傳 Sign(*mpk, R, S, m*)，其中 $R = L \bigcup \{upk_i : i \in L\}$，$S = \{(psk_j, usk_j) : j \in T\}$，設置 $L_5 = L_5 \bigcup \{ID\}$。

在分析不可偽造性，我們會給予以下兩個 game 來模擬 Type Ⅰ與 Type Ⅱ攻擊者之行為：

Game Unforgeability against Type Ⅰ Adversary $\boldsymbol{A_I}$：$\boldsymbol{l_I}$ 為 challenger，k 屬於 N 是一個秘密參數。

1. $\boldsymbol{l_I}$ 執行 SetUp(1^k)得到 *param*，輸入 *param* 並請求 $\boldsymbol{A_I}$，從 $\boldsymbol{A_I}$ 得到 master public key。將(*param, mpk*)傳送給 $\boldsymbol{A_I}$。
2. 當 $\boldsymbol{A_I}$ 收到 *mpk* 之後開始詢問每個 oracle。
3. $\boldsymbol{A_{II}}$ 開始詢問所有的 oracles，最後輸出(L^*, t^*, m^*, σ^*)。

   我們宣稱 $\boldsymbol{A_{II}}$ 贏了這場遊戲，如果：

   1. $L^* \subseteq L_0$。
   2. $Verify(param, mpk, R^*, t, m^*, \sigma^*) = 1$ 其中 $R^* = L^* \bigcup \{upk_i : i \in L^*\}$。
   3. $|L^* \bigcap (L_1 \bigcup L_2) \bigcap (L_3 \bigcup L_4)| < t$。
   4. $(m^*, R^*, T^*) \notin L_5$ 其中 $T^* \subseteq L^*$。

Game Unforgeability against Type Ⅱ Adversary $\boldsymbol{A_{II}}$：$\boldsymbol{l_{II}}$ 為 challenger，k 屬於 N 是一個秘密參數。

4. $\boldsymbol{l_{II}}$ 執行 SetUp(1^k)得到 *param*，輸入 *param* 並請求 $\boldsymbol{A_{II}}$，此時 $\boldsymbol{A_{II}}$ 被限制不能存取任何

的 oracle。

5. $\boldsymbol{A}_{\text{II}}$ 開始詢問所有的 oracles 除了 RevealPartialKey，最後輸出$(L^*, t^*, m^*, \sigma^*)$。我們宣稱$\boldsymbol{A}_{\text{II}}$贏了這場遊戲，如果：
   1. $L^* \subseteq L_0$。
   2. $Verify(param, mpk, R^*, t, m^*, \sigma^*) = 1$ 其中 $R^* = L^* \bigcup \{upk_i : i \in L^*\}$。
   3. $|L^* \bigcap (L_3 \bigcup L_4)| < t$。
   4. $(m^*, R^*, T^*) \notin L_5$ 其中 $T^* \subseteq L^*$。

接著，利用以下 game 來證明其匿名性：

Game Anonymity:

$\boldsymbol{I}$ 為 challenger， $\boldsymbol{A}$ 為攻擊者，k 屬於 N 是一個秘密參數。

1. $\boldsymbol{I}$ 執行 SetUp(1^k)並且請求 $\boldsymbol{A}$ 為了獲得 mpk。在這個階段，$\boldsymbol{A}$ 不被允許存取任何一個 oracle。
2. $\boldsymbol{A}$ 可以開始向 oracle 提出詢問，除了 CreateUser 進一步修改輸入為(ID, psk_{ID})，如果 ID 不屬於 L_0 則執行 UserKeyGen，以及回傳 upk_{ID} 和在 UserKeyGen 中所用到的 w_{ID}，因此 $\boldsymbol{A}$ 可以從新計算 usk_{ID}。這裡也表示著 RevealSecretKey 是不需要用到的。
3. $\boldsymbol{A}$ 請求一個 challenge 藉由傳送 (T_0^*, T_1^*, L^*, m^*) 的值給 $\boldsymbol{I}$。其中 $T_0^* \bigcup T_1^* \subseteq L^* \subseteq L$， $|T_0^*| = |T_1^*| = t$ 但是 $|T_0^* \bigcap T_1^*| < t$。$\boldsymbol{I}$ 選擇 $b \xleftarrow{R} \{0,1\}$，並且連同 $\sigma \leftarrow Sign(param, mpk, R^*, S_b^*, m^*)$ 一起提供給 $\boldsymbol{A}$， $R^* = L^* \bigcup \{upk_i : i \in L^*\}$ 以及 $S_b^* = \{(psk_j, usk_j) : j \in T_b^*\}$。
4. $\boldsymbol{A}$ 對於 b 輸出其猜測的 b'。

$\boldsymbol{A}$ 贏的遊戲的條件為 $b = b'$ 且 $Verify(param, mpk, R^*, t, m^*, \sigma^*) = 1$。$\boldsymbol{A}$ 的優勢為贏得遊戲的機率大於隨機猜測(i.e. 1/2) 的機率。

2.2 Chang's et al. 之免憑證門檻型環簽章機制

本章節首先介紹學者所提出的免憑證門檻型環簽章的機制，其方法詳細敘述如下：

- **SetUp**：給定一個安全參數 k，$k \in N$。輸入 1^k，隨機挑選兩個循環群 G_1 和 G_1 其 order 為 q，g 為 G_1 的生成元，存在一個可計算性的 bilinear map $e: G_1 \times G_1 \to G_2$ 並且挑選兩個 hash function 分別為 $H_1 : \{0,1\}^* \to G_1$ 以及 $H_2 : \{0,1\}^* \to Z_q^*$。最後設置並輸出系統參數 $param = (1^k, G_1, G_2, q, g, H_1, H_2)$。
- **MasterKeyGen**：輸入 $param$，隨機選取 $msk \in_R Z_q$ 為系統秘密金鑰 msk 並且設置 master public key 為 $mpk = g^{msk}$。
- **PartialKeyGen**：輸入($param$, mpk, ID)，設定 partial key 為 $psk = H_1(ID)^{msk}$。
- **UserKeyGen**：輸入($param$, ID)，挑選一隨機亂數 $usk_{ID} \in_R Z_q$ 而使用者公開金鑰為 $upk: g^{usk_{ID}}$。
- **Sign**：輸入($param$, mpk, R, S, m)，其中 $R = L \bigcup \{upk_i : i \in L\}$，$L$ 為環成員中身分的集合，$S = \{psk_{ID_\pi}, usk_{ID_\pi}\}$，$ID_\pi \in L$ 是實際的簽章者。完成簽章的過程如下：
  首先隨機挑選 $r_1, r_2 \in_R Z_q$ 以及對於環成員中每位成員，除了簽章者 ID_π 皆挑選 $c_i \in_R Z_q$。計算

  $c = H_2(param, mpk, R, m, g^{r_1} \Pi_{i \in L \setminus \{ID_\pi\}} upk_i^{c_i}, e(g^{r_2}, g) \Pi_{i \in L / ID_\pi} e(H_1(i), mpk)^{c_i})$

  、 $c_{ID_\pi} = c - \sum_{i \in L / \{ID_\pi\}} c_i$ 、 $s = r_1 - c_{ID_\pi} \cdot usk_{ID_\pi}$ 、

  $W = g^{r_2} \Big/ psk_{ID_\pi}^{c_{ID_\pi}}$。最後輸出簽章為

$\sigma = (s, W, \cup_{i \in L}\{c_i\})$。

Verify：輸入(*param, mpk, R,* 1*, m,* σ)，其中為 $\sigma = (s, W, \cup_{i \in L}\{c_i\})$，判斷 $\sum_{i \in L} c_i \equiv ? H_2(param, mpk, R, m, g^s \Pi_{i \in L} upk_i^{c_i})$，$e(W, g)\Pi_{i \in L / ID_\pi} e(H_1(i), mpk)^{c_i})$(mq) o 相等輸出 1 表示驗證者接受此簽章，不相等則輸出 0。

3. 不採用 pairing 的免憑證環簽章

3.1 免憑證環簽章機制

我們所提出的免憑證環簽章機制有幾個角色，分別為 KGC、簽章者以及驗證者，而方法由下列幾個演算法所組成：

- **SetUp**：給定一個安全參數 k，$k \in N$。輸入 1^k，隨機挑選 prime order 為 q 的乘法循環群 G_1，g 為 G_1 的生成元，並且隨機挑選兩個 hash function 分別為 $H_1 : \{0,1\}^* \to Z_q^*$ 以及 $H_2 : \{0,1\}^* \to Z_q^*$。最後設置並輸出系統參數 $param = (1^k, G_1, q, g, H_1, H_2)$。
- **MasterKeyGen**：輸入 *param*，隨機選取 $x \in_R Z_q$ 為系統秘密金鑰 *msk* 並且設置 master public key 為 $mpk = g^x$。
- **PartialKeyGen**：輸入(*param, mpk, ID*)，設定 partial key 為 $psk = s + xH_1(ID)$，其中 $s \in_R Z_q^*$ 為一隨機亂數，設定 $upk_{i,2} = g^s$。
- **UserKeyGen**：輸入(*param, ID*)，挑選一隨機亂數 $z \in_R Z_q^*$ 設 z 為使用者私密金鑰而使用者公開金鑰為 $upk : \{upk_{i,1} = g^z, g^s\}$。
- **Sign**：輸入(*param, mpk, R, S, m*)，其中 $R = L \bigcup \{upk_i : i \in L\}$，$L$ 為環成員中身分的集合，$S = \{psk_{ID_\pi}, usk_{ID_\pi}\}$，$ID_\pi \in L$ 是實際的簽章者。完成簽章的過程如下：

  首先隨機挑選 $r_1, r_2 \in_R Z_q$ 以及對於環成員中每位成員，除了簽章者 ID_π 皆挑選 $c_i \in_R Z_q$。接著計算

  $c = H_2(param, mpk, R, m, (g^{r_1} \Pi_{i \in L \setminus \{ID_\pi\}} upk_{i,1}^{c_i}), (\Pi_{i \in L / ID_\pi} (upk_{i,2} \cdot mpk^{H_1(ID)})^{c_i} \cdot g^{r_2}))$

  、$c_{ID_\pi} = c - \sum_{i \in L / \{ID_\pi\}} c_i$ 、$u = r_1 - c_{ID_\pi} \cdot usk_{ID_\pi}$ 、

  $v = r_2 - c_{ID_\pi} \cdot psk_{ID_\pi}$。最後輸出簽章為

  $\sigma = (u, v, \cup_{i \in L}\{c_i\})$。
- **Verify**：輸入(*param, mpk, R,* 1*, m,* σ)，其中為 $\sigma = (u, v, \cup_{i \in L}\{c_i\})$，判斷 $\sum c_i \equiv ? H_2(param, mpk, R, m, (g^u \Pi_{i \in L} upk_{i,1}^{c_i}), (\Pi_{i \in L} (upk_{i,2} \cdot mpk^{H_1(ID)})^{c_i} \cdot g^v))$(mq) o d 相等輸出 1 表示驗證者接受此簽章，不相等則輸出 0。

3.2 分析與討論

以下分析參考 Boneh 等人的方法提出的機制其不可偽造性和其匿名性。[2][3]

Theorem 1.

若 $\boldsymbol{A_I}$ 贏得 Game Unforgeability against Type Ⅰ Adversary 的優勢至少為 ε_{A_I}，而時間為 T，最多向 signature oracle 詢問 q_s 次以及最多向 random oracle 詢問 q_H 次，在 G_1 解決離散對數問題的機率至少為 $1/q_H^2 \cdot \varepsilon_{A_I} / poly(k)$，而所需時間則為 $T + q_s t_1 + q_H t_2$，$poly(k)$是多項式包含秘密參數 k，t_1 和 t_2 分別為模擬 signature oracle 及 random oracle 所需的時間。[6]

Proof 1.

假設在 G_1 裏頭給定一個隨機實例 $g_1 = g^a$。

1. H_1：輸入 *ID*，假如這不是第 π 次的詢問，也就是說當 $i \neq \pi$，$\boldsymbol{I_I}$ 隨機選取 $\alpha_i \in_R Z_q$ 並且設置

$upk_{i,2}$ 以及 $H_1(ID)$，設 $upk_{i,2} \cdot mpk^{H_1(ID)} = g^{\alpha_i}$ ；否則 $\boldsymbol{l_I}$ 設 $upk_{i,2} \cdot mpk^{H_1(ID)} = (g_1)^{\alpha_i}$ 。$\boldsymbol{l_I}$ 儲存一個清單 L_{h_1} ，其中記錄著 $(ID, \alpha_i, H_1(ID))$。

2. H_2：輸入 δ_i ，$\boldsymbol{l_I}$ 隨機選取 $h_i \in_R Z_q$ ，$\boldsymbol{l_I}$ 將 (δ_i, h_i) 儲成一個清單 L_{h_1} 。
3. CreateUser ：輸入 ID，如果 $ID \in L_0$ ，則回傳之前所產生的 upk_{ID}(從 L' 的清單中得到)。否則 $\boldsymbol{l_I}$ 將 ID 帶入 H_1 做詢問。在此假設清單 L_{h_1} 中的值組為 $(ID, \alpha_{ID}, H_1(ID))$。如果這不是第 π 次的詢問， $\boldsymbol{l_I}$ 設置 $psk_{ID} = s + xH_1(ID)$。否則 $\boldsymbol{l_I}$ 將 psk_{ID} 設為 $\perp$ 。對於上述兩種例子，當 $z_{ID} \in_R Z_q$ 則 $\boldsymbol{l_I}$ 回傳 $upk_{ID,1} = g^{z_{ID}}$。$\boldsymbol{l_I}$ 將更新清單 $L_0 = L_0 \cup \{ID\}$，並且將 $(ID, psk_{ID}, upk_{ID}, \alpha_{ID}, z_{ID})$ 儲存在清單 L'。
4. RevealPartialKeys：輸入 ID，如果 $ID \notin L_0$ ，則回傳 $\perp$ ，否則從清單 L' 中取回 psk_{ID} 並且回傳 $psk_{ID} \neq \perp$。B_1 設置清單 $L_1 = L_1 \cup \{ID\}$。如果 $psk_{ID} \neq \perp$ 則 $\boldsymbol{l_I}$ 捨棄並且停止動作
5. ReplacePartialKey ：輸入 ID 、 psk'_{ID} ，如果 $ID \in L_0$，則清單 L' 將原本的 psk_{ID} 取代更新成 psk'_{ID}。$\boldsymbol{l_I}$ 設置清單 $L_2 = L_2 \cup \{ID\}$ 。
6. RevealSecretKey：輸入 ID，如果 $ID \in L_0$ ，則從清單 L' 中取回 z_{ID} 、設置清單 $L_3 = L_3 \cup \{ID\}$ 並且回傳 z_{ID} ，否則回傳 $\perp$ 。
7. RelpacePublicKey：輸入 ID、upk'_{ID} 以及 usk'_{ID} ，如果 $ID \notin L_0$，則清單 L' 將原本的 upk_{ID} 取更新成 upk'_{ID} 而 usk_{ID} 取代成 usk'_{ID}。接著 $\boldsymbol{l_I}$ 設置清單 $L_4 = L_4 \cup \{ID\}$ 。
8. Osign：輸入 m、L 以及 ID，設置計算 $\delta = (param, mpk, R, m, (g^u \Pi_{i \in L} upk_{i,1}^{c_i}), (\Pi_{i \in L}(upk_{i,2} \cdot mpk^{H_1(ID)})^{c_i} \cdot g^v))$ 計算 $c = \sum_{i \in L} c_i$ ，將 $H_2(\delta)$ 指定給 c。最後輸出，並將 (δ, c) 加入清單 L_{h_1} 中以及更新 $L_5 = L_5 \cup \{m, R, ID\}$ 。

藉由模擬 oracle 兩次，在第二次的模擬過程中 $\boldsymbol{l}$ Ⅱ隨機選取 c'^* ， $c'^* \neq c^*$ 。當 $c'^* \neq c^*$ ，其中 $c'^* = \sum_{i \in L^*} c'^*_i \mod q$，$c^* = \sum_{i \in L^*} c^*_i \mod q$，在這裡至少有一對 (c'^*_β, c^*_β) ， $c'^*_\beta \neq c^*_\beta$ ，$\beta \in L^*$。兩次的模擬當中所產生的合法簽章中，可以得到

$$g^{v^*}(upk^*_{\beta,2} \cdot mpk^{H_1(ID)})^{c_\beta} = g^{v'^*}(upk'^*_{\beta,2} \cdot mpk^{H_1(ID)})^{c'_\beta}$$

而可以得到 $g^{u^* - u'^*} = (upk'^*_{\beta,2} \cdot mpk^{H_1(ID)})^{c'^*_\beta - c^*_\beta}$ 。

如果是 β CreateUser 第 π 次詢問，則 $upk_{i,2} \cdot mpk^{H_1(ID)} = (g_1)^{\alpha_\beta}$ ，a 就等於 $a = (v^* - v'^*) / (c'^*_\beta - c^*_\beta)\alpha_\beta$ 。

Theorm 2.

若 $\boldsymbol{A}$ Ⅱ贏得 Game Unforgeability against Type Ⅰ Adversary 的優勢至少為 $\varepsilon_{A_{II}}$ ，而時間為 T，最多向 CreatUser 詢問 q_{CU} 次、最多向 signing oracle 詢問 q_s 次以及最多向 random oracle 詢問 q_H 次，在 G_1 中解決離散對數問題的機率至少為 $1/q_{CU}^2 \cdot \varepsilon_{A_I} / poly(k)$ ，而所需時間則為 $T + q_s t_1 + q_H t_2$ ，$poly(k)$是多項式包含秘密參數

k，t_1和 t_2 分別為模擬 signature oracle 及 random oracle 所需的時間。

Proof 2.

I_{II}為離散對數問題的攻擊者，A_{II}是在 Game Unforgeability against Type Ⅱ Adversary 的敵人。假設在 G_1 裏頭給定一個隨機實例 $g_1 = g^a$。我們會顯示當 A_{II}贏得遊戲時，則 I_{II}可以計算出 a 的值。首先 I_{II}挑選 q、G_1以及 g 然後執行 SetUp(1^k)。*param* 被設為(G_1, q, g, H_1, H_2)。I_{II}請求 A_{II}來獲得 *mpk*。$mpk = g^x$，$x \in Z_q$。I_{II}隨機選取一個指標 $\pi \in [1, q_{CU}]$。而 oracles 的模擬過程如下：

1. H_1：輸入 ID，隨機挑選 $Q_{ID} \in G_1$ 並且回傳之。I_{II}包含一個清單 L_{h_1} 用來存放 (ID, Q_{ID})。
2. H_2：同證明一，輸入 ξ_i， I_{II}隨機選取 $h_i \in_R Z_q$，I_{II}將 (ξ_i, h_i)存放於清單 L_{h_2}。
3. CreateUser：輸入 (ID, psk_{ID})，如果 $ID \in L_0$，則檢索 L' 並回傳 $upk_{ID,1}$。否則，I_{II}挑選 $z_{ID} \in_R Z_q$ 並且設置 $upk_{ID,1} = g^{usk_{ID}}$。在這之中，如果此次非第 π 次的詢問，則設置 $usk_{ID,1} = z_{ID}$。如果是第 π 次的詢問，則設置 $upk_{ID,1} = g_1^{z_{ID}}$ 以及 $usk_{ID,1} = \perp$。在這兩個例子中，皆會回傳 $upk_{ID,1}$，並設置 $L_0 = L_0 \cup \{ID\}$ 以及將 $(ID, psk_{ID}, usk_{ID,1}, upk_{ID,1}, z_{ID})$ 存放在清單中。
4. ReplacePartialKey：同證明一，輸入 ID、psk'_{ID}，如果 $ID \in L_0$，則清單 L' 將原本的 psk_{ID} 取代更新成 psk'_{ID}。B_1設置清單 $L_2 = L_2 \cup \{ID\}$。
5. RevealSecretKey：輸入 ID，如果 $ID \notin L_0$ 則回傳 $\perp$。否則，如果 $usk_{ID,1} \neq \perp$ 回傳 $usk_{ID,1}$ 並設置 $L_3 = L_3 \cup \{ID\}$；如果 $usk_{ID,1} = \perp$ 則停止並丟棄。
6. ReplacePublicKey：同證明一，輸入 ID，如果 $ID \in L_0$，則從清單 L' 中取回 x_{ID}、設置清單 $L_3 = L_3 \cup \{ID\}$ 並且回傳 x_{ID}，否則回傳 $\perp$。
7. Osign：同證明一

藉由模擬 oracle 兩次，在第二次的模擬過程中 I_{II}隨機選取 c'^*，$c'^* \neq c^*$。當 $c'^* \neq c^*$，其中 $c'^* = \sum_{i \in L^*} c_i'^* \mod q$，$c^* = \sum_{i \in L^*} c_i^* \mod q$，在這裡至少有一對 (c'^*_β, c^*_β)，$c'^*_\beta \neq c^*_\beta$，$\beta \in L^*$。兩次的模擬當中所產生的合法簽章中，可以得到

$$g^{u^*} upk_{\beta,1}^{*c^*_\beta} = g^{u'^*} upk_{\beta,1}^{*c'^*_\beta}$$

而可以得到 $g^{u^* - u'^*} = upk_{\beta,1}^{*c'^*_\beta - c^*_\beta}$。如果是 β CreateUser 第 π 次詢問，則 $upk^*_\beta = (g^a)^{z_\beta}$，$a$ 就等於 $a = (u^* - u'^*) / (c'^*_\beta - c^*_\beta) z_\beta$。

Theorem 3.

免憑證環簽章在 Game Anonymity 是保持匿名的。

Proof 3.

此證明的想法是參考 Dodis 等學者[5]在 Anonymous Identification in *Ad Hoc* Groups 所

提出的 anonymity proof。

我們定義了兩個遊戲 games E_0 和 E_1，E_0 是原本的 anonymity game 定義如上，而 E_1 是修改過的版本。在開始敘述細節之前，先來回顧一下 Game Anonymity。總共有四個步驟，在步驟一和二當中，challenger $\boldsymbol{l}$ 為了得到 mpk 請求 adversary $\boldsymbol{A}$ 並且模擬相對應的 oracles。在步驟三之中，$\boldsymbol{A}$ 傳送兩個簽章者的集合給 $\boldsymbol{l}$，接著 $\boldsymbol{l}$ 隨機選取其中之一的簽章者集合以及產生一個相對應的簽章。在最後的步驟四，$\boldsymbol{A}$ 輸出所猜測的簽章者集合，而簽章是可以對應的。我們定義兩個遊戲 E_0 和 E_1 如下：

遊戲 E_0：和原本的 Game Anonymity 相同。

遊戲 E_1：這個遊戲在原本的 Game Anonymity 遵循了步驟一、二和四。

對於利用 Sign 所產生的簽章 $\sigma=(u,v,\cup_{i\in L}\{c_i\})$，如果 $i\in L$ 不是真實的簽章者，則 c_i 的選擇是獨立的且均勻分散在 Z_q。如果 i 是真實的簽章者，由於 c 是 random oracle 所輸出的，c_i 是獨特的，並由 c 和 c_j ($j\in L$ 但不是真實的簽章者)所決定的。當所有的參數用來決定 c_i 皆為均勻分散且各自相互獨立，c_i 也會是均勻分布的。此外，當 r、r 是在 Z_q 隨機且均勻的選擇並且獨立於 c_i、psk_i 和 usk_i，則 u 和 v 也會是分別均勻分布在 Z_q 以及 G_1。對於一個由 OSign 所產生的簽章，我們可以見到所有的提到的參數皆為均勻分布且相互獨立的。因此在遊戲中 E_0 猜測到正確的 b 值為 1/2 且在兩個遊戲中的分布是相同的。換句話說 $\boldsymbol{A}$ 在 Game Anonymity 的優勢是可忽略的。

我們所提出的環簽章機制運用在演憑證密碼系統上在驗證部分不採用計算量較大的 pairing，因此在計算效率上比 Chang 等學者更有效率。在安全分析上，我們沿用 Chang 等人的安全定義，經由以上證明，我們的機制建構於離散對數安全性可以滿足不可偽造性以及分析其匿名性。

4. 結論

免憑證密碼系統在近幾年受到極大的關注，主因是它的架構解決了基於身份密碼系統的潛在安全問題，更適合用於安全通訊上，相關的應用已經有不少研究提出了。本文提出了一個有效率的免憑證密碼系統上的環簽章機制，因為不使用 pairing，其效率上頭會較 Chang 等人的機制來的好。而我們亦分析了其安全性。

致謝

本研究由國科會補助完成，計畫編號：NSC99-2221-E-005-078。

參考文獻

[1] S. Al-Riyami and K. Paterson, “Certificateless public key cryptography”, ***Asiacrypt2003***, LNCS 2894, pp. 452-473, Springer-Verlag, 2003.

[2] D. Boneh and X. Boyen, “Short Signatures Without Random Oracles”, ***Eurocrypt2004***, LNCS 3027, pp. 56-73, 2004.

[3] D. Boneh, H. Shacham, and B. Lynn, “Short signatures from the Weil pairing”, ***Journal of Cryptology***, Vol. 17, No. 4, pp. 297-319, 2004.

[4] S. Chang, D.S. Wong, Y. Mu, and Z. Zhang, “Certificateless Threshold Ring Signature”, ***Information Sciences***, Vol. 179, No. 20, pp. 3685-3696, 2009.

[5] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. “Anonymous identi_cation in ad-hoc groups.” ***In Advances in Cryptology Eurocrypt 2004***, volume 3027 of Lecture Notes in Computer.

[6] B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng, “Key Replacement Attack Against a Generic Construction of Certificateless Signature”, ***ACISP 2006***, LNCS 4058, pp.235-246, Springer-Verlag, 2006.

[7] A.Shamir, “Identity based cryptosystems and signature schemes”, ***Crypto 1984***, LNCS, Vol.196, pp. 47-53, Springer-Verlag, 1984.

[8] L. Zhang, F. Zhuang, W. Wu, “A provably secure ring signature scheme in certificateless cryptography”, ***Provsec 2007***, LNCS, vol. 4784, Springer-Verlag, pp. 103–121, 2007.