

國立雲林科技大學資訊工程系

碩士論文

Department of Computer Science and Information Engineering
National Yunlin University of Science & Technology
Master Thesis

應用零信任與 FIDO 技術於內網安全之身份驗證機制
The Application of Zero Trust and FIDO Techniques for
Authentication Mechanism in Intranet Security

黃偉恩

Wei-En Huang

指導教授：郭文中 博士

Advisor : Wen-Chung Kuo, Ph.D.

中華民國 112 年 6 月

June 2023

國立雲林科技大學
碩士班學位論文考試委員會審定書
National Yunlin University of Science and Technology
Thesis Oral Defense Approval Form

本論文係 黃偉恩 君在本校 資訊工程系碩士班 所提論文 應用零信任與FIDO技術於內網安全之身份驗證機制 碩士資格水準，業經本委員會評審認可，特此證明。
The student HUANG, WEI-EN enrolled in the Master's program in Department of Computer Science and Information Engineering has satisfactorily passed the oral defense of the thesis The Application of Zero Trust and FIDO Techniques for Authentication Mechanism in Intranet Security.

口試委員：
Oral defense
committee members

郭文中

郭文中

林峻立

林峻立

林易泉

林易泉

指導教授：
Advisor(s)

郭文中

郭文中

所長：
Dean of Graduate
Institute

郭文中

中 華 民 國

112 年

7 月 28 日

摘要

近年來，駭客對網路世界的威脅日益嚴重，各大企業也陷入了資安防護的考驗。在常見的資訊系統中，密碼扮演著重要的角色，但也是最容易被駭客攻擊並竊取。一旦駭客竊取了密碼，就可以偽裝成合法身份進入企業網路環境並進一步採取橫向移動，持續破壞更多的企業基礎設施。

為減輕基於密碼登入系統的攻擊風險，使得無密碼登入機制應運而生。在 FIDO 聯盟訂定的標準協定下，提供企業無密碼驗證機制，也就是說，企業伺服器端只需儲存公鑰，而不是以往儲存使用者密碼，將可降低企業伺服器遭破壞的風險。此外，本論文透過手機作為漫遊驗證器進行生物辨識，手機端只在需要驗證時才會提出私鑰進行驗證，然而無密碼登入機制並無法完全防止駭客攻擊。因此，本論文在企業內網中採用零信任的網路架構，僅在通過驗證後才開啟專屬動態隧道，並持續驗證使用者身份。使用者只能透過隧道存取目的資源，以最小化降低資安風險。經過模擬實作後，我們所提出的涵蓋 FIDO 和零信任兩大概念之身分認證機制，不僅去除密碼遺失帶來的風險，也讓企業內網具有更好的對抗性。

關鍵字：零信任、FIDO、無密碼登入

ABSTRACT

In recent years, the threat of hackers to the cyber world has become increasingly serious, putting major corporations under the test of cybersecurity defense. In common information systems, passwords play a crucial role that is also the easiest to be attacked and stolen by hackers. Once a hacker steals a password, he can impersonate a legitimate identity, infiltrate the corporate network environment, and further adopt lateral movement to continually damage more infrastructure.

To mitigate the risk of attacks for the password login system, passwordless login mechanisms have emerged. The standard protocols are proposed by the FIDO Alliance, it can provide the passwordless authentication mechanisms for companies. In this setup, the enterprise server needs only to store the public key, instead of storing user passwords as was done in the past, reducing the risk of enterprise server compromise. Additionally, it will implement biometric recognition through phones as roaming authenticators in the thesis. The phone only presents the private key for authentication when it needed. However, passwordless login mechanisms cannot completely prevent hacker attacks. Therefore, we will adopt a zero-trust network architecture within the corporate intranet, where a dedicated transient tunnel is opened only after passing authentication, and user identity continues to be verified. Users can only access the targeted resources through the tunnel, minimizing and reducing cybersecurity risks. After simulation implementation, the identity authentication mechanism is proposed, encompassing both FIDO and zero-trust concepts, not only eliminates the risks brought about by password loss but also enhances the resilience of the corporate intranet in the thesis.

Keywords: Zero Trust, FIDO, Passwordless

誌謝

在這份論文中，我希望能將我最深的感謝獻給幾位對我產生重大影響的人。首先，我要感謝我的指導教授，郭文中老師。您的熱情和耐心無疑為我開闢了新的視野，使我能夠在學術上取得突破。您對我不厭其煩的指導和鼓勵，使我能夠專注於研究，並克服許多困難。我由衷感謝。

接著，我要感謝黃玉枝老師，您對我學術上的挑戰和鞭策，使我不斷提升自己，並學會思考問題的多元面向，讓我在學術路途中取得新的理解和洞見。我深深地感謝您的關心和支持。

最後，我要感謝我的同學翁瀟嶸、才揚、張景翔、蔡惟惇、黃羿翔，同時也感謝其他實驗室同學們的支持，我們一起討論、一起學習、一起克服難題，這種經驗對我來說是無比寶貴的，每天一起待到晚上討論論文，使我在面對困難時不會感到孤單。我們的共同努力使我們在學術上取得了許多重大的突破，這些都是我將會珍藏的回憶。

感謝你們所有人的付出和努力，這份論文的完成，有賴於你們的支持與鼓勵。這不僅是我的成就，也是我們共同的成就。再次，我要對你們表達我最深的感謝和敬意。

目錄

摘要	i
ABSTRACT	ii
誌謝	iii
目錄	iv
表目錄	vi
圖目錄	vii
一、緒論	1
二、相關研究	4
2.1 FIDO	4
2.2 橢圓曲線加密演算法	6
2.3 橢圓曲線數位簽章算法	8
2.4 進階加密標準	9
2.5 零信任簡介	11
2.6 零信任架構	11
2.7 零信任網路	13
2.8 容器化技術	14
2.9 傳統網路安全疑慮	15
2.10 傳統密碼系統安全疑慮	17
2.11 小結	18
三、研究方法	19
3.1 系統概述	19
3.2 FIDO 認證機制	20
3.3 暫態專屬隧道(Temporary Isolated Tunnel)	25
3.4 SSH Tunneling	27
3.5 零信任架構	28
四、實驗結果	31
4.1 實驗軟體規格	31
4.2 實驗環境	31
4.3 實驗模擬環境之註冊流程	32
4.4 實驗模擬環境之登入流程	34
4.5 實驗模擬環境之持續驗證流程	38

4.6 攻擊情境	40
4.7 相關論文比較	41
五、結論與未來展望	42
5.1 結論	42
5.2 未來展望	43
參考文獻	44



表目錄

表 1 ECC 和 RSA 金鑰長度對應安全性[12]	6
表 2 ECDSA 參數對照表	8
表 3 零信任部署方式比較表	19
表 4 FIDO 流程參數對照表	21
表 5 相關論文比較表	41



圖目錄

圖 1 FIDO 身份驗證架構[2]	5
圖 2 AES 加密[12]	10
圖 3 AES 解密[12]	10
圖 4 零信任存取[10]	11
圖 5 核心零信任邏輯元件[13]	13
圖 6 Docker 與虛擬機區別[11]	15
圖 7 傳統網路架構圖	17
圖 8 零信任與 FIDO 安全架構圖	20
圖 9 FIDO 註冊	23
圖 10 FIDO 登入—手機與伺服器通訊	24
圖 11 FIDO 登入—手機、電腦和伺服器通訊	25
圖 12 暫態專屬隧道前置建立工作	25
圖 13 暫態專屬隧道存取內網資源流程圖研究成果	26
圖 14 持續驗證流程	27
圖 15 SSH Tunneling 應用	28
圖 16 在登入情境下零信任運作過程	29
圖 17 持續驗證判斷流程圖	30
圖 18 模擬實驗環境	31
圖 19 APP 初始頁面	32
圖 20 APP 註冊初始頁面	32
圖 21 APP 註冊頁面輸入資料	33
圖 22 APP 註冊頁面進行生物辨識	33
圖 23 APP 註冊頁面生物辨識驗證成功	33

圖 24 APP 註冊頁面生物辨識驗證失敗	33
圖 25 FIDO 伺服器生成 128bits 挑戰碼	34
圖 26 使用者手機在本機生成金鑰	34
圖 27 使用者訊息存入資料庫	34
圖 28 電腦代理程式初始頁面	35
圖 29 電腦代理程式藍芽伺服器開啟	35
圖 30 APP 初始頁面	35
圖 31 APP 選擇藍芽配對	35
圖 32 電腦代理程式藍芽伺服器接收資料	36
圖 33 APP 登入頁面	36
圖 34 電腦代理程式接收 token	37
圖 35 電腦代理程式前端頁面	37
圖 36 使用者首次存取內網庫存系統網頁	37
圖 37 電腦端管理員登入	38
圖 38 持續驗證頁面	38
圖 39 持續驗證之成功頁面	39
圖 40 持續驗證之失敗頁面	39
圖 41 電腦端普通使用者登入	39
圖 42 攻擊者使用 Nmap 嗅探內網	40
圖 43 攻擊者使用 Nmap 嗅探開發伺服器	41

一、緒論

隨著科技的演進，資訊已經成為生活中不可缺少的一個部分。導致在日常生活中經常會使用到許多資訊系統，使用者必須記住複雜的密碼並且符合密碼安全設定原則。而在資訊系統中身份驗證扮演十分重要的角色，決定一個系統的安全性，身份驗證又可分為三大類，使用者知道的(Something you know)、使用者擁有的(Something you have)、使用者有的特徵(Something you are)。為考量方便性和安全性間的平衡，常見資訊系統通常以密碼為辨識使用者的重要依據，只有擁有密碼就能完全代表這個數位身份。

在現今高度數位化的時代，駭客看準密碼是作為資訊系統中的弱點並加以破解攻擊，導致許多資安事件都與駭客竊取盜用密碼有關。一般使用者往往在密碼管理存在許多的風險，例如將密碼寫在小紙條上，任何有心人都能將密碼記住並竊取該使用者的數位身份。駭客擁有密碼就能在數位世界中完全取代合法使用者，可見現有的密碼機制已經存在嚴重的安全問題。於是FIDO(Fast IDentity Online)聯盟發布的白皮書[1][2][3]中闡述無密碼登入機制帶來的好處和如何實作。

目前已有學者投入研究探討 FIDO 帶來的優缺點，Lyastani 等人[4]和 Lin 等人[5]著手分析 FIDO 與傳統驗證方式之優缺點和使用者接受度評估，研究指出 FIDO 無密碼認證機制更實用更容易被接受，減少伺服器外洩帶來的風險和使用者不需要再記住複雜的密碼。FIDO 認證透過基於公鑰密碼學技術，伺服器不再需要儲存使用者的密碼，僅儲存使用者的公鑰，即便伺服器遭受入侵，公鑰被竊取並不會造成嚴重的危害性。

FIDO 認證主要利用生物特徵進行身份驗證，善用使用者的手機當作漫遊認證器，並通過藍芽傳輸將驗證後的令牌傳送至使用者電腦，使用者就能安全的進行存取內網資源，就能透過增加便利性同時降低駭客濫用密碼的風險。目前已有多个研究利用 FIDO 機制實際用於系統中，如導入 FIDO 機制運用於校園系統[7]

和 FIDO 機制之文件共享系統[8]。

於是我們利用 FIDO 機制來進行本論文系統中的驗證，使用者在登入時，將透過手機收集生物特徵進行驗證，通過驗證後使用者的手機將會收到從伺服器端發出的挑戰碼，然後使用其私鑰簽名並傳送回伺服器，得到令牌後使用者可以透過藍芽將令牌傳送至電腦，而這種近場連接使得駭客難以盜取令牌。為了實現持續驗證的概念，除了在首次登入時收集使用者的生物特徵進行驗證外，每當使用者嘗試存取敏感資料時，手機端會再次要求進行生物特徵驗證。此外伺服器會將連線過程完整記錄下來。

再來分析傳統網路既有之安全風險，在過去的資訊安全防護大多以護城河概念進行，主要檢查從外部網路進入內部網路的流量，而內部網路之間的流量並未受到嚴格管控。此外只要通過一次身份驗證，系統就會認定該身份為合法使用者，並允許其存取所有內部網路資源。然而從駭客的角度來看，只要冒用合法身份，就可威脅內部網路的所有資源。為了解決這個問題，我們提出研究旨在改善傳統的 VPN 和密碼登入系統，並參考 Emin Huseynov 提出的方法[9]，將傳統 VPN 與 FIDO 機制相結合，以取代密碼驗證。並且我們落實連線權限最小化，以更嚴格的方式驗證每筆請求，進一步利用 FIDO 機制提供持續驗證的機制。我們透過動態生成使用者與資源之間的連線來實現權限最小化，並嚴格監控使用者存取資源的過程。一旦發現可疑行為，系統會立即要求使用者再次驗證。

結合上述所說，利用 FIDO 和零信任建構安全的內網環境並有效控制連線情況，最後我們模擬一個傳統的企業內部網路，並在網路基礎設施中部署了 VPN 伺服器。將模擬在 VPN 憑證外洩的情況下，傳統網路環境可能會遇到的駭客攻擊，並比較傳統網路架構和本文系統架構的防禦成效，實驗結果顯示，本論文網路架構可以有效地防止駭客冒用合法身份進行攻擊。

本文分為六章節：第一章節說明研究動機與目的；第二章相關研究；第三章詳細描述研究方法；第四章研究結果與相關論文比較；第五章結論與未來發展；

最後詳列參考文獻。



二、 相關研究

2.1 FIDO

FIDO 聯盟成立的宗旨在於解決強制認證設備互動性，以及用戶面臨大量複雜用戶名和密碼的困擾。該組織旨在推廣更安全且更便捷的身份驗證方法，降低對密碼的依賴，為了實現這一目標，FIDO 聯盟制定了一系列開放標準和解決方案，在 FIDO1 主要推出了兩個標準：FIDO UAF (Universal Authentication Framework) 和 FIDO U2F (Universal 2nd Factor)，目前已經發展到 FIDO2，FIDO2 相較於前一代更具體，主要由 WebAuthn 和 CTAP 組成，屆時使用者都能透過現有裝置如 Windows、IOS、Android、主流瀏覽器中使用 FIDO 驗證，將無密碼化的願景推進一大步[1-3]。

在驗證器選擇也提供兩種方式，分別是 Roaming Authenticators 和 Platform Authenticators，雖然對於 FIDO 標準來說兩種驗證器都扮演相同的角色，但是對於使用者而言，使用情境卻截然不同。

在提高安全性和用戶體驗方面，FIDO 的優勢包括(1)它通過非對稱加密技術提高安全性；(2)採用生物辨識使身份驗證過程更為便捷，提高了用戶滿意度。不過在實際應用 FIDO 標準時，我們仍然需要面對一些挑戰和限制，例如生物辨識技術的準確性和可靠性問題可能會對 FIDO 標準的廣泛應用構成障礙，同時設備兼容性也是阻礙 FIDO 標準發展的一個因素。

如圖 1 所示，可以看到 WebAuthn、CTAP 各自在 FIDO 架構中扮演的角色，還有與 RP APP Server 和 FIDO Server 的傳輸流程。

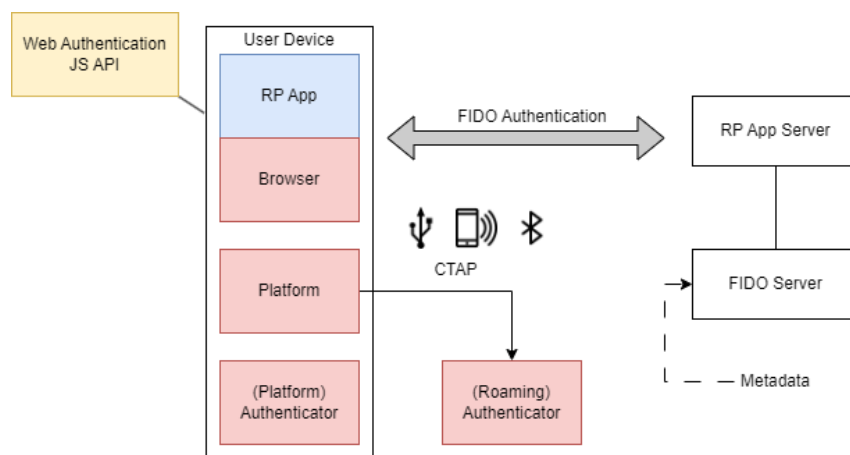


圖 1 FIDO 身份驗證架構[2]

2.1.1 FIDO UAF (FIDO Universal Authentication Framework)

FIDO UAF 是一個用於建立無密碼身份驗證解決方案的框架，通過生物辨識技術（如指紋、臉部識別等）或其他本地驗證方式實現用戶身份驗證。

2.1.2 FIDO U2F (FIDO Universal Second Factor)

FIDO U2F 則是一個簡單的雙因素身份驗證解決方案，讓用戶在輸入密碼後使用物理安全密鑰（如 YubiKey）進行二次驗證。

2.1.3 WebAuthn (Web Authentication)

以 API 的形式呈現，開發者可以輕鬆調用 API 實現 FIDO 驗證，各大主流系統和瀏覽器都有內建支援。

2.1.4 CTAP(Client to Authenticator Protocol)

這是將一代的 U2F 進行改良，在 FIDO1 中使用 U2F 往往需要專用硬體，在成本與實用性上缺乏優勢，而 CTAP 是一種傳輸協定，使用者僅需要使用現有裝置如手機內建硬體，並搭配手機藍芽、NFC、USB 進行認證，這種方式就要透過 CTAP。

2.1.5 Roaming Authenticators

Roaming Authenticators 指的是透過 USB、藍芽或是 NFC 等協議連線到客戶端設

備，也就是充當身份認證器的角色，並且還能允許使用者為多個客戶端設備認證。

2.1.6 Platform Authenticators

Platform Authenticators 本身就是客戶端設備，常見於具有生物辨識的裝置上或是有 TPM 晶片之裝置，不須再透過網路傳輸而是直接在設備本身上驗證並使用系統。

2.1.7 RP App Server (Relying Party Application Server)

指的是需要驗證的服務，可能是一個網站或是應用程式，提供使用者註冊和認證，也就是後端伺服器的角色。

2.1.8 FIDO Server

為身份驗證過程的核心，負責處理由 RP App Server 發送的身份驗證請求，並驗證該請求。當使用者註冊新設備時，FIDO Server 會儲存公鑰以用於後續的身份驗證。當用戶嘗試登錄時，FIDO Server 會使用這個公鑰來驗證使用者的身份。

2.2 橢圓曲線加密演算法

橢圓曲線加密演算法 (Elliptic Curve Cryptography, ECC) 這個演算法是由 Neal Koblitz 和 Victor S. Miller 在 1987 年提出[10]，是一種基於橢圓曲線數學的公鑰加密演算法，以點座標進行運作，在此以 Secp256k1[11]為例作為介紹，在曲線 $y^2 = x^3 + 7$ 上，且有一模數來限制在有限域內。橢圓曲線加密演算法提供比 RSA 更小的計算量但有一樣的安全性，ECC 已被廣泛應用於現代加密系統中，包括 SSL/TLS 協議、移動通訊系統、智能卡、物聯網等領域。

表 1 所示，相同安全性強度的情況下，ECC 所需金鑰長度更短，故本篇論文採用 ECC 為加密演算法。

表 1 ECC 和 RSA 金鑰長度對應安全性[12]

ECC 金鑰長度 (位元)	RSA 對應金鑰長度 (位元)	安全性強度 (位元)
160	1024	80

定義是滿足 $y^2 = x^3 + ax + b$ 形式的平面曲線，給定 a 和 b 是實數，即給定

兩個點 P 和 Q 在橢圓曲線上，定義兩個點的「加法運算」。選擇兩個點 P 和 Q ，並畫一條通過這兩點的直線。這條直線將會與曲線相交，稱這個點為 R 。然後對 R 進行"反射"操作，即將其翻轉到曲線的另一邊，將這個新的點稱為 S 。在這種情況下，點 P 和點 Q 的和為點 R ，即 $P+Q=R$ 。

本文採用 Secp256k1 這條曲線，其函式 a 參數為 0 和 b 參數為 7， $y^2 = x^3 + 7$ ，而 p 是曲線的純量模數，限制在一個有限域內，橢圓曲線加法大致可以分為兩種情狀，第一種情況是兩個點相異座標相加如式(1-3)和第二種情況兩個點座標重疊如式(4-6)，並且使用 ElGamal 加密演算法進行訊息加密。

第一種兩個點相異座標相加：

$$\lambda = (y_2 - y_1)/(x_2 - x_1) \bmod p \quad (1)$$

$$x_3 = \lambda^2 - x_1 - x_2 \bmod p \quad (2)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \bmod p \quad (3)$$

第二種情況兩個點座標重疊：

$$\lambda = (3x_1^2 + a)/(2y_1) \bmod p \quad (4)$$

$$x_3 = \lambda^2 - x_1 - x_2 \bmod p \quad (5)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \bmod p \quad (6)$$

2.2.1 金鑰生成

選擇一橢圓曲線 E_C ，並且訊息接收方選擇一個點座標 e_1 和私鑰 d ，並且計算 $d \times e_1$ 等於 e_2 ，訊息接收方則將 E_C 、 e_1 、 e_2 公開發布。

2.2.2 訊息加密

訊息發送方加密明文訊息 m ，並且將訊息 m 對應到橢圓曲線上一點 P ，然後訊息發送方選擇一隨機數 r ，計算 $C_1 = r \times e_1$ ，接著再計算 $C_2 = P + r \times e_2$ ，最後將 C_1 和 C_2 發送給訊息接收方。

2.2.3 訊息解密

訊息接收方收到 C_1 和 C_2 ，並且計算 $P = C_2 - (d \times C_1)$ ， P 就是明文訊息對應的

點座標。

2.3 橢圓曲線數位簽章算法

橢圓曲線數位簽章算法(Elliptic Curve Digital Signature Algorithm, ECDSA)是一種非對稱加密算法，它基於 ECC 是公開金鑰加密算法的一種。因此在數位簽章和身份驗證等應用中被廣泛使用。算法安全性建立在橢圓曲線加密演算法上，ECDSA 利用橢圓曲線的離散對數問題來實現加密和解密，這使得攻擊者無法從公開的加密訊息中推斷出私鑰，從而保護了資料安全性。

ECDSA 使用一對公鑰和私鑰來進行數字簽名和驗證，私鑰是由一個大的隨機整數生成的，而公鑰是由私鑰和一個橢圓曲線上的點計算而得，以 Secp256k1 為例，方程式是 $y^2 = x^3 + 7$ ，這個曲線上的點是 (x, y) ， x 和 y 都是 256 位元的十六進位字串，且需滿足方程式，如表 2。

表 2 ECDSA 參數對照表

參數	說明
E_C	所選擇之橢圓曲線並選定在有限域 C
n	級數 order
G	基點
Q	long term 公鑰
d	long term 私鑰
k	short term 私鑰
(s, r)	訊息簽章值
$h(M)$	訊息 M 之雜湊值
(u_1, u_2)	簽章驗證參數

1. 金鑰產生

訊息發送方選擇一個隨機數 d ，並利用式(7)算出座標 Q ， Q 則是代表簽章公鑰簽章產生

$$d \times G = Q \quad (7)$$

2. 簽章產生

選一個隨機數 k ，利用式(8)求出 K ，以 K 的 x 軸，稱之為 x_i ，利用式(9)求出 r ，若 r 為0則需要重新選擇隨機數 k

$$k \times G = K = (x_i, y_i) \quad (8)$$

$$r = x_i \bmod n \quad (9)$$

$$s = k^{-1} \cdot (h(M) + d \cdot r) \bmod n \quad (10)$$

訊息發送方將訊息 M 進行雜湊得 $h(M)$ ，以式(10)計算出訊息簽章值，最後將訊息簽章值 (s, r) 傳送到訊息接收方。

3. 簽章驗證

$$u_1 = h(M) \cdot s^{-1} \bmod n \quad (11)$$

$$u_2 = r \cdot s^{-1} \bmod n \quad (12)$$

$$u_1 \times G + u_2 \times Q \stackrel{?}{=} k \times G \quad (13)$$

簽章接收方會收到訊息簽章值 (s, r) ，並另外計算出訊息 M 的雜湊值 $h(M)$ ，以式(9)和式(10)計算驗證參數，得出簽章驗證參數 (u_1, u_2) ，最後將簽章驗證參數 (u_1, u_2) 代入式(12)驗證。

2.4 進階加密標準

進階加密標準(Advanced Encryption Standard, AES)是由美國國家標準技術局(National Institute of Standards and Technology, NIST)所發布的對稱式區塊加密法[6]，

目的是取代 DES 的標準加密法，所以命名為進階加密標準。在 AES 候選研討會中評選準則包括安全性、成本、實作三個部份，最後是由比利時密碼學家 Joan Daemen 和 Vincent Rijmen 勝出，成為現今所看到的 AES。

AES 加密法區塊大小必須為 128 位元，金鑰長度則是 128、192、256 位元三種長度，其運算回合數對應金鑰長度 128、192、256 位元三種長度分別是十、十二、十四回合，加密的每個回合都會經過幾個過程。

圖 2 所示，加密過程包括 SubBytes、ShiftRows、MixColumns、AddRoundKey，如

圖 3 所示，解密過程則是 AddRoundKey、InvShiftRows、InvSubBytes、InvMixColumns。

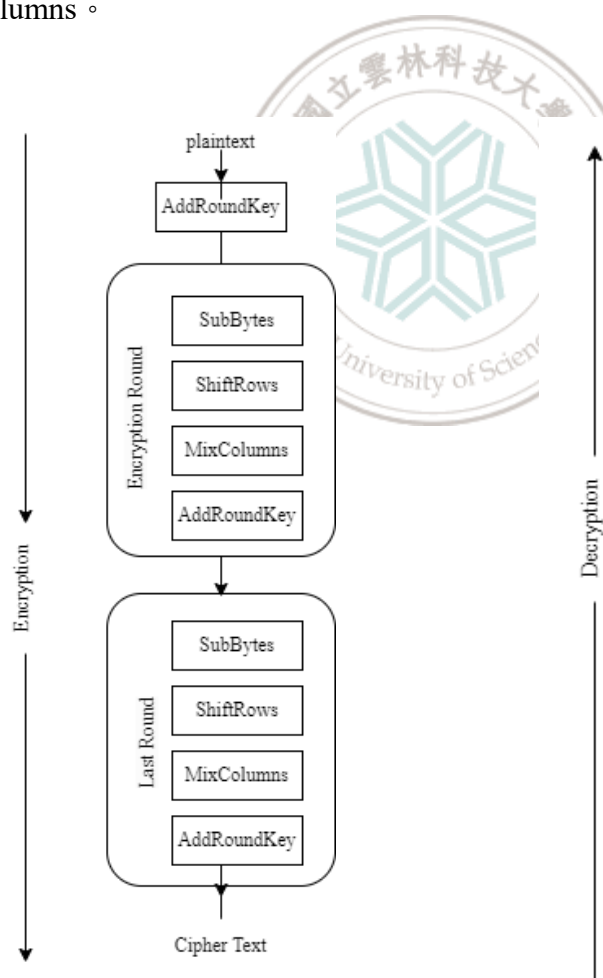


圖 2 AES 加密[12]

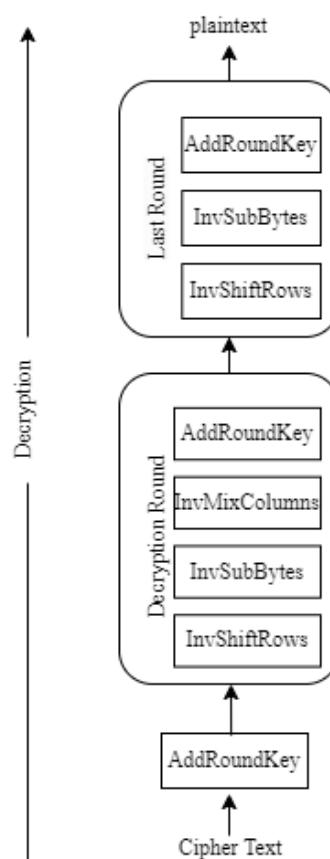


圖 3 AES 解密[12]

2.5 零信任簡介

零信任 (Zero Trust) 是一種網路安全策略，旨在確保所有通訊都通過安全加密進行，以防止未經授權的存取和資料洩露，在零信任環境中，單一企業資源的存取應該使用一個 session，並且遵循最小權限原則，這意味著只授予用戶完成特定任務所需的最少權限。採用動態決策的方式，根據存取者的身份和他們所請求的應用程式或服務來確定授予的權限。為確保企業資產的安全和完整性，企業需要持續監控其資產並對可能的威脅保持警覺。在零信任環境中，所有認證和授權過程都是動態的並且嚴格執行，以避免權限濫用和內部威脅。為提高安全性，企業應積極收集資產和網路基礎設施的相關資訊，並根據收集到的資訊制定改進措施。通過實施零信任策略，企業可以更有效地保護其資料和網路資源免受潛在攻擊。

2.6 零信任架構

零信任架構 (Zero Trust Architecture) 是根據美國國家標準技術研究所 (National Institute of Standards and Technology, NIST) 的 SP 800-207 文件[13]，如圖 4 所示，這是一個零信任存取的簡易運作流程。使用者在未信任區連線到 PDP(Policy Decision Point)/PEP(Policy Enforcement Point)，經過 PDP/PEP 驗證後，再連線到隱式安全區，最後才能成功存取到資源，這一過程確保了在整個存取過程中對身份和權限的嚴格控制，有助於阻止未經授權的存取和資料洩露。

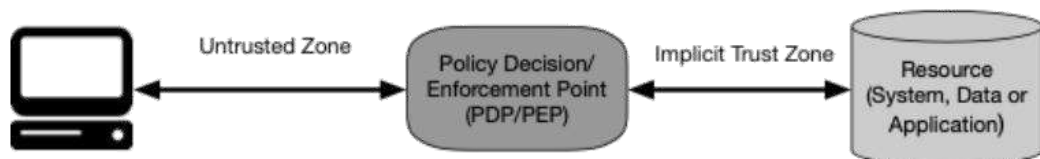


圖 4 零信任存取[10]

在實際部署上不單單僅有 PDP/PEP 的中介就能達成零信任，實際會多了幾個元件，以下是元件的描述：

1. Policy engine (PE)：決定一個存取者是否能存取資源，並且使用自身企業

之策略和外部來源如 Continuous diagnostics and mitigation (CDM) system 來協助評斷，負責與 Policy administrator 配對，進行策略引擎並記錄決策（批准或拒絕），位於 Control plane。

2. Policy administrator(PA)：生成任何 session 特定的身份驗證和驗證令牌或憑證，以及負責建立和/或關閉主體與資源之間的通訊路徑，實作可能會與 PE 整併成同一個元件，位於 Control plane。
3. Policy enforcement point (PEP)：如同閘道器的存在，位於客戶端與資源端中間的媒介，負責與使用者直接互動，負責轉發來自 PA 指令與政策更新，執行啟用、監視和最終終止主體與企業資源之間的連接，位於 Data Plane。
4. Continuous diagnostics and mitigation (CDM) system：檢查存取者是否有符合軟體更新規範。
5. Industry compliance system：特定產業如醫療的資訊安全法規。
6. Threat intelligence feed(s)：收集來自新發現的軟體缺陷和識別惡意程式。
7. Network and system activity logs：彙整資產日誌、網路流量、資源存取行為和其他事件，對企業資訊安全提供實際回饋。
8. Data access policies：資料存取政策：根據存取資源的屬性、規則、政策用一套規則編列。
9. Enterprise public key infrastructure, PKI：負責記錄儲存所有資源、應用程式等產生之公鑰，並統一在 PKI 管理。
10. ID management system：建立、儲存、管理企業用戶和身份紀錄，如姓名、電子郵件、憑證、存取屬性、分配的資產。
11. Security information and event management (SIEM) system：收集以安全為中心的訊息，供日後分析，用於完善決策。

將這些文件建構之後，如圖 5 所示，每當使用者存取資源時，必須先經過 PEP，PEP 會與 PA 溝通並發起驗證，PE 根據多個資料來源如 CDM、Industry compliance system、Threat intelligence feed(s)等作為依據，由 PA 進行允許或拒絕請求，PEP 再依照 PA 決策結果啟動或終止連線，並且 PA 政策是動態更新，持續更新最新的政策給 PEP。

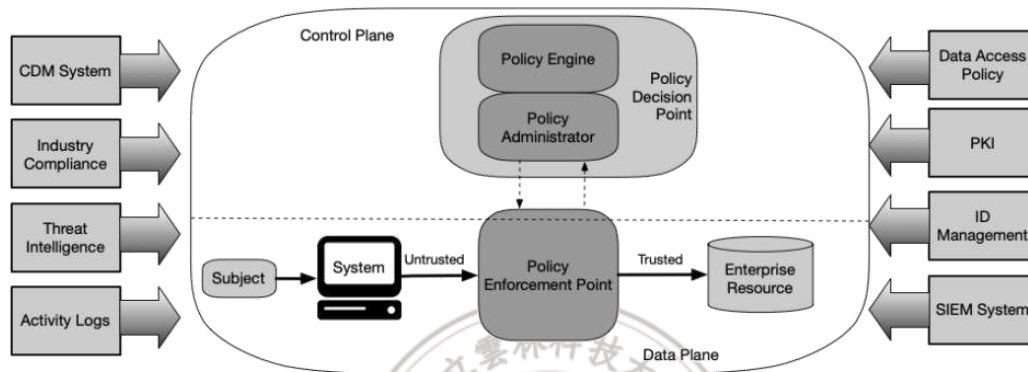


圖 5 核心零信任邏輯元件[13]

2.7 零信任網路

針對網路層面稱之為零信任網路（Zero Trust Network），意味著在網路層面上實施零信任原則並重新安全範圍，不再將經過驗證視為是安全的，而是動態進行調整。通常通過強化身份治理(Using Enhanced Identity Governance)、微分段(Using Micro-Segmentation)、軟體定義邊界(Using Network Infrastructure and Software Defined Perimeters)技術來實現，並且有些情境下會有兩個以上的方式來實作一個零信任架構。

2.7.1 強化身份治理(Enhanced Identity Governance)

資源在沒有使用者存取時，則沒有必要建立存取資源隧道。基於身份和其分配的權限，來評判是否擁有足夠的權限存取一個資源，並且依照使用裝置、資產情況、環境因素進行決策。

2.7.2 微分段(Micro-Segmentation)

將單個或是多個資源放置到安全的開道器下，即是特定網段，並採用次世代

防火牆作為 PEP，由特定網路設備來動態進行授權，保護資源在未經授權的情況下被存取或嗅探，不過在管理成本上和快速應變缺乏優勢。

2.7.3 軟體定義邊界(Software Defined Perimeters)

隨著企業網路變得越來越分散，它不再依賴於靜態的網路邊界，而是動態地創建安全連接，僅允許經過驗證和授權的使用者與設備存取資源，透過中央網路控制器(PA)進行動態授權，依據 PE 來決定重新配置網路，PEP 接收 PA 的指令，最終實現使用者經過 PEP 代理請求流量。

2.7.4 部署方式

實際部署上有幾種變化方式，可細分為四類型：

1. Device Agent/Gateway-Based Deployment：

為每個資產配置各自的閘道器，透過企業所配發的電腦，利用本地代理軟體進行連線，以存取企業資產。

2. Enclave-Based Deployment：

為 Device Agent/閘道器-Based Deployment 之變形，改成將一系列資產都採用同一個閘道器，透過企業所配發的電腦，利用本地代理軟體進行連線，以存取企業資產，也因閘道器保護對象是一系列資產，無法單獨保護每個資產。

3. Resource Portal-Based Deployment：

閘道器後方可以是單一個資產或是一系列資產，並無須在客戶端設備上安裝代理軟體，所以無法有效監控設備之情況。

4. Device Application Sandboxing：

每一資產都以沙盒形式運作，確保資產有效隔離，不過企業對於資產維護較無有效管理。

2.8 容器化技術

容器是一種現代化的輕量級虛擬化技術，它允許開發人員將程式及其相關的

依賴和設定檔打包成一個標準化的可執行單元。Docker 使用了 Linux 內核特性，如 cgroups 和 namespaces，來隔離容器的程式和網路，從而使每個容器表現得就像一個獨立的系統。

不同於傳統虛擬機，Docker 容器不需要啟動自己的操作系統，而是共享主機的內核。讓容器輕量化並可以在幾秒內啟動且使用更少的系統資源。由於它們不需要額外的操作系統，Docker 容器在啟動速度和性能方面遠遠超過傳統虛擬機。

在 AWS 中詳細介紹 Docker 的使用情境和與虛擬機器之差異[1]，如圖 6 所示，可以得知 Docker 和虛擬機器在架構上的不同，Docker 在架構上沒有獨立的 Guest OS，並且與本機共享 Guest OS 層，使得 Docker 更具輕量，而虛擬機器則是為每個獨立個體都分別建立 Guest OS，在資源消耗上較為多。

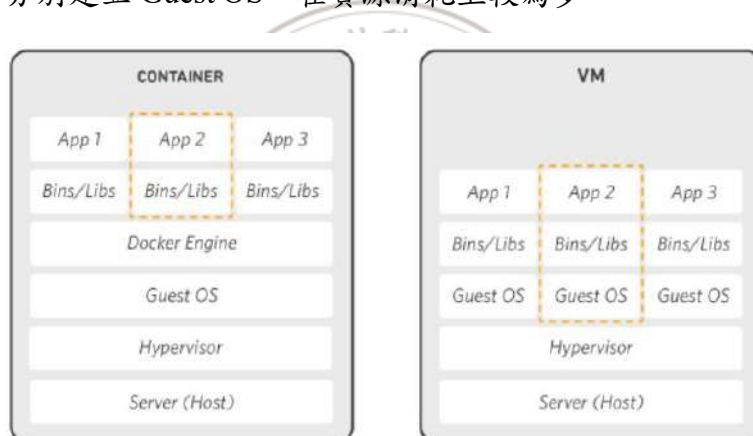


圖 6 Docker 與虛擬機區別[11]

在安全性方面提供額外的保護層，因每個容器都有自己的 cgroups 和 namespaces，所以即使容器受到攻擊不會影響到本機或是其他容器，但在設定上仍然要保持權限最小原則。

2.9 傳統網路安全疑慮

在一般傳統網路架構下，即在網路邊界部署防火牆和入侵偵測系統等資安設備，以護城河的形式保護其內部資源，但是這種方式存在既有的缺陷。傳統網路的策略在於通過一次檢查，就認定是可信任的連線來源，內網之間並無嚴格檢查

的機制。這種策略存在問題，一旦有駭客從外網連線並通過檢查，就可在整個內部網路環境中橫行無阻。

在近年受到疫情影響，許多企業發展出遠端辦公的工作方式。即便現今已經度過了疫情，遠端辦公的工作方式仍然會在許多企業中出現。當員工從自家網路連線至企業內網時，大多會使用由企業提供之虛擬私人網路(virtual private network, VPN)來進行連線。VPN 網路設計將遠端使用者視為可信任的，因而讓傳統網路邊界變得模糊不清，無法有效判斷連線者是否是合法使用者，迫使在安全管控上更加複雜。另外許多企業已將許多資訊服務從地端移至雲端，傳統網路邊界的安全策略已逐漸失效。基於使用者的網路位置來判斷真實身份易受中間人攻擊，且 VPN 不具備詳細的權限管控設定。每當連線者透過 VPN 連線到企業內網則與其他內網的連線者是相同的權限，駭客經由竊取 VPN 憑證進一步入侵企業網路環境，遠比突破層層資安設備還來的容易，故需要尋找取代傳統網路架構的解決方案。

如圖 7 所示，傳統網路架構可能存在這樣的配置，從外網至內網配置防火牆為主要抵擋外部攻擊的設備，配置路由器連線到各個資源服務。官方網站則大多會放置到 DMZ，以便與內網系統資源隔離。在黑框所示稱為內網資源服務，有庫存系統供內部員工管理庫存。藍框中虛擬化的 VPN 伺服器 and 軟體開發伺服器。

如果有一位倉庫管理部門員工透過公司 VPN 連線至內網，這位員工可以存取庫存系統、軟體開發伺服器、VPN 伺服器。但是軟體開發伺服器本應不是該員工的存取範圍，以 VPN 的概念分配連線權限的話，該名員工擁有整個內網的完整存取權限也增加內網風險。一旦任意員工的 VPN 遭受到駭客竊取，整個內網環境就受到嚴重的威脅，而且入侵的過程中不容易被管理員察覺。

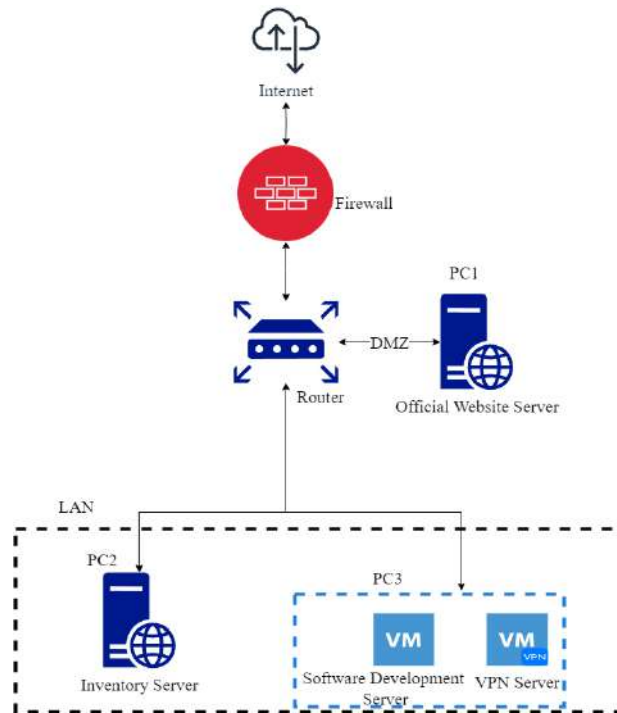


圖 7 傳統網路架構圖

2.10 傳統密碼系統安全疑慮

密碼系統本質存在缺陷，使用者往往會在多平台上使用相同一組密碼，一旦某個平台的密碼被破解，駭客就能輕鬆到其他平台使用相同的密碼登入。許多使用者在密碼設定上會以好記的單字、數字為組合，這類密碼易受到字典檔攻擊和暴力破解。傳統密碼系統會受到釣魚攻擊和社交工程，即便使用者設定相對複雜的密碼，駭客仍然可以欺騙手法取得使用者密碼，進而完全偽冒使用者在網路上的身份，系統本身是完全無法辨別連線者是否為合法用戶。

傳統密碼系統通常在伺服器中儲存和驗證密碼，一旦駭客攻陷伺服器，密碼將全數被竊取，即便伺服器儲存的是密碼的雜湊值，駭客仍然會使用彩虹表攻擊。所以不管如何將密碼儲存在伺服器都不是完善的安全策略，最終導致大規模的資安事件產生。

2.11 小結

回顧相關文獻得知，[16]及[9]不符合權限最小化原則，[15]並無 FIDO 認證機制，[13]及[14]缺乏動態控制隧道之功能，最後[9][15][16]皆無持續驗證機制。在應用場景方面，[15]僅適合部署在無線網路環境，[16]僅用於私有雲環境，[9]則用於一般 VPN 網路環境。

因應 COVID-19 後延伸出新的工作模式，遠端辦公已經成為新型態的工作模式，同時帶來網路安全環境新的攻擊。所以本文提出應用零信任與 FIDO 技術之身分驗證機制，主要以權限最小化、FIDO 認證機制、動態控制隧道、持續驗證等安全機制提供內網保護，並針對遠端辦公之場景進行研究。



三、 研究方法

3.1 系統概述

本文提出零信任和 FIDO 結合之網路架構，目的是解決傳統網路中易受到攻擊的安全疑慮，本文秉持零信任原則將每次連線以最小權限進行資料傳輸，應用結合 FIDO 機制改善傳統密碼系統的固有缺陷，去除密碼帶來的風險和使用者使用上的不便。

以零信任核心邏輯元件作為基礎架構加以改善，如表 3 所示，在多種部署零信任之方式進行考量，最終使用 Enclave-Based Deployment 作為部署基底。

表 3 零信任部署方式比較表

部署方式 比較項次	Device Agent/Gateway- Based Deployment	Enclave-Based Deployment	Resource Portal- Based Deployment
安裝代理程式	有	有	無
獨立閘道器	有	無	無
建置與部署	繁瑣	容易	容易
監控設備	有	有	無
資源隔離性	好	好	差

配合 FIDO 作為身份認證方式，運作流程架構，如圖 8 所示，使用者首先會與 PEP 進行連線，PEP 會和 PA/PE 查詢使用者是否已有過往連線紀錄，如果有則進入 FIDO 登入流程，如果沒有則進入 FIDO 註冊流程，在註冊流程裡使用者會透過 PEP 和 FIDO 伺服器進行驗證，最終會將使用者的公鑰儲存在伺服器中，以便使用者下次連線時透過將挑戰碼簽章，並回傳給 FIDO 伺服器進行驗證簽章，最後將 token 發放給使用者。

在本文系統下使用者方可在居家環境中安全地使用企業內網資源，不再需要記住複雜的密碼，使得駭客難以竊取使用者身份，取而代之的是使用現有的硬體設備進行身份確認。使用者只要擁有生物辨識功能的手機，即可在手機端進行生物辨識並與伺服器溝通，使用者體驗比往常方便且快速且不需額外花費成本。

採用 ECC 和 AES 混合加密作為訊息加密方式，以保證使用者手機和伺服器之間的傳輸是受到加密保障的，並將手機端與伺服器溝通後產生的 token 經由藍芽傳送至 PC，藍芽傳輸過程採用 AES 加密以確保點到點傳輸都是安全的，最後 PC 則可以向 PEP 發起暫態專屬隧道(Temporary isolated tunnel)。

暫態專屬隧道是為每位使用者特製的，只有在使用者通過驗證才能建立暫態專屬隧道，暫態專屬隧道僅能存取使用者存取的目標資源，其餘內網資源則完全沒有網路存取權。為落實零信任中的持續驗證概念，除了首次連線手機端會與伺服器進行通訊外，在使用暫態專屬隧道進行與資源連線的過程中，當使用者存取機敏性 API 和使用機敏指令時，則會要求使用者再次進行生物辨識驗證，並且將最新的 token 以當前金鑰加密後傳送，確保資料傳輸都是加密的。

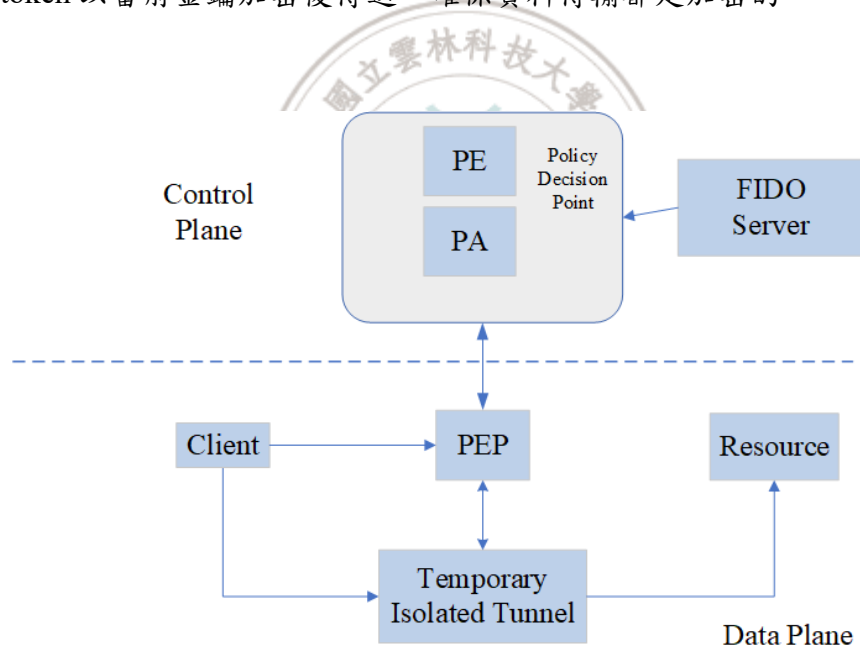


圖 8 零信任與 FIDO 安全架構圖

3.2 FIDO 認證機制

我們利用 FIDO 本地使用者手機上的生物辨識替代傳統的密碼驗證。這不僅增強用戶身份的保護，還減少對密碼的依賴，降低網路攻擊的風險。採用 Roaming Authenticators 的形式進行認證，手機端會將驗證結果傳送至使用者正在使用的設備如電腦，使用者電腦方可安全地上網。

在本論文方法中將傳統的 FIDO 機制進行改進，為落實零信任之概念，需在使用者手機端安裝代理程式，可有效監控使用者手機的運作情況。當偵測到手機端有疑慮被入侵或是非法操作，系統就會得知手機端設備可能存在風險，提取私鑰的流程就會被中斷，這是傳統 FIDO 無法完成的。因為傳統 FIDO 機制在預設情況下，完全信任手機處於安全環境，只要有驗證請求則會無條件地完成它。此外傳統 FIDO 認證以 Roaming Authenticators 的形式進行認證時，往往會在使用者裝置如瀏覽器上彈窗顯示 FIDO 認證的訊息，此舉動將會使得使用者正在瀏覽的頁面受到干擾，於是本文方法改善此問題，提供使用者更佳無感的驗證方式。

本文將認證機制分為註冊和登入，以下會分別詳細介紹，FIDO 機制之相關參數參照請參閱表 3。

表 4 FIDO 流程參數對照表

參數	說明
sk_C	客戶端私鑰
PK_C	客戶端公鑰
sk_S	伺服器私鑰
PK_S	伺服器公鑰
ch	128 位元挑戰碼
$h(ch)$	ch 之 SHA-256 雜湊值
$h(PK)$	PK 之 SHA-256 雜湊值
(s, r)	訊息簽章值
$S_{1h(ch)}$	伺服器對 ch 的 SHA-256 雜湊值簽章
$S_{2h(ch)}$	客戶端對 ch 之 SHA-256 雜湊值簽章
$V(ch, PK, S_{2h(ch)})$	伺服器驗證簽章
G	橢圓曲線基點
$sessionkey$	伺服器與手機端共享密鑰
$token$	128 位元存取令牌
$sessionkey_{BT}$	手機端和 PC 共享密鑰
$Encryption$	使用 $sessionkey_{BT}$ 對 $token$ AES 加密
$Decryption$	使用 $sessionkey_{BT}$ 對 $token$ AES 解密
$token_{BT}$	$sessionkey_{BT}$ 加密後的 $token$
$PK_S(token)$	以 PK_S 對 $token$ 加密

$$(x_3, y_3) = sk_C \times PK_S \quad (14)$$

$$(x_3, y_3) = sk_S \times PK_C \quad (15)$$

$$sessionkey = x_3 \quad (16)$$

3.2.1 FIDO 註冊機制

在使用者 APP 安裝在手機後，APP 中已經擁有 FIDO 伺服器之公鑰，以便進行以下傳輸工作。如圖 9 所示，流程步驟：

- 一、使用者在手機端輸入 ID 和 displayName。
- 二、進行生物辨識。
- 三、ID 和 displayName 發送至 FIDO Server。
- 四、當 FIDO 伺服器接收到請求後，生成 ch ，並將 ch 雜湊值簽章後會產生 $S_{1h(ch)}$ 。
- 五、FIDO 伺服器回傳給使用者 ch 和 $S_{1h(ch)}$ 。
- 六、使用者驗證伺服器的簽章。
- 七、使用者生成金鑰對 (PK_C, sk_C) 。
- 八、使用者將 ch 經過雜湊後產生 $h(ch)$ 。
- 九、使用者使用 sk_C 對 $h(ch)$ 進行簽章產生 $S_{2h(ch)}$ 。
- 十、使用者將 $S_{2h(ch)}$ 和 PK_C 傳送至 FIDO 伺服器。
- 十一、伺服器對 ch 進行雜湊產生 $h(ch)$ ，接著伺服器使用 PK_C 驗證 $S_{2h(ch)}$ 。
- 十二、伺服器將 $h(PK)$ 和 displayName 發佈至公鑰一致性檢查系統，並通知使用者。
- 十三、使用者確認自身公鑰確實與伺服器接收一致，並告知伺服器。
- 十四、伺服器將 ID、 PK_C 、displayName 一併儲存至資料庫。
- 十五、伺服器告知使用者資訊已正確接收。

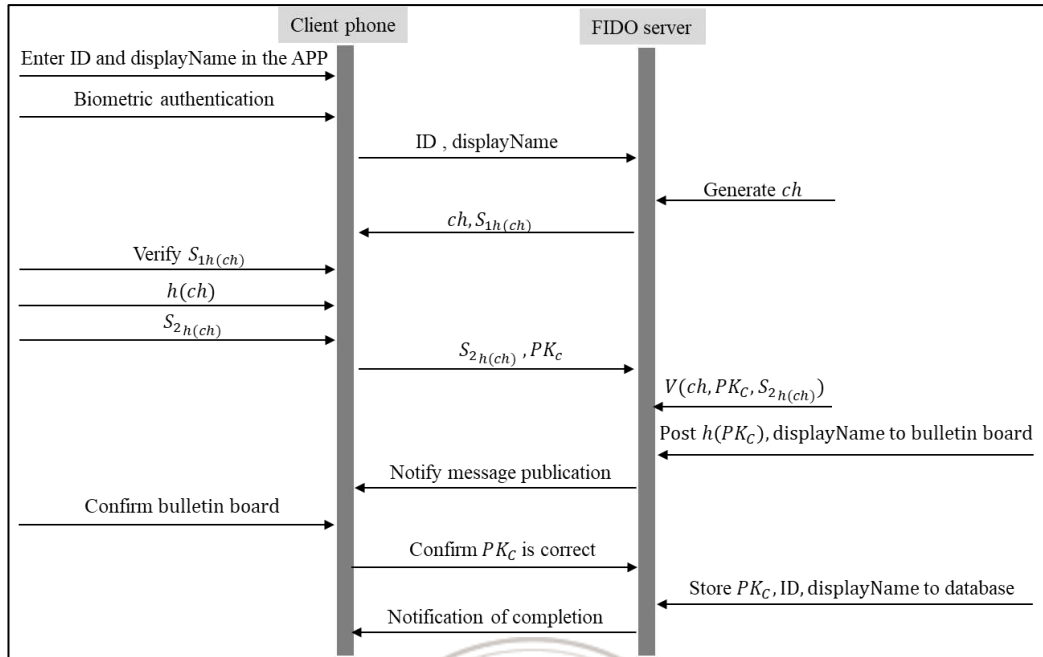


圖 9 FIDO 註冊

3.2.2 FIDO 登入機制

首先依照式(14)和式(15)可以算出一樣的點座標，並將 x 軸座標點當作 *sessionkey* 式(16)。手機與 FIDO 伺服器通訊如圖 10 所示，其使用與通訊流程步驟如下：

- 一、使用者 PC 藍芽連線到手機。
- 二、使用者在手機端輸入 ID 和 displayName。
- 三、進行生物辨識。
- 四、ID 和 displayName 發送至 FIDO Server。
- 五、當 FIDO 伺服器接收到請求後，生成 ch ，並將 ch 雜湊值簽章後會產生 $S_{1h(ch)}$ 。
- 六、FIDO 伺服器回傳給使用者 ch 和 $S_{1h(ch)}$ 。
- 七、使用者驗證 $S_{1h(ch)}$ 。
- 八、使用者計算出 ch 之雜湊值。

- 九、使用者使用 sk_C 對 $h(ch)$ 進行簽章後產生 $S_{2h(ch)}$ ，並傳送 $S_{2h(ch)}$ 到 FIDO 伺服器。
- 十、FIDO 伺服器端驗證 $S_{2h(ch)}$ 。
- 十一、 FIDO 伺服器端生成 $token$ ，並用 $sessionkey$ 進行對 $token$ 加密。
- 十二、 FIDO 伺服器傳送加密後的 $token$ 至使用者手機端。
- 十三、 使用者手機端用 $sessionkey$ 對 $token$ 進行解密。

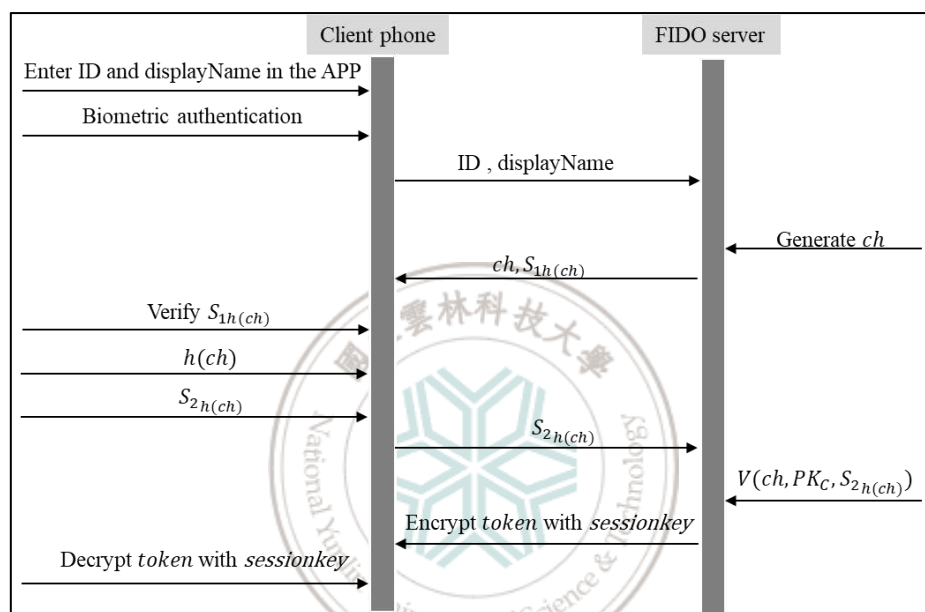


圖 10 FIDO 登入—手機與伺服器通訊

當手機端得到解密後的 $token$ ，如圖 11 展示手機與 PC 通訊的流程，並且在傳輸的過程中每段都保持加密，防止第三方未授權的竊聽。步驟說明如下：

- 一、手機端對 $token$ 使用 $sessionkey_{BT}$ 加密，其中 $sessionkey_{BT}$ 是在電腦與手機第一次配對時所建立的，用於日後的加解密。
- 二、手機端經由藍芽將 $token_{BT}$ 傳送至 PC。
- 三、PC 使用 $sessionkey_{BT}$ 對 $token_{BT}$ 進行解密。
- 四、PC 將 $token$ 使用 PK_S 加密後產生 $PK_S(token)$ ，轉傳至 FIDO 伺服器。

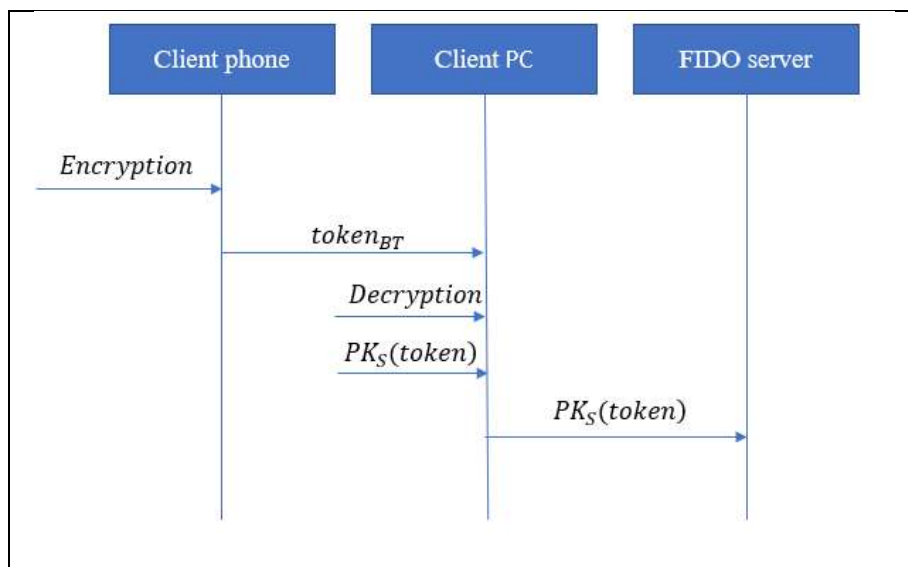


圖 11 FIDO 登入—手機、電腦和伺服器通訊

3.3 暫態專屬隧道(Temporary Isolated Tunnel)

採用 Docker container 作為暫態專屬隧道(Temporary Isolated Tunnel)的基底，如圖 12 所示，事先建立一個專屬的 Docker 映像檔，並且限制權限最小化，在此使用 SSH (Secure Shell) 作為基底。SSH 是一種加密的網路傳輸協定，通常用於遠端系統管理，使得使用者可以在不安全的網路環境中使用安全的傳輸，而且 SSH 可以提供比一般 FTP 更高的安全性。

映像檔在建置時以非 root 帳號運行 SSH 服務，並僅有執行 SSH Tunneling 的功能，將 SSH 配置檔在 Docker 映像檔創建時規劃好，如 PermitRootLogin 設定為 no、SSH 設定檔禁止寫入存取，最後採用金鑰作為使用者驗證機制。當 PEP 接收到正確 token 時將會建立 Docker container 並設定好金鑰。



圖 12 暫態專屬隧道前置建立工作

如圖 13 所示，當使用者已經通過驗證，使用者手機端會有 token，接著轉送 token 至 PC，PC 將 token 傳送到 PEP，PEP 會驗證 token 並且設定好金鑰，最後開啟暫態專屬隧道再通知使用者手機端，PC 就能通過暫態專屬隧道連線到內網資源了。

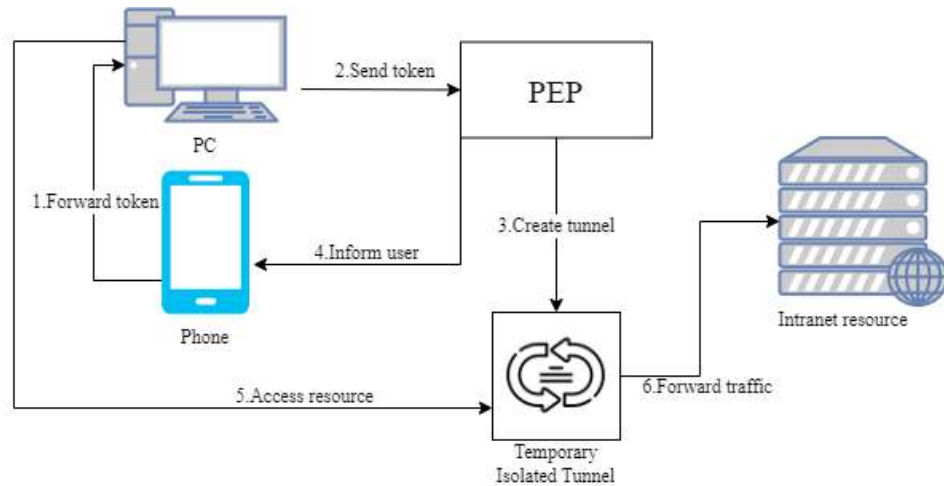


圖 13 暫態專屬隧道存取內網資源流程圖研究成果

如圖 14 所示，為落實持續驗證的概念，當使用者已經通過首次驗證後還得進行持續驗證，以確保使用者的數位身份是本人使用，系統中固定時效內或是存取更多機敏性的資料時必須進行一次生物辨識，依照資產的機敏性和使用者連線情況做調整，token 必須在時效內進行更新，如果使用者沒有通過驗證，就無法得到最新的 token，暫態專屬隧道將會立即關閉。本論文系統也提供機敏指令和 API 觸發驗證的機制，當使用者使用機敏指令和 API 時，系統將會觸發請求使用者生物辨識，以便確認當前存取者為使用者本人。

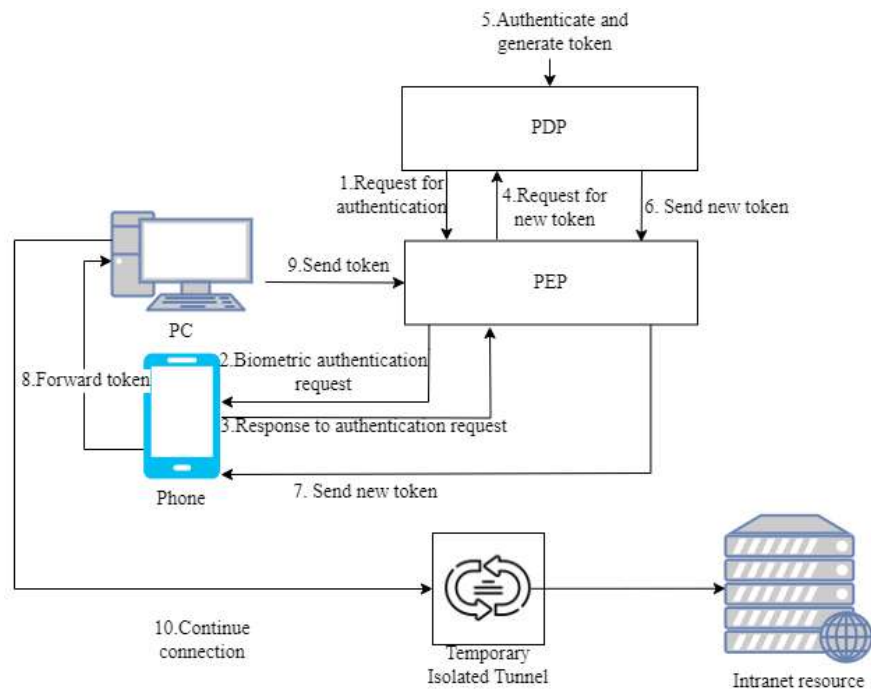


圖 14 持續驗證流程

3.4 SSH Tunneling

本文採用 SSH Tunneling 技術來達成動態建立及刪除隧道，內部網頁並未直接對外網公開，僅有 Temporary Isolated Tunnel 這個元件可以存取到內部網頁，確保連線權限以最小化原則，故利用 SSH Tunneling 建立隧道，使用者通過 FIDO 驗證後則會依照圖 15 所示。使用者電腦會與 Temporary Isolated Tunnel 進行 SSH 連線，Temporary Isolated Tunnel 再將流量以相同的方式進行 SSH Tunneling 將流量連線到內部網頁的主機上，並且使用者並不知道內部網頁真實的 IP 和情況以減少攻擊風險，並保障內網資源之安全性。

在連線的過程中 Temporary Isolated Tunnel 只負責 SSH Tunneling，隧道的開啟或是關閉都是由 PDP 來決策，一旦使用者沒有通過持續驗證，Temporary Isolated Tunnel 則會收到指令並關閉動態專屬隧道。

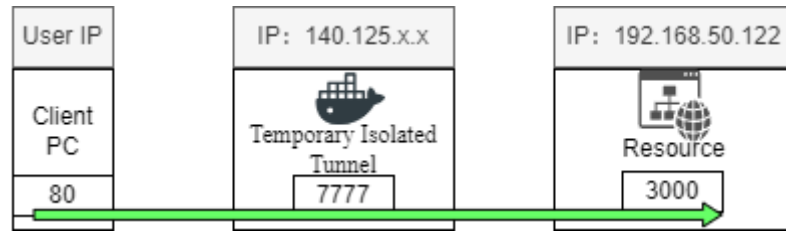


圖 15 SSH Tunneling 應用

3.5 零信任架構

將零信任概念融入架構和驗證機制對於資訊安全有很良好的改善，以 FIDO 作為認證手段，以下介紹使用者使用本文的網路架構是如何存取內網資源。

3.5.1 在註冊流程下零信任架構如何運作

在首次註冊流程，使用者開啟 APP 並輸入 ID、displayName 傳送至 PEP，PEP 將資料轉送至 PDP。PDP 可分為兩個元件，分別是 PA 和 PE，PE 會判斷來源使用者是否曾經註冊過，如果沒有註冊過則進去註冊流程，向 FIDO 伺服器請求註冊用挑戰碼(challenge code)，此時使用者手機經過 PEP 和 PDP 後，與 FIDO 伺服器建立連線，FIDO 伺服器與使用者成功交換資料後，使用者到公鑰公佈欄確認公鑰已正確被儲存，最後伺服器會將使用者資訊儲存至伺服器內。

3.5.2 在登入流程下零信任架構如何運作

在登入流程，使用者開啟藍芽並連線到電腦，依照圖 16 所示，會依照流程開始驗證並發送 token，首先在手機開啟 APP 並輸入 ID、displayName 傳送至 PEP，PEP 將資料轉送至 PDP。PDP 可分為兩個元件，分別是 PA 和 PE，PE 會判斷來源使用者是否曾經註冊過，如果有註冊過則進去登入流程，FIDO 伺服器會回傳給 PE 經由使用者公鑰加密過的 token，PE 再將 token 轉發給 PA，PA 再轉發給 PEP，PEP 最終才會將 token 送到使用者手中，並且 FIDO 伺服器將 token 和使用者資訊

進行記錄。當使用者接收到 token 後經由自身私鑰解開，並使用對稱式加密 token 後經由藍芽傳送至電腦，電腦在解密後得到明文 token，最後使用伺服器端公鑰加密後傳送，FIDO 伺服器在驗證後就能向 PEP 請求暫態專屬隧道，電腦可以通過暫態專屬隧道連線到內網資源，PEP 會持續紀錄使用者與資源間連線的情況，當電腦沒有轉發到最新的 token 到 PEP，PA 則會終止暫態專屬隧道。

圖 16 在登入情境下零信任運作過程

3.5.3 在持續驗證流程下零信任架構如何運作

每個內網資源存取都有對應的 session 時效，依照資源的機敏性進行調整，高機敏性資源則有更短的時效。使用者進行高權限存取時內網資源會再次發出驗證請求，以確保降低資安風險。使用者如果更換連線網路或設備時將會觸發使用者狀態異常，確保當前使用者是合法使用者。

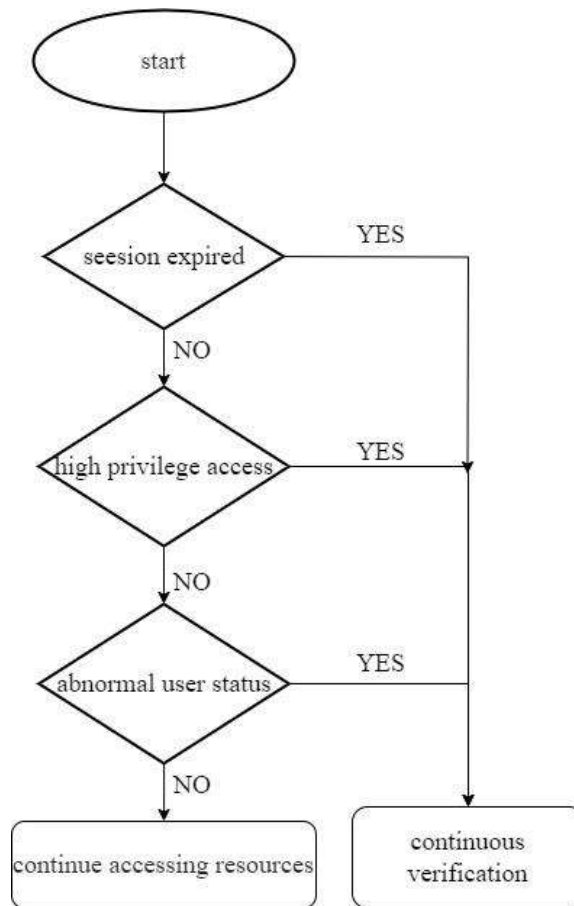


圖 17 持續驗證判斷流程圖

四、實驗結果

4.1 實驗軟體規格

在本實驗中使用了必要的軟硬體來完成實驗，以下將說明軟硬體的配置，並將其分類為伺服器、手機和筆記型電腦三個主要部分。PDP 元件以 Node.js 作為核心開發語言，並且為自主研發。PEP 元件以 Node.js 作為研發之程式語言並且搭配 NGINX 來實現負載平衡。FIDO Server 採用 Python3.8 並搭配 MongoDB 進行資料儲存和管理。對於手機部分選擇使用 Android 9 作為開發版本，以確保兼容性和效能。而在筆記型電腦方面，選擇 Ubuntu 20 作為作業系統，並搭配 Python 3.8 作為電腦端軟體開發語言，充分利用其穩定性和開放性作為實驗的執行平台，透過這樣的配置能夠更有效地進行實驗並取得可靠的結果。

4.2 實驗環境

如圖 18 所示，企業網路環境在導入零信任與 FIDO 技術後將防火牆替換成 PEP，並且對於每筆連線有強烈的身份驗證及權限管控。在內網環境中允許任何內網使用者存取公司之庫存系統，但僅能看到部分資訊，必須透過管理員登入才能看到更多的庫存資訊。

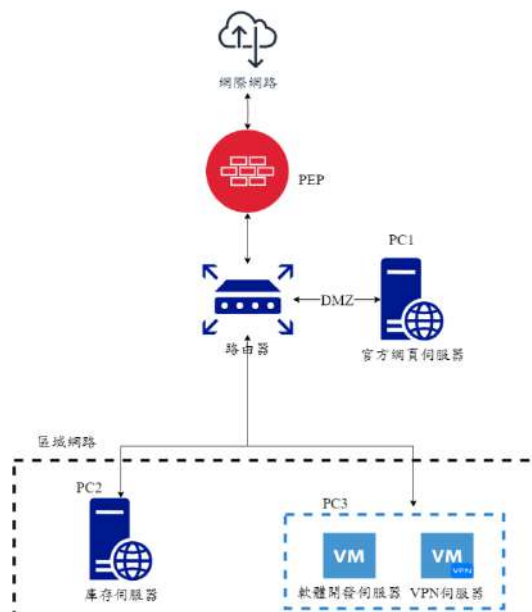


圖 18 模擬實驗環境

4.3 實驗模擬環境之註冊流程

手機登入使用流程：(1)如圖 19 使用者先在 APP 點擊 REGISTER 按鈕進行註冊作業；(2)如圖 20 在註冊頁面有兩個欄位分別是 ID、displayName，其中 ID 是使用者編號，displayName 則是系統對於使用者的稱呼；(3)如圖 21 使用者由註冊頁面輸入系統派發的 ID 和自定義的 displayName，在範例中輸入 ID 為 myid 和 displayName 為 mydisplayname2；(4)如圖 22 所示按下指紋辨識的圖案則會觸發生物辨識請求；(5)當使用者成功驗證後則會出現如圖 23，並在畫面顯示紅字的驗證成功，失敗則會如圖 24 所示顯示紅字的驗證失敗。

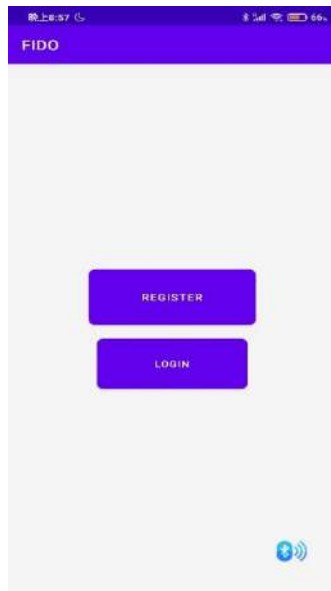


圖 19 APP 初始頁面



圖 20 APP 註冊初始頁面



圖 21 APP 註冊頁面輸入資料



圖 22 APP 註冊頁面進行生物辨識



圖 23 APP 註冊頁面生物辨識驗證成功



圖 24 APP 註冊頁面生物辨識驗證失敗

在成功通過生物驗證後會將 ID、displayName 傳送至 FIDO 伺服器，如圖 25 所示，FIDO 伺服器會以亂數生成 128bits 的挑戰碼最後回傳至使用者手機端，如圖 26 所示，使用者手機在本機生成金鑰對並以十六進制呈現，將挑戰碼進行 SHA-256 後以私鑰簽章，最終傳送簽章訊息和公鑰發送至 FIDO 伺服器。

```
/preregister
:challenge_code 17e15efae0daef999f1482dc8920b89d
```

圖 25 FIDO 伺服器生成 128bits 挑戰碼

```
/getkeypairhex
publicKeyHex 0489d4da01a7055b8ad7ebae5090a071fcbdcc757ccd409a12d7f9974f34f05b3717a2c8ae487dc8e9b
9288dde3a22385d918119f6dfaf77c894a580730d385333
privateKeyHex f2a5ac56e7c6953f33b4ee9f39856fe19dd912069de5ea85883e30493400b49b
```

圖 26 使用者手機在本機生成金鑰

FIDO 伺服器接收來自使用者的資料後會進行簽章驗證，最後發佈到公鑰公佈欄上，使用者經由公鑰公佈欄確認自身公鑰已正確被接收，如圖 27 最後 FIDO 伺服器會將使用者訊息存入資料庫，訊息包含公鑰的十六進制、簽章後的挑戰碼、ID、displayName 以供未來登入時查找對應身份。

```
{
  "_id" : ObjectId("6458bbb2a68d24ecde5e9e0a"),
  "publicKeyHex" : "0489d4da01a7055b8ad7ebae5090a071fcbdcc757ccd409a12d7f9
974f34f05b3717a2c8ae487dc8e9b9288dde3a22385d918119f6dfaf77c894a580730d385333",
  "hashedChallengeHex" : "a7eb3cdf29d0e6850fa15732d0e3dcfdb0d093fa0203403f
e8c875ceefaf5a39",
  "name" : "myid",
  "displayName" : "mydisplayname2",
  "__v" : 0
}
```

圖 27 使用者訊息存入資料庫

完成註冊後使用者已將訊息安全的傳送至 FIDO 伺服器，在下次登入時則需提供 ID、displayName 進行登入。

4.4 實驗模擬環境之登入流程

使用者在電腦上開啟本系統的連線程式，會顯示當前的狀態，包括使用者名稱、狀態、Token，如圖 28 所示。接著按下 Click me 按鈕，啟動電腦代理程式，使得電腦端開啟藍芽伺服器監聽，如圖 29 所示，以便接下來接收來自手機端的資料。

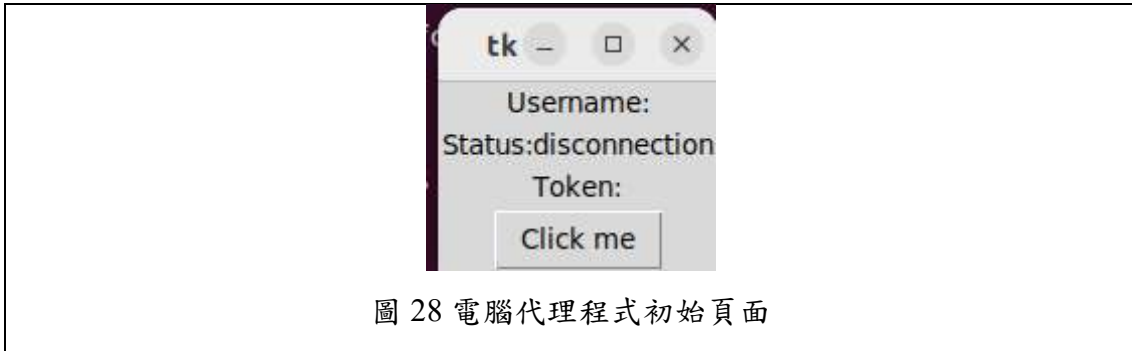


圖 28 電腦代理程式初始頁面



圖 29 電腦代理程式藍芽伺服器開啟

如圖 30 所示，使用者在 APP 中點擊右下角藍芽圖案進行藍芽配對作業，如圖 31 所示，在範例中選擇藍芽裝置叫做 es602-UX550VE，並且點擊 CONNECT，此時手機端會發送 hello 的字串至電腦，如圖 32 所示，電腦藍芽伺服器成功接收來自手機端的字串並且印出，證明連線無誤才會進行下一步。

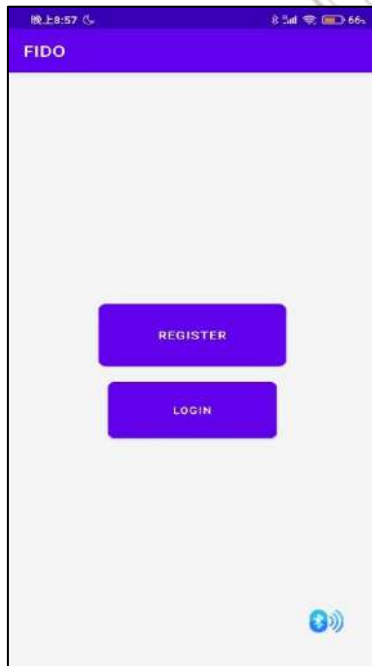


圖 30 APP 初始頁面

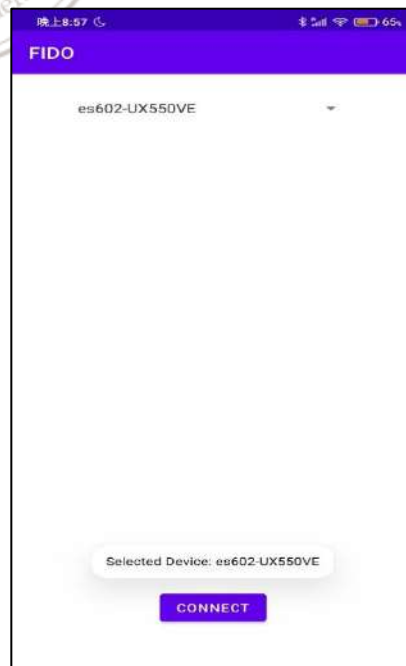


圖 31 APP 選擇藍芽配對

```
es602@es602-UX550VE:~/Downloads/fido_blueetooth_PC$ python mainUI.py
File 'notification.txt' does not exist.
rf-server running
Waiting for connection on RFCOMM channel 1
Accepted connection from ('A8:9C:ED:D2:80:BB', 1)
Received hello
Disconnected from ('A8:9C:ED:D2:80:BB', 1).
```

圖 32 電腦代理程式藍芽伺服器接收資料

確實手機和電腦藍芽已正常連線後則在手機端進行登入，如圖 33 所示，在手機端 APP 上輸入先前註冊的資訊，並且通過生物辨識驗證，手機端會將 token 經由藍芽傳送到電腦端。



圖 33 APP 登入頁面

如圖 34 所示，電腦已經成功接收到加密過後的 token，並且將 token 解密後傳送到伺服器，最後獲得暫態專屬隧道，如圖 35 所示，並且將最新的資訊更新到電腦程式的前端頁面，顯示當前使用者名稱以及狀態是變為已連線狀態。


```
Accepted connection from ('A8:9C:ED:D2:80:BB', 1)
Received {"encrypted": "2c168d5c020022de04358af90f7e2dd36c0ef6e9b4113eb47992a5bfd6e2aa7ff813b7d0216fa74a67a134708bf55999", "ivhexData": "9f74c62aaa2ed6493c7dfef95dfacd7e"}
Updating UI...
encrypted: 2c168d5c020022de04358af90f7e2dd36c0ef6e9b4113eb47992a5bfd6e2aa7ff813b7d0216fa74a67a134708bf55999
ivhexData: 9f74c62aaa2ed6493c7dfef95dfacd7e
OpenSSH_8.9p1 Ubuntu-3ubuntu0.1, OpenSSL 3.0.2 15 Mar 2022
```

圖 34 電腦代理程式接收 token

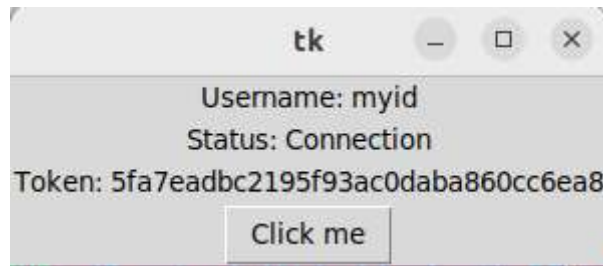


圖 35 電腦代理程式前端頁面

如圖 36 所示，此時使用者的電腦已經可經由暫態專屬隧道連線到內網的庫存系統網頁服務，在首次登入的情況下，使用者僅有看到商品名稱和數量，無法看到詳細的價錢資訊，必須透過完成持續驗證方可提升權限。

內部系統	
輸入名稱	
管理員登入	
商品資料	
商品名稱	數量
商品A	10
商品B	5
商品C	20
商品D	8
商品E	15
商品F	12
商品G	9
商品H	18
商品I	21
商品J	11
商品K	13
商品L	17
商品M	6
商品N	7
商品O	19

圖 36 使用者首次存取內網庫存系統網頁

4.5 實驗模擬環境之持續驗證流程

在 4-3 章節中描述登入的流程後，使用者方可順利透過暫態專屬隧道存取到內網的庫存系統網頁上，本文系統秉持權限最小化原則以及持續驗證機制，要求使用者取得更高權限時要進行 FIDO 認證。

在第一個持續驗證的範例中，此使用者是擁有進階權限的管理員帳號，所以可以在驗證後查看更多的資訊，其他使用者則無法透過驗證提升權限，如圖 37 所示，在電腦網頁輸入使用者 ID 並且按下管理員登入。

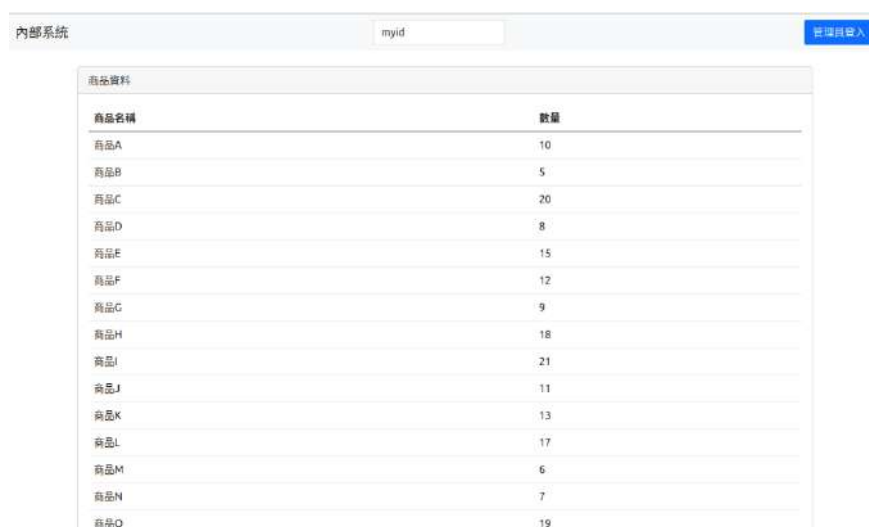


圖 37 電腦端管理員登入

管理員登入按鈕按下後，如圖 38 所示，網頁上顯示轉圈圈之動畫，此時使用者到手機端再次到登入頁面進行生物辨識，通過驗證後則再網頁按下”驗證後按下”的按鈕，如圖 39 所示，如驗證成功則會成功顯示總價格欄位，如圖 40 所示，驗證失敗則會彈窗顯示驗證失敗。



圖 38 持續驗證頁面

驗證成功		
內部系統		
myid		
管理員登入		
商品資料		
商品名稱	數量	總價格
商品A	10	100
商品B	5	50
商品C	20	200

圖 39 持續驗證之成功頁面

驗證失敗		
內部系統		
myid		
管理員登入		
商品資料		
商品名稱	數量	
商品A	10	
商品B	5	
商品C	20	

圖 40 持續驗證之失敗頁面

在第二個持續驗證的範例中，此使用者為普通使用者帳號，無法在驗證後查看更多的資訊，如圖 41 所示，在網頁上不會看到管理員登入之按鈕，也無法透過直接調用 API 來進行管理員持續驗證，由此可以看出系統對於不同權限之使用者有不同的權限頁面分級。

內部系統	
商品資料	
商品名稱	數量
商品A	10
商品B	5
商品C	20
商品D	8
商品E	15
商品F	12
商品G	9
商品H	18
商品I	21
商品J	11
商品K	13
商品L	17
商品M	6
商品N	7
商品O	19

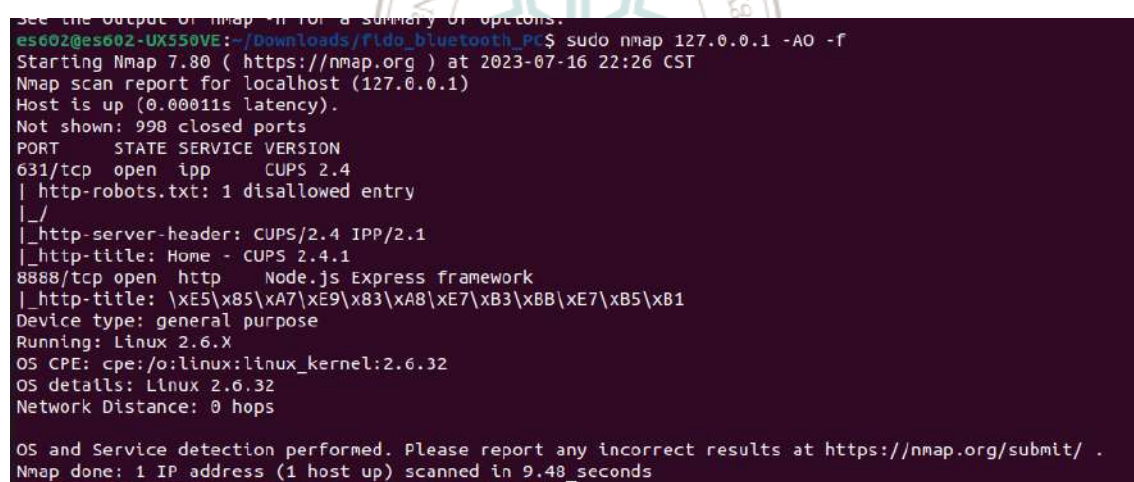
圖 41 電腦端普通使用者登入

4.6 攻擊情境

在這個章節中會探討可能的攻擊情境，透過情境模擬我們可以理解攻擊者可能採取的手法，並且在本文方法對抗攻擊情境時如何應對防禦，協助了解系統解決了哪些資安風險。

4.6.1 使用者電腦遭受入侵

在使用者電腦遭受入侵這個情境中，攻擊者已經駭入已取得庫存系統授權的裝置，攻擊者會嘗試利用使用者之電腦進行橫向移動。首先攻擊者會使用嗅探工具 Nmap 來了解目標伺服器上所擁有的服務，因本文所有連線透過暫態專屬隧道，並且將本機上的 port 透過 SSH Tunneling 來轉送流量，故攻擊者掃描的 IP 對象是 127.0.0.1。如圖 42 所示，攻擊者最終只會發現庫存系統服務，其餘服務則不會因嗅探而暴露。



```
see the output of nmap -h for a summary of options.
es602@es602-UX550VE:~/Downloads/fido_blueetooth_PC$ sudo nmap 127.0.0.1 -AO -f
Starting Nmap 7.80 ( https://nmap.org ) at 2023-07-16 22:26 CST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00011s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
631/tcp   open  ipp      CUPS 2.4
|_ http-robots.txt: 1 disallowed entry
|_/
|_ http-server-header: CUPS/2.4 IPP/2.1
|_ http-title: Home - CUPS 2.4.1
8888/tcp   open  http     Node.js Express framework
|_ http-title: \xE5\x85\xA7\xE9\x83\xA8\xE7\xB3\xBB\xE7\xB5\xB1
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.48 seconds
```

圖 42 攻擊者使用 Nmap 嗅探內網

如果攻擊者透過其他管道得知軟體開發伺服器之連線 port 為 80，一樣使用 Nmap 進行嗅探，如圖 43 所示。攻擊者會無法得知軟體開發伺服器服務的情況，由此可見暫態專屬隧道將每個資源進行了隔離，使得攻擊者無法成功進行橫向移動。

```
Nmap done: 1 IP address (1 host up) scanned in 1.94 seconds
es602@es602-UX550VE: ~/Downloads/fido_bluetooth_PC$ sudo nmap 127.0.0.1 -p 80 -A0
Starting Nmap 7.80 ( https://nmap.org ) at 2023-07-16 21:04 CST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000057s latency).

PORT      STATE SERVICE VERSION
80/tcp    closed http
Too many fingerprints match this host to give specific OS details
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.92 seconds
```

圖 43 攻擊者使用 Nmap 嗅探開發伺服器

4.7 相關論文比較

Jung 等人提出[15]用於保護無線網路安全仍然缺少一定的缺陷，駭客依舊可以經由竊取密碼進行內網破壞。Gordin 等人提出[16]用於私有雲添加無密碼登入，但無法有效完整驗證使用者，仍然會在驗證後遭受駭客竊取憑證。Huseynov 提出[9]缺乏權限有效管控，一旦任何使用者憑證遭到竊取，整個內網面臨嚴重危害。於是本文提出同時採用零信任和 FIDO 驗證機制用於保護企業內網並改善上述論文中的缺點，以帶來更便捷的使用和安全資源存取的流程。

如表 5 所示，Jung 等人[15]、Gordin 等人[16]、Huseynov[9]與本論文之比較，比較項次有權限最小化、流量加密、FIDO 驗證機制、動態控制隧道、持續驗證和使用情境。

表 5 相關論文比較表

文獻名稱 比較項次	Jung 等人[15]	Gordin 等人[16]	Huseynov [9]	本論文
權限最小化	有	無	無	有
流量加密	有	有	有	有
FIDO 驗證機制	無	有	有	有
動態控制隧道	有	無	無	有
持續驗證	無	無	無	有
使用情境	無線網路	私有雲	VPN	遠端辦公

五、 結論與未來展望

5.1 結論

網路提供人們便利，但也因此帶來巨大的資安威脅，所以我們提供零信任結合 FIDO 之解決方案，去除密碼帶來的風險和傳統網路的缺陷，能夠解決許多現有網路安全技術存在的問題，例如傳統的 VPN 連線方式存在憑證竊取之風險。

為了實現系統安全性和使用者便利性的雙贏局面，搭配 FIDO 實現無密碼登入和持續驗證機制，使用者無需輸入密碼即可完成驗證，從而避免密碼外洩和使用者驗證疲勞攻擊。使用者僅需透過壓指紋的動作即可在系統中實現橢圓曲線非對稱式加密和橢圓曲線簽章演算法。

採用非對稱式加密來保證訊息的機密性和簽章演算法來保證訊息的完整性，同時結合手機既有硬體具有生物辨識的特性，既保證安全性又提供便利性，駭客難以破解非對稱式加密，因此保障整體內網系統的安全。身份驗證是零信任認證的一大考驗，採用無密碼登入和生物辨識驗證的方式，也減少憑證竊取和使用者驗證疲勞攻擊帶來的危害，加強系統的安全性和可靠性。

在連線過程中我們採用暫態專屬隧道來實現零信任的動態存取機制，每筆連線都具有時效性和最小權限原則，並在限定時間內或觸發機敏 API 時要求使用者再次進行生物辨識驗證，這減少憑證遭受竊取帶來的危害，即使憑證不慎外洩也不會導致整個內網環境受到攻擊。因此駭客是無法僅攻破內外網隔離的防火牆來取得完整內網環境的信任。

5.2 未來展望

從傳統網路移植到零信任架構是一個漫長的過程，並非一氣呵成就能實現架構轉換，在部署零信任架構相較傳統網路有更多的步驟和複雜性，也讓管理上可能產生的疏忽導致最終更不安全。所以需要具有整合管理平台的工具，使得管理人員可以更簡單的去理解網路基礎設施架構和權限控制派發，並且需要有效的連線紀錄和整理，搭配人工智慧有效偵測惡意連線，再依照每個企業環境進行動態決策，建構安全資訊安全環境。



參考文獻

- [1] Machani, S. & Field, N. (2022). White Paper: Choosing FIDO Authenticators for Enterprise Use Cases, <https://media.fidoalliance.org/wp-content/uploads/2022/03/FIDO-White-Paper-Choosing-FIDO-Authenticators-for-Enterprise-Use-Cases-RD10-2022.03.01.pdf>.
- [2] FIDO Government Deployment Working Group (2022). White Paper: FIDO for e-Government Services, <https://media.fidoalliance.org/wp-content/uploads/2022/12/FIDO-e-Government-White-Paper.pdf>.
- [3] Ghosh, D. J., Bhatnagar, T., Mathur, A., & Ramaswamy, S. (2022). White Paper: FIDO Authentication in Digital Payment Security, <https://media.fidoalliance.org/wp-content/uploads/2022/09/FIDO-White-Paper-FIDO-Authentication-in-Digital-Payment.pdf>.
- [4] Lyastani, S. G., Schilling, M., Neumayr, M., Backes, M., & Bugiel, S. (2020). Is FIDO2 The Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. In 2020 IEEE Symposium on Security and Privacy (SP), pp. 268-285.
- [5] Lin, W. H., Yang, G. Y., & Yeh, K. H. (2022). Integrating FIDO Authentication with New Digital Identity in Taiwan. In 2022 IEEE 11th Global Conference on Consumer Electronics (GCCE), pp. 311-312.
- [6] Daemen, J., & Rijmen, V. (1999). AES proposal: Rijndael.
- [7] Morii, M., Tanioka, H., Ohira, K., Sano, M., Seki, Y., Matsuura, K., & Ueta, T. (2017). Research on Integrated Authentication sing Passwordless Authentication Method. In 2017 IEEE 41st annual computer software and applications conference (COMPSAC), Vol. 1, pp. 682-685.

- [8] Lee, M., Kang, H., Kwak, J., Kim, D., Jeong, H., & Seo, J. T. (2020). Proposed On-site Document Sharing System Using FIDO. In Proceedings of The 2020 ACM International Conference on Intelligent Computing and Its Emerging Applications, pp. 1-6.
- [9] Huseynov, E. (2022). Passwordless VPN using FIDO2 Security Keys: Modern Authentication Security for Legacy VPN Systems. In 2022 4th International Conference on Data Intelligence and Security (ICDIS), pp. 01-03.
- [10] Koblitz, N. (1987). Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177), pp.203-209.
- [11] Ulla, M. M., & Sakkari, D. S. (2023). Research on Elliptic Curve Crypto System with Bitcoin Curves–SECP256k1, NIST256p, NIST521p and LLL. *Journal of Cyber Security and Mobility*, pp.103-128.
- [12] Forouzan, B. A. (2008). *Cryptography and Network Security*. McGraw-Hill.
- [13] Stafford, V. A. (2020). Zero trust architecture. NIST special publication, 800-207.
- [14] Docker 與虛擬機之差異 <https://aws.amazon.com/tw/docker/>.
- [15] Jung, B.G., Yoo, Y.S., Kim, K., Kim, B.S., Lee, H., & Park, H.S. (2022). ZTA-based Federated Policy Control Paradigm for Enterprise Wireless Network Infrastructure. In 2022 27th Asia Pacific Conference on Communications (APCC). IEEE, pp. 1-5.
- [16] Gordin, I., Graur, A., Vlad, S., & Adomniței, C. I. (2021, November). Moving Forward Passwordless Authentication: Challenges and Implementations for the Private Cloud. In 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), pp. 1-5.