行動裝置之兩階段身分認證機制

羅嘉寧
銘傳大學 電腦與通訊工程學系
deer@mail.mcu.edu.tw

摘要

隨著雲端服務的興起，使用者有越來越多的資訊儲存在雲端服務中。傳統的文字密碼機制已無法保障使用者的帳號安全，使用第二因子認證及兩階段認證已逐漸成為趨勢。本文介紹 Google 所提供的兩階段驗證機制 Google authenticator，及 FIDO 聯盟所提出的通用認證框架 (Universal Authentication Framework，UAF)、通用第二因子認證 (Universal 2nd Factor，U2F) 及 FIDO2 機制。

關鍵詞：兩階段驗證、身分認證、第二因子認證

Two-step mobile user authentication mechanisms

Jia-Ning Luo
Department of Information and Telecommunications Engineering, Ming-Chuan University.
deer@mail.mcu.edu.tw

Abstract

With the rise of cloud services, users have more and more information stored in cloud services. The traditional text password mechanism has been unable to protect the user's account security. The use of the second factor authentication and two-stage authentication has gradually become a trend. This article introduces the Google authenticator, a two-stage authentication mechanism provided by Google, and the Universal Authentication Framework (UAF), Universal 2nd Factor (U2F), and FIDO2 mechanisms proposed by the FIDO Alliance.

Keywords: two-step verification, user authentication, second-factor authentication

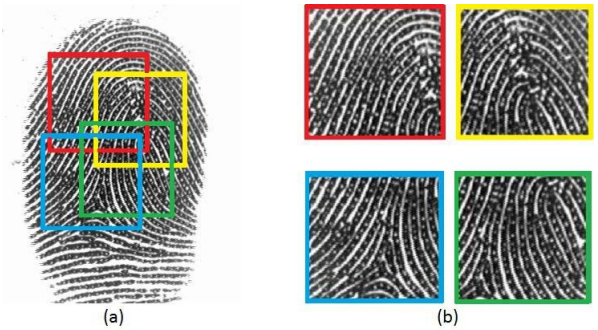
壹、前言

在當前的社會中，當今行動裝置提供了極大的靈活性與便利性。人們漸漸地採用智慧型手機來處理日常事務及進行線上金融交易，其中許多牽涉到使用者的敏感資訊。因此，行動裝置的認證機制亦發重要。

在早期行動裝置僅使用文字密碼來驗證使用者的身分。然而文字密碼卻容易被竊取。例如在多個網站上使用相同密碼，透過網路下載軟體，按下電子郵件訊息或簡訊中的連結。為了減少文字密碼的使用，並更加強化行動裝置的安全性，行動裝置的身份認證逐漸衍生出雙因子身分認證 (Two-Factor-Based Authentication, 2FA)、多因子認證(Multi-Factor authentication, MFA)或兩階段認證(two-step verification)。雙因子身分認證是通過結合兩種不同的身分認證因子對合法使用者進行身分驗證的一種方式。常用的雙因子身分認證組合為使用密碼加上個人的生物特徵，如指紋或是臉部特徵。

使用個人手指指紋的生物特徵來辨識個人的身分已是一個歷史悠久且應用相當廣泛的技術。指紋的特徵具有持久性、獨特性及再生性。在人的一生中，從出生到老死，其指紋並不會有太大的改變，即使磨損也會恢復原形。尤其是手指指紋的再生性，比其他部位的指紋來個更強。而且人與人之間的指紋特徵皆不相同，很難找到兩個人具有完全相同指紋特徵。

然而使用指紋驗證並不是一定安全。在 Roy 等人的研究 [13]中發現，因為在實務上由於全尺寸的指紋感測器所需感測元件之體積較大，因此在智慧型手機上面的指紋感測器並非採用全尺寸的指紋感測器，而只是一個精簡版的小型感應器。例如 iPhone 上的指紋感測器只能感應手指中心約 0.8cm x 0.8cm 的面積，而非整隻手指的指紋。所以系統僅是比對由指紋感測器所讀到的部分指紋來與系統資料庫中的指紋比對，只要有部分符合即可通過身分認證。當使用者於認證資料庫中輸入多隻手指的指紋資料時，亦即只要由指紋感測器所讀到的部分指紋，跟任一手指之部分比對成功即可。在 Roy 的研究中，嘗試藉由指紋資料庫的比對，找到符合智慧型手機上面的指紋感測器大小，且與個人指紋相似的的部份指紋，稱為 MasterPrint。Roy 發現在身分驗證時，若在五次的嘗試登入過程中使用五個 MasterPrint，則登入成功率為 26.46%，遠高於全指紋感測器的錯誤接受率 (False Acceptance Rate, FAR) 的 0.1%。



圖一：由完整指紋鎖擷取之部份指紋 [13]

另一種的生物身分驗證方式為採用臉部辨識。例如 Apple 的 Face ID 技術為利用原深感測相機擷取超過 30,000 個隱形的點並加以分析，進而製作臉部的深度測繪圖，

同時擷取臉部的紅外線影像。然後再將深度測繪圖和紅外線影像轉換為數學表徵，並將此表徵與登記的臉部資料比對。

然而使用指紋或 Face ID 認證並無法完全保護使用者的隱私。使用者有權拒絕透露所使用的文字密碼，但卻無法拒絕司法單位採取其生物識別資訊。有新聞指稱, 於今年八月，一個美國的 FBI 探員於取證過程中，要求一犯罪嫌疑人使用 Face ID 掃描以解鎖其手機。

諸如 Apple、Google 或 Microsoft 等公司，為了進一步的加強身分認證的安全性，逐漸的採用兩階段驗證機制。最早期的兩階段驗證機制通常是綁定使用者的手機門號或電子郵件帳號，再傳送一特定的網站連結或一個一次性密碼至使用者設定的手機門號或電子郵件帳號中。使用者必須將收到的一次性密碼輸入至認證系統中，或點選該特定網站連結以證明自己的確擁有該手機門號或電子郵件帳號以完成身份認證。然而採用此種方式的認證機制並不安全。攻擊者可偽造電子郵件或簡訊，加入釣魚網站的網址，以誘騙使用者於未經意的狀況下透露自己的帳號及文字密碼。因為釣魚網站往往使用短網址服務如 goo.gl、bit.ly 等來轉址，使得使用者無法輕易察覺，也因此迫使 Google 於 2018 年 3 月關閉其 goo.gl 轉址服務 [16]。

而另一個更大的問題是使用 SMS 簡訊發送一次性密碼或網址一點都不安全 [14]。現今電話網路的底層是採用 SS7 架構。SS7 網路架構並非封閉的網路，任何一個第二類電信業者，皆可輕易存取 SS7 網路。也因此入侵者利用 SS7 網路其缺陷並攔截呼叫和 SMS，以竊取透過 SMS 簡訊所傳送的一次性密碼。更有甚者，政府部門可直接監聽使用者的通信內容以攔截一次性密碼。

因此，採用特製 App 取代 SMS 簡訊以接收認證訊息逐漸成為主流。在下一個章節中，我們將介紹 Google 所開發的兩階段認證機制 Google authenticator 所採用的基於時間之一次性密碼機制。Google 並在 GitHub 開放 Google authenticator 的原始碼供大眾使用 [10]。

而為了更進一步的保障使用者的帳號安全，Google 所參與的 FIDO 組織 [1] 提出了一系列的身分認證機制 UAF、U2F 及 FIDO 2 [4][5][9]，除了將行動裝置上的使用者認證系統用於 Web 服務外，另加入額外的硬體設備作為身分認證之用，我們將於第三章介紹。

貳、一次性密碼驗證機制

Google authenticator 為 Google 所開發之兩階段認證機制，除提供 Google 服務所需的身分認證外，亦開放給其他第三方軟體的身分認證之用。Google authenticator 使用了一種基於雜湊訊息鑑別碼 (hashed Message authentication code, HMAC) 之一次性密碼 (one-time password) 驗證機制，稱為 HOTP (HMAC-Based One-Time-Password, RFC-

4226) [1], 及基於時間的一次性密碼 (Time-based One-Time Password, TOTP, RFC-6238) [12]兩種技術，每次可以產生一個 6 位數的一次性密碼.

在使用 HOTP 機制產生一次性密碼時，驗證雙方須先產生一個金鑰 (key)，並經由安全通道傳遞到對方並協議一個初始計數器 (counter) 的內容。接著，使用以下步驟算出一次性密碼：

1. 使用 HMAC-SHA-1 演算法，計算 HS=HMAC-SHA-1(K,C) 的值。HS 字串的長度為 20 bytes (160 bits)
2. 將 HS 字串表示成 HS[0], HS[1], HS[2], …, HS[19].
   接著, 拿出 HS 字串的最後 4 個位元，即 HS[19] 的最低 4 位元。
   offset = HS[19] & 0x7F. Offset 的值為 0 – 15 之間。
3. 取出 HS 字串中，從 offset 位置開始的 4 bytes, 並去掉最高位元後，取得 31-bits 的 P.
   P = HS[offset] HS[offset+1] HS[offset+2] HS[offset+3] & 0x7FFFFFFF
4. 從 P 中取得一個 6 – 8 位數字:
   HOTP = P mod 10^x, x = 6, 7, or 8

然而在採用 HOTP 時，認證雙方除了要有同樣的金鑰 K 外，還需維持計數器 C 的同步。每次算出一個 HOTP 值後，雙方會需要將計數器的值加 1。然而一但雙方的計數器失去同步，則很難再計算出相同的 HOTP。

因此，Google authenticator 採用基於時間的一次性密碼的 TOTP 技術。TOTP 技術由主動開放認證聯盟 (Open Authentication) 所開發。TOTP 技術基於上述的 HOTP 技術，但使用系統時間來計算出計數器的值，稱為時間計數器 (Time Counter, TC)。採用 TOTP 時，認證雙方需要使用網路時間協定 (Network Time Protocol, NTP) 以維持時間的同步。TOTP 的方法如下:

1. 認證雙方協議一個初始時間值 *T0* (可直接設定為 0), 及一個時間間隔 *TS* (預設為 30 秒)。
2. 計算 $TC = \lfloor (CurrentUnixTime - T0)/TS \rfloor$，其中 CurrentUnixTime 為 Unix 中自 1970 年 1 月 1 日（00:00:00 GMT）以來的秒數。TC 則為將計算結果無條件捨去小數點以下部分，取得一整數值。
3. 計算 HS=HMAC-SHA-1(K, TC)。HMAC-SHA-1 函數的計算結果為 160 bits, 因此 HS 字串長度為 20 bytes。
4. 將 HS 字串表示成 HS[0], HS[1], HS[2], …, HS[19]。
   接著, 拿出 HS 字串的最後 4 個位元，即 HS[19] 的最低 4 位元。
   offset = HS[19] & 0x7F. Offset 的值為 0 – 15 之間。

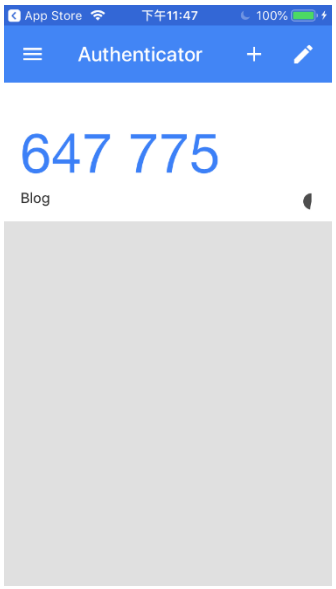
5. 取出 HS 字串中，從 offset 位置開始的 4 bytes, 並去掉最高位元後，取得 31-bits 的 P.
   P = HS[offset] HS[offset+1] HS[offset+2] HS[offset+3] & 0x7FFFFFFF
6. 從 P 中取得一個 6 – 8 位數字:
   TOTP = P mod 10^x, x = 6, 7, or 8

當一 APP 要使用 Google 之一次性密碼驗證時，使用者須先在行動裝置中安裝 Google authenticator 之 App. 接著, 驗證方需要產生一個金鑰，並由使用者手動輸入 Google authenticator 中。為了讓輸入金鑰的過程更簡易，Google authenticator 採用兩種金鑰的輸入方式。第一種為將金鑰 K 根據 base32 加以編碼。Base32 編碼使用 32 個字符 (字母 A-Z 及數字 2-7), 將金鑰每五個位元產生成一個字符。至於為何不採用常用的 Base64, 是因為 Base32 不區分大小寫，且排除了數字 “1”, “8”, “0” 等容易與字母 “I”, “B”, “O” 混淆的字。然而即使採用 base32 編碼，要使用行動裝置的虛擬鍵盤輸入一個長度較長的金鑰依舊是非常不便，因此 Google authenticator 輸入金鑰的第二種方式即為將金鑰 K 經過 base32 編碼後，以二維條碼的方式顯示在螢幕上，而使用者只要使用相機拍照二維條碼，即可輸入金鑰。如下圖中，使用者所開發的 App 可以要求使用者於 Google authenticator 中輸入 base32 編碼的金鑰，也可以直接掃描二維條碼 (該條碼內容為 otpauth://totp/Blog?secret=UKLDPUD4TMMM7KXZ).

圖二：Google authenticator 可掃描二維條碼輸入金鑰

在實際使用時，Google authenticator 設定的時間間隔 *TS* 為 30 秒，初始時間值 *T0* 則設定為 0。因此每隔 30 秒，TOTP 的值即會自動更新一次。考慮到使用者的行動裝置與驗證方的系統時間可能會有些微差距，而使用者讀取到 TOTP 並輸入網頁驗證時也有些微的時間差距。因此在驗證方檢查使用者輸入的 TOTP 值時，除了以目前時間計算出的$TC = \lfloor (CurrentUnixTime - T0)/TS \rfloor$ 來計算 HS=HMAC-SHA-1(K, TC) 外，尚須考慮 HMAC-SHA-1(K, TC-1) 及 HMAC-SHA-1(K, TC-2) 等先前的 TOTP 數值，而此時間寬容度則由驗證方決定。若驗證方容許五分鐘的時間誤差，則需計算出 10 個

TOTP 值供比對。但是當時間寬容度越高則較容易遭受到攻擊，而時間寬容度越低則越可能造成使用者的不便。。



圖三：Google authenticator 所產生的一次性密碼

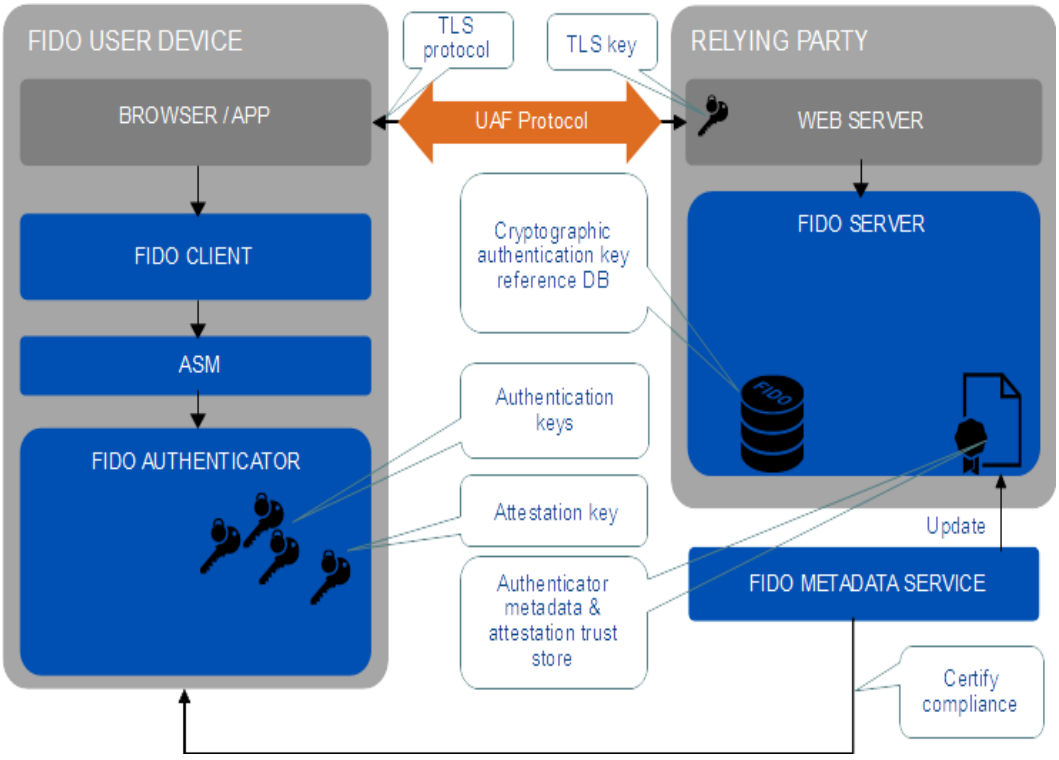
參、FIDO 的身分認證機制

在使用 Google authenticator 認證時，使用者於身分驗證時必須開啟行動裝置的 Google authenticator 程式以取得一個一次性密碼，接著在於網頁認證時輸入該密碼。雖然安全，但是過程卻費力及費時。為了更進一步加強身分認證機制的安全，西元 2014 年底由 Google 等逾 150 家業者所共同組成的線上快速身份認證聯盟 FIDO (Fast IDentity Online) Alliance [1]發表 FIDO 1.0 標準[7]，並於 2017 年公佈 FIDO 2.0 標準 [9]。

FIDO 1.0 包含了兩種認證協定，分別為通用認證框架 (Universal Authentication Framework，UAF) [4] 與通用第二因子認證 (Universal 2nd Factor，U2F)[5]。在 UAF 標準中，為利用生物辨識或其他第二因子認證方法以連結使用者與裝置。使用者只需於第一次使用(如系統開機時)，輸入一次密碼後，以後僅需使用第二因子認證，不再需要使用密碼，如此可降低密碼遭受釣魚攻擊的威脅。而通用第二因子認證 U2F 標準則需使用一個額外的令牌(token)裝置當成身分認證的第二因子。每次認證時除了需鍵入密碼外，亦須提示第二因子之令牌方可通過認證。即使密碼被盜取，攻擊者因為無法取得第二因子之設備，因此無法通過身分認證。採用通用第二因子認證 U2F 時，因密碼的重要性較低，因此使用者可以設定安全度較低的密碼，如四位數的數字 PIN 碼。不論是使用 UAF 或是 U2F 認證，於認證過程中皆需確保使用者親自參與認證過程。例如在

UAF 中，使用者必須輸入自己的指紋; 而在 U2F 中，使用者必須碰觸所使用的認證裝置上的按鈕。以下我們將分別介紹 UAF、U2F 及 FIDO2 機制。

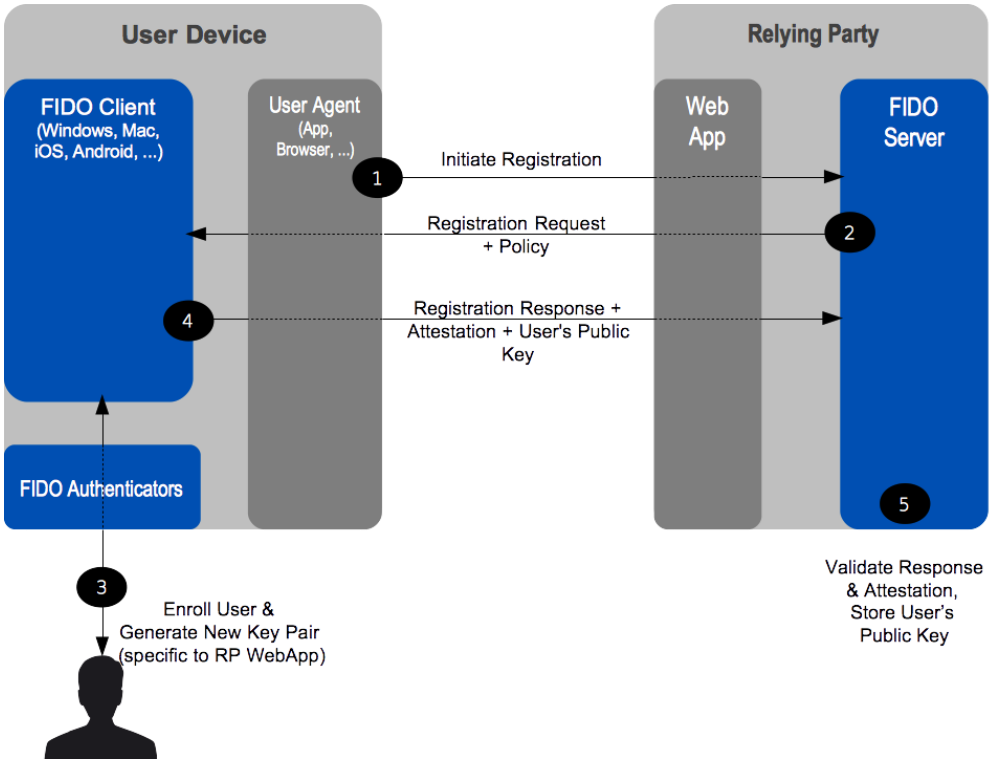
3.1 通用認證框架 (Universal Authentication Framework，UAF)



圖四：FIDO 通用認證框架 UAF [4]

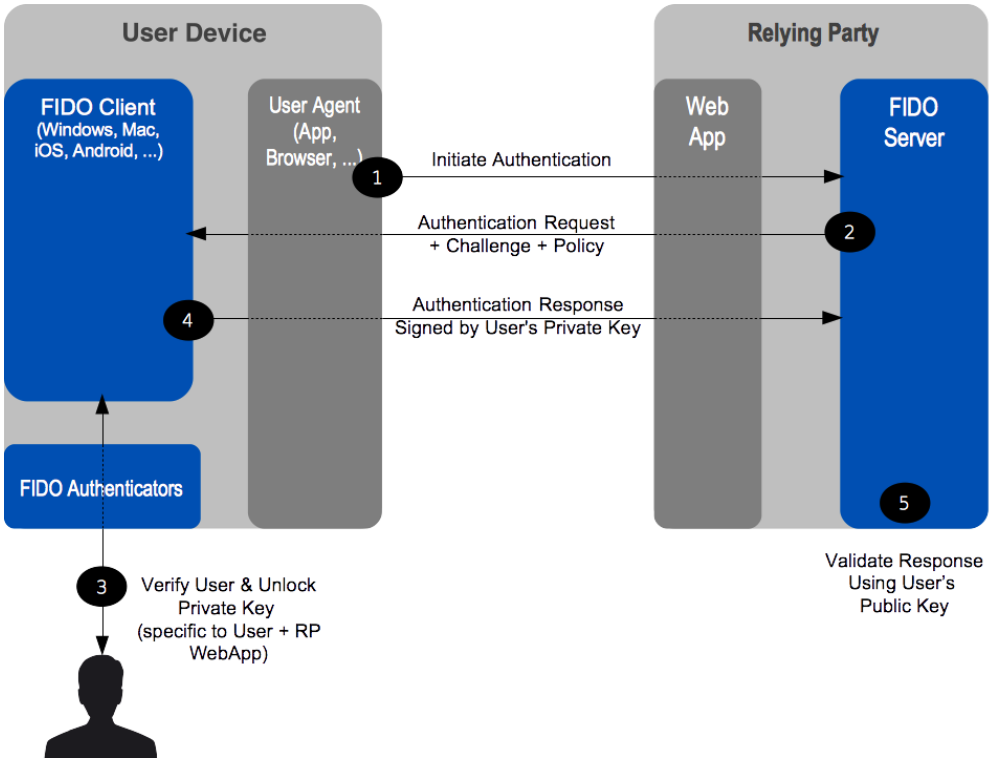
在 UAF 架構下包含兩個主體，分別為使用者用來認證的設備 (FIDO User Device) 及使用者所要存取的線上服務 (Relying Party)，如圖四所示。於使用者設備中，需安裝 FIDO 客戶端 (FIDO Client) 用於存取使用者設備所提供的身分驗證服務，稱為 FIDO authenticator (如指紋或 Face ID 解鎖)，而在線上服務中，亦須安裝一 FIDO 伺服器 (FIDO Server)用以儲存使用者的設備資訊。

當使用者需登入線上服務時 (圖五之步驟 1)，線上服務會使用 UAF 協定呼叫 FIDO 客戶端(圖五之步驟 2)，再由 FIDO 客戶端要求使用者通過本地驗證(圖五之步驟 3) 若使用者通過認證，則會產生一組屬於特定線上服務的公開金鑰，並將其中的私鑰儲存於行動裝置的安全元件中。隨後 FIDO 客戶端會將該金鑰對中的公開金鑰藉由線上服務回傳至 FIDO 伺服器(圖五之步驟 4)。FIDO 伺服器驗證後，隨即將該公開金鑰儲存於伺服器中(圖五之步驟 4)。對所有資訊傳遞過程皆採用 TLS 加密。



圖五：FIDO UAF 註冊程序[4]

當使用者完成註冊程序後，在隨後的登入過程中，FIDO 伺服器將會隨機產生一個亂數 Challenge 傳至使用者設備的 FIDO 客戶端中(圖六之步驟 2)。隨後由 FIDO authenticator 發起使用者的認證程序。當使用者認證成功後，FIDO authenticator 將會使用相對於該線上服務的私密金鑰，將 Challenge 簽名後回傳至 FIDO 伺服器(圖六之步驟 4)，然後 FIDO 伺服器將以使用者的公開金鑰加以驗證(圖六之步驟 5)。



圖六：FIDO UAF 驗證程序[4]

3.2 通用第二因子認證協定 (Universal 2nd Factor，U2F)

在 U2F 兩階段身分認證協定中，包含三個主體。除了與 UAF 協定相同的使用者用來認證的設備 (FIDO User Device) 及使用者所要存取的線上服務 (Relying Party) 外，使用者需要事先註冊一個強第二因子(strong second factor) 的外接設備作為身分識別之用。外接設備可使用 USB, NFC 或是藍芽連接至使用者的設備 [2][3][6]。

在註冊程序中，此外接設備將會產生一組公開金鑰對，並將公開金鑰傳遞至 FIDO 伺服器的資料庫中。而在後續使用 U2F 協定時，於第一階段的身分認證的過程中，使用者需輸入使用者代號及密碼。之後，驗證端可以於任意時刻發起第二階段身分認證，要求使用者出示所擁有的外接設備。此時，使用者必須將外接設備插入 USB，按下外接設備上的按鈕 [6]; 或是採用 NFC 介面時，則需將外接 NFC 裝置靠近使用者的裝置 [3]。與 UAF 協定相同，所有資訊傳遞過程皆採用 TLS 加密。

U2F 與 UAF 不同在於需使用一個外接設備作為身分認證之用，亦即以外接的 U2F 裝置取代 UAF 協定中的 FIDO authenticator。使用者於註冊階段時，由外接裝置產生一組公開金鑰對及一個計數器，並將公開金鑰及計數器傳遞至 FIDO 伺服器端。而認證階段時，FIDO 伺服器將會產生一個亂數 nonce 傳遞至外部 U2F 裝置，U2F 裝置將對該亂數以相對應的私密金要簽署後回傳至 FIDO 伺服器，於過程中亦會一並更新計

數器的值。若有惡意的第三者可以複製使用者擁有的 U2F 裝置，也會因為在認證過程中無法取得一致的計數器值，導致驗證失敗。

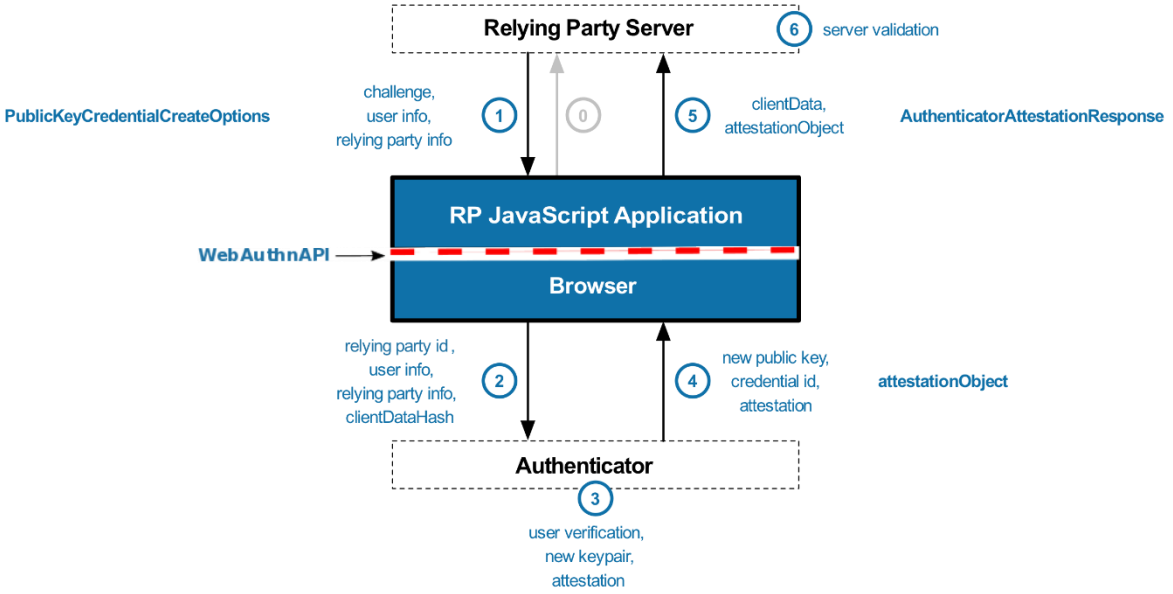
為了減少硬體成本的負擔，U2F 設備並為綁定特定的使用者。亦即不同的使用者，可以使用同一個 U2F 裝置作為身分認證之用。相對的，每個使用者亦可以註冊一個以上的 U2F 裝置，於身分認證時，只需使用其中一個裝置即可。此外，為了防止 U2F 設備被複製，U2F 裝置於產生公開金鑰對時，也會一並產生一個計數器。而認證階段時，會將更新後的計數器

3.3 FIDO 2 協定

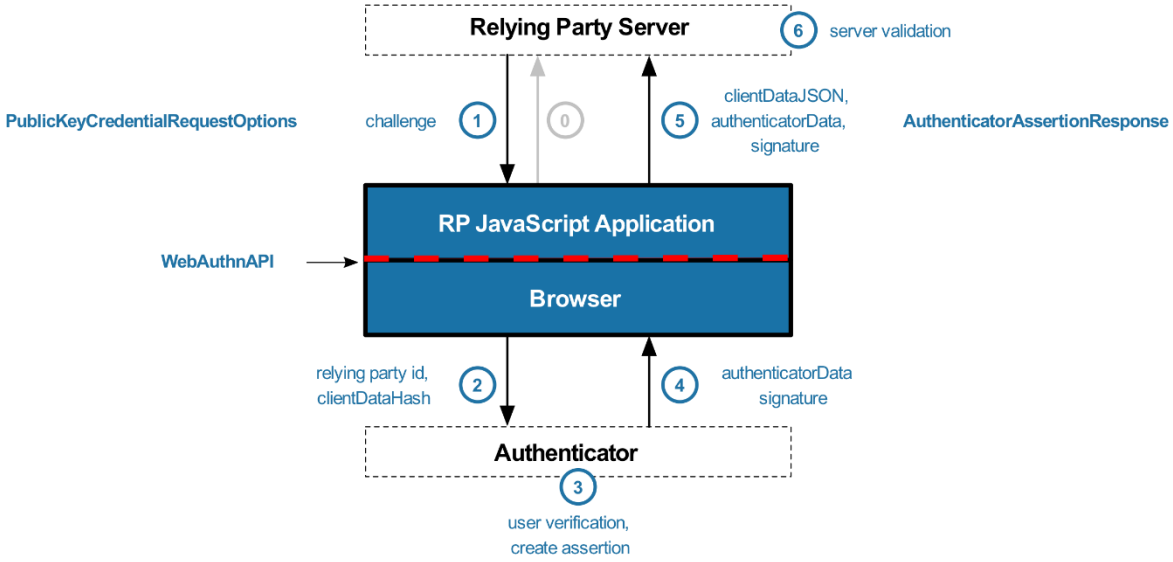
3.2 節中所提之 U2F 協定是由 Google 及 Yubico 所共同研發，並由 Google 員工所測試後再提至 FIDO 成為標準。而於 2017 年，基於 U2F 協定更研發出 FIDO 2 標準[9]。在 U2F 協定中，需要兩階段的驗證，即第一階段的密碼驗證及第二階段的外接設備驗證。Google 開發了名為 Titan 的外接式安全密鑰設備[15]，並從 2017 年開始在內部測試。Google 並於 2018 年七月開始正式對外販售 Titan，並提供 USB 及 USB/藍牙/NFC，如圖七所示。而 FIDO 2 則為 U2F 的無密碼 (Passwordless) 延伸版本，可省略第一階段的密碼驗證過程，只用 U2F 外接設備驗證 [18]。

FIDO2 標準為包含兩個部分，webauthn [17] 及 Client To Authenticator Protocol (CTAP2) [8]. FIDO2 將原先 U2F 中的 web API 加以延伸，訂定為 webauthn 並取得 W3C 的認可。新的 webauthn 同時相容於 U2F 及 FIDO2 。在 webauthn 標準中制定了 WebAuthnAPI，作為行動裝置端 (Browser) 與伺服器端 (Relying Party Server) 間通訊的標準，包含使用者註冊及使用者認證兩部分，如圖八及圖九所示。在原先的 U2F 協定中，與外部設備通訊的協定(client-side protocol)，改稱為 CTAP1。FIDO 2 提出延伸的版本 CTAP2 [8], 除了 CTAP 1 中所定義的 USB, 藍牙及 NFC 外，可支援更多的外部設備，如另一隻手機，或 IC 卡等。

圖七：Google Titan 安全密鑰[15]



圖八: FIDO2 使用者註冊流程 [17]



圖九: FIDO2 使用者認證流程 [17]

肆、結論

為了保障使用者的帳號安全，各個雲端服務廠商紛紛提出兩階段身分驗證機制。Google 將其所開發的兩階段身分驗證機制免費開放給大眾使用。FIDO 組織更進一步地提出使用額外認證裝置的 U2F 及 FIDO 2 標準。Google 於今年八月公開販售其所開發符合 U2F 協定的 Titan 安全密鑰裝置，並建議如政治人物、公司主管和媒體等使用

Titan 以保障其帳號安全，甚至建議敏感的政治人物應使用兩個以上的安全密鑰裝置。然而，需要額外攜帶硬體設備不免降低大眾使用的意願。如何在使用者的便利性與安全性之間取得平衡點，並開發出便於使用的兩階段驗證機制，則有待於大家的努力了。

參考文獻

[1] FIDO Alliance, https://fidoalliance.org (2018/10/12).
[2] FIDO Bluetooth® Specification v1.0, https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-bt-protocol-v1.2-ps-20170411.html (2017/04/11).
[3] FIDO NFC Protocol Specification v1.0, https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-nfc-protocol-v1.2-ps-20170411.html (2017/04/11).
[4] FIDO UAF Architectural Overview, https://fidoalliance.org/specs/fido-uaf-v1.1-ps-20170202/fido-uaf-protocol-v1.1-ps-20170202.html (2017/02/02).
[5] FIDO Universal 2nd Factor (U2F) Overview, https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html (2017/04/11).
[6] FIDO U2F HID Protocol Specification, https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-hid-protocol-v1.2-ps-20170411.html (2017/04/11).
[7] FIDO 1.0, https://fidoalliance.org/fido-1-0-specifications-published-and-final/ (2014/12/9).
[8] FIDO 2.0: Client To Authenticator Protocol, https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-client-to-authenticator-protocol-v2.0-rd-20170927.html (2017/09/27).
[9] FIDO 2.0: Overview, https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-overview-v2.0-rd-20170927.html (2017/10/4).
[10] Google authenticator open source, https://github.com/google/google-authenticator (2017/12/17).
[11] RFC 4226, HOTP: An HMAC-Based One-Time Password Algorithm, https://tools.ietf.org/html/rfc4226 (2018/10/12).
[12] RFC 6238, TOTP: Time-Based One-Time Password Algorithm, https://tools.ietf.org/html/rfc6238 (2018/10/12).
[13] A. Roy, N. Memon and A. Ross, "MasterPrint: Exploring the Vulnerability of Partial Fingerprint-based Authentication Systems," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2013-2025, 2017.
[14] SS7 Vulnerabilities and attack exposure report 2018, https://www.gsma.com/membership/ss7-vulnerabilities-and-attack-exposure-report-2018/ (2018/10/12).
[15] Titan Security Key, https://cloud.google.com/titan-security-key/ (2018/10/12).

[16] Transitioning Google URL Shortener to Firebase Dynamic Links, https://developers.googleblog.com/2018/03/transitioning-google-url-shortener.html (2018/3/30).
[17] Web Authentication: An API for accessing Public Key Credentials Level 1, https://www.w3.org/TR/webauthn/ (2018/8/7).
[18] What is FIDO2, https://www.yubico.com/2018/05/what-is-fido2/ (2018/5/24).

[作者簡介]

羅嘉寧博士為國立交通大學資訊工程系博士，現為銘傳大學電腦與通訊工程學系副教授及中華民國資訊安全學會理事。其主要研究專長為網路安全，物聯網系統及機器人控制。