

**科技部補助  
大專學生研究計畫研究成果報告**

計畫

名稱 : 基於雲端環境下智慧製造系統之安全通道實作分析

報告類別 : 成果報告

執行計畫學生 : 江忠晏

學生計畫編號 : MOST 110-2813-C-011-010-E

研究期間 : 110年07月01日至111年02月28日止，計8個月

指導教授 : 羅乃維

處理方式 : 本計畫可公開查詢

執行單位 : 國立臺灣科技大學資訊管理系

中華民國 111年02月22日

(一) 摘要.....	3
(二) Abstract.....	4
(三) 前言與研究目的.....	5
(四) 文獻探討.....	8
1. 物聯網安全及架構.....	8
2. 安全通道建構技術.....	8
2.1 虛擬私人網路.....	8
2.2 穿隧協議.....	9
3. IPsec 機制 .....	9
3.1 封裝模式.....	9
3.2 安全協議.....	9
3.3 安全關聯.....	10
3.4 網際網路金鑰交換.....	11
3.5 實作模式.....	11
3.6 XFRM 框架 .....	12
3.7 加密演算法、雜湊函式.....	14
4. 硬體加速.....	15
(五) 研究方法及步驟.....	21
1. 上雲架構模擬設計 .....	21
1.1 工業物聯網架構流程.....	21
1.2 雲端服務架構流程.....	23
2. 安全通道研究方法.....	27
2.1 自定義演算法為小眾.....	27
2.2 符合當今硬體趨勢.....	28
2.3 提供產業更好的解決方式.....	28
3. 路由器作業系統及硬體加密加速分析.....	28
4. 安全通道建置及分析.....	30
4.1 工廠本地端.....	30

4.2	雲端架構端.....	33
4.3	行動裝置端.....	35
(六)	結果與討論.....	37
1.	上雲架構模擬設計.....	37
2.	路由器作業系統及硬體加密加速.....	39
3.	安全通道建置及分析.....	41
3.1	TP-Link Archer C7 AC1750 至雲端架構安全通道.....	41
3.2	iPhone 13 (iOS 15.3) 至雲端架構安全通道 .....	49
4.	執行計畫過程遇到之困難或阻礙.....	50
4.1	EC2 Instance 作為 NAT (已解決) .....	50
4.2	安全通道防火牆規則處理 (已解決) .....	50
4.3	路由器作業系統及硬體加密加速 (未解決) .....	50
4.4	Banana PI R64 至雲端架構安全通道建置 (未解決) .....	51
5.	結論與未來展望.....	54
(七)	參考文獻.....	55

110 年度科技部大專學生研究計畫結案報告  
基於雲端環境下智慧製造系統之安全通道實作分析  
國立臺灣科技大學 資訊管理系 江忠晏

### (一) 摘要

雲端及智慧製造的興起促使許多中小製造業從原先的本地部署（On-Premises）架構，轉變為雲端架構（Cloud），舉凡基礎設施即服務（Infrastructure as a Service）、平台即服務（Platform as a Service）或軟體即服務（Software as a Service）皆是雲端架構的範疇。然而即使雲端架構具備降低硬體、IT 及部署時間成本、增加靈活性等優點，但勢必也會面臨雲端服務供應商安全、網路傳輸安全等資通安全紕漏。故本研究分析中小製造業從本地部署架構轉型至雲端架構之中的安全架構建置及安全通道傳輸性能。

本研究基於 IaaS 架構，使用 IPsec 建置工廠內部路由器至雲端服務站到站（Site-to-Site）安全通道，其中安全通道採用的對稱式加密演算法為 AES128，雜湊演算法為 SHA1。最後分別地分析站對站（Site-to-Site）及端對端（End-to-End）網路吞吐量（Throughput）及延遲（Latency），並透過開源軟體 Wireshark 進行封包分析，以驗證安全通道安全性。

關鍵詞：安全通道、雲端架構、智慧製造、網路效能

## (二) Abstract

The rise of cloud architecture and smart manufacturing has prompted many small and medium-sized manufacturing industries to shift from on-premises architecture to cloud architecture. For instance, infrastructure as a service (IaaS), platform as a service (PaaS) or software as a service (SaaS) are in the domain of cloud architecture.

Even though cloud architecture has a lot of advantages, like reducing the cost of hardware, IT and deployment time, besides increasing flexibility as well. However, it still must face information security problems, such as cloud service provider security and network transmission security. Therefore, the performance and security of secure channel were analyzed in this research, when transiting from on-premises architecture to cloud architecture for small and medium-sized manufacturing industries.

Based on IaaS architecture, a site-to-site secure channel (IPsec) from the factory router to the cloud service was built, where AES128 was used as the symmetric encryption algorithm and SHA1 was used as the hash function. Finally, throughput and latency of site-to-site and end-to-end network were respectively analyzed in this research, and used opensource software Wireshark to capture packet to verify the security of the secure channel.

Keywords: Secure Channel, Cloud Architecture, Smart Manufacturing, Network Performance

### (三) 前言與研究目的

2020 年全球深受新冠肺炎影響，許多產業開始採取遠距辦公，也有許多產業開始進行數位轉型，台灣的中小製造業也不例外，所幸經濟部從 2018 年起籌組智慧製造輔導團，協助我國中小企業導入智慧製造，提前做好準備。因此現今中小製造業實施智慧製造更是個趨勢，其中雲端技術是不可或缺的一環。

從 2019 年臺灣企業雲端大調查結果中能發現，臺灣的一般製造業上雲比例達到近 50%，其中最常上雲的應用為 24 小時服務或非關鍵性任務的應用。這項統計數據也意味著目前越來越多製造業接觸到雲端服務，將工廠內資料上傳雲端，從傳統的本地部署（On-Premises）架構如圖 3-1，將系統服務（例監控軟體等）置於工廠本地端，轉向雲端架構（Cloud）如圖 3-2，將系統服務部署於雲端伺服器，以利工廠管理階層在外能透過手機經由網際網路監測數據，因此雲端技術亦是架構中重要的一部分[1]，對於普遍企業使用雲端服務的優點不外乎為以下幾點。

#### 1. 降低 IT 預算及時間成本：

前期購入伺服器機台、儲存空間及環境建置，以及後期維持及管理皆需花費大量 IT 預算及時間成本，對於一般中小企業上雲是有一定程度的負擔[1]。此外，對於非 24 小時服務來說，更是一種沉沒成本。因此採取雲端架構不僅能降低 IT 預算及時間成本，亦能避免沉沒成本問題。

#### 2. 具有服務調節及擴展性：

在企業中可以適時的在雲端伺服器中藉由容器化（Container）技術新增、停止或刪除服務實體容器，各取所需，以增加企業服務靈活彈性。

#### 3. 易於系統維護：

以公有雲為例，由於公有雲為供應商服務，因此供應商會定期推出更新，省去自行維護時間。

#### 4. 易於管理控制：

部分企業會將工廠內部生產資料（稼動率、良率、當前完成率）上傳雲端，並設置權限，以利外出的管理階層能夠即時透過網際網路了解廠內狀況[1]，或是透過數據分析了解工廠長期走向，進而擬定新的管理政策。

不過一體兩面，若將資料上傳雲端就容易延伸出網路傳輸安全、資料傳輸完整性、資料外洩等資通安全問題。

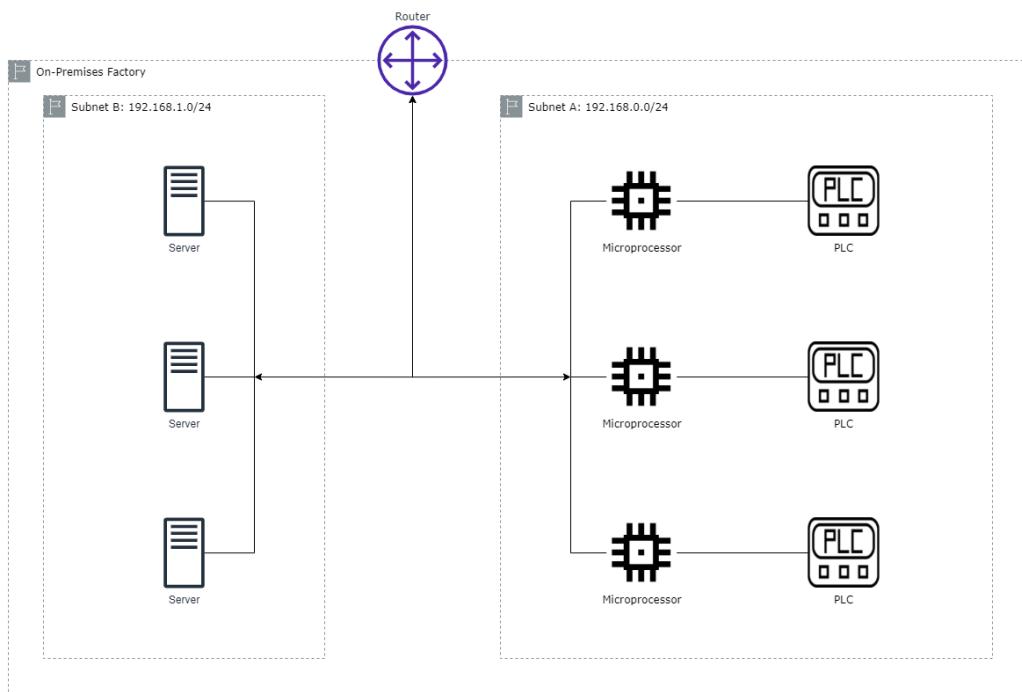


圖 3-1 本地部署（On-Premises）架構

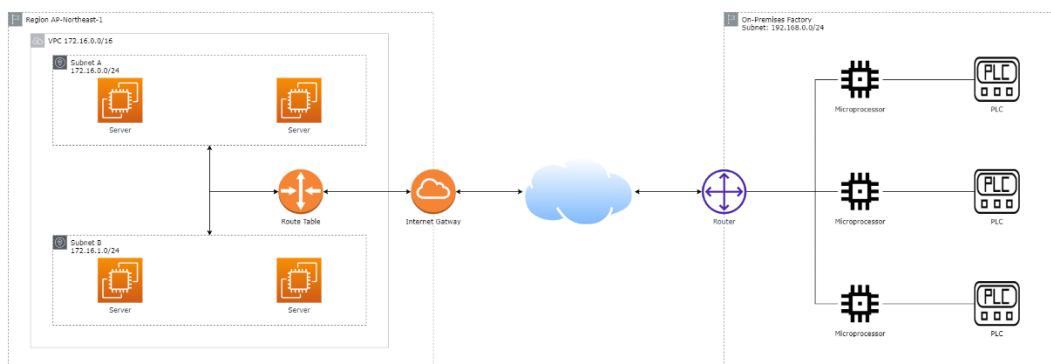


圖 3-2 雲端架構

然而，倘若要將工廠內的傳輸資料進行加密，安全地上傳至雲端，可能會遭遇到許多的難題。例如在製造業環境下，大多工具機為市售 CNC 或 PLC，僅提供資料傳輸功能，不開放修改內部程式，或是有些控制器則因為運算性能等因素無法即時的進行加密運算將傳輸資料加密上傳雲端。

由於網路傳輸安全面向過於廣闊，故本研究僅針對於工廠內部路由器至雲端服務伺服器之間傳輸性能及安全。因此，本研究在（五）研究方法及步驟中，首先探討如何建立工業物聯網、雲端服務伺服器及手機應用程式，以模擬中小製造業上雲環境，再來研究如何利用工廠內部現有的路由器，建立至雲端服務的安全通道（IPsec），最後查究如何於路由器中分析非硬體加速、硬體加速網路吞吐率（Throughput）、延遲（Latency），及驗證安全通道安全性。

## (四) 文獻探討

### 1. 物聯網安全及架構

在這一小節分作為以底層硬體及以系統軟體架構兩個面向。

首先，以物聯網安全底層硬體開始介紹，M. Rao 等人及 J. Kirupakar and S. M. Shalinie 皆提到隨著物聯網蓬勃發展，隱藏的資安疑慮也是需要重視的[2,3]。因此，M. Rao 等人於[2]所做的研究為基於 Virtex-5 及 Virtex-7 FPGA 晶片，實作 IPSec 認證標頭 (AH) SHA-3 雜湊函式。之所以選擇以上兩塊晶片是因為他們高性能的表現，以 Virtex-7 晶片最終達到的頻率為 392MHz。

J. Kirupakar and S. M. Shalinie 於[3]所做的研究為在於工業物聯網環境下，由於現有的邊緣節點裝置(Gateway)為低功率，有著受限的 CPU 及記憶體性能，因此實作輕量化的入侵檢測系統 (Intrusion-detection system)。當外部進行 DDOS 攻擊時，偵測任一邊緣節點的 CPU 使用率、記憶體使用率等相關重要指標，若是相鄰節點不在相對數值範圍內的話便進行入侵警報。

以系統軟體架構的方面，A. Sun 等人於[1]提出了一個基於企業服務匯流排 (Enterprise Service Bus) 所延伸出的雲端服務匯流排 (Cloud Service Bus) SaaS 平台，其主要目的為整合生產商用軟體於他們所提出的架構平台上，例如將 ERP 及 CRM 等軟體部署在平台上，不僅減少中小企業購入、管理及維護等 IT 預算，中小企業更能夠透過網際網路，進行存取部署於雲端生產商用軟體。

### 2. 安全通道建構技術

#### 2.1 虛擬私人網路

使用虛擬私人網路 (Virtual Private Network) 不僅能以廣域網域的便宜價錢，享受到專屬線路的安全，對於建置及維護更為彈性。T. Goethals 等人的研究[4]指出當前邊緣裝置已有足夠的能力運行容器實體及微服務，其中資料的安全是必須重視的，因此透過 VPN 傳輸能確保資料的安全性。為了評估目前市面上 VPN 軟

體的擴充性，他們透過 Kubernetes 分配了大量的 VPN 客戶端的容器實體。從研究結果來看，當 VPN 客戶端小於 250 個，市面上其他 VPN 軟體與 WireGuard 回應時間大抵相同。而當 VPN 客戶端增加到 250 個後，其他軟體的回應時間成對數上升，WireGuard 仍保持低回應時間。然而在中小製造業在雲端服務環境下，VPN 客戶端普遍不會超過 250 個，因此可適情況所需選擇最適合自己工廠 VPN 軟體及技術。

## 2.2 穿隧協議

目前有以下四種主流的穿隧協議 (Tunneling Protocol)，分別為通用路由封裝 (Generic Routing Encapsulation)、端對端隧道協定 (Point to Point Tunneling Protocol)、第二層穿隧協議 (Layer 2 Tunneling Protocol)、網際網路安全協定 (Internet Protocol Security)。

S. Jahan 等人的研究結論[5]提到，GRE 適合於時間敏感的應用程式；IPSec 則適合於安全性敏感的應用程式；PPTP、L2TP 並無加密機制，需搭配其他協議才可進行加密，例 L2TP over IPsec，且 L2TP 需設定使用者端軟體，架設較為麻煩。目前企業所使用的 CPE-based VPN 中，鑑於其安全性，故以 IPsec VPN 為主流。

## 3. IPsec 機制

### 3.1 封裝模式

IPsec 提供了兩種封裝模式 (Encapsulation Mode) [6]可作使用，分別為傳輸模式 (Transport Mode) 及隧道模式 (Tunnel Mode)。傳輸模式建立在兩台主機上，兩邊的主機都要架設 IPsec 協議，而隧道模式則是透過雙方閘道器或路由器進行 IPsec 協議。

### 3.2 安全協議

IPsec 具有兩種安全協議 (Security Protocol) [7,8]，分別為 AH (Authentication

Header) 及 ESP (Encapsulating Security Payload)。AH 及 ESP 都具備資料完整性、資料來源驗證，然而 AH 並不支持資料加密，故此篇著重於 ESP 封裝協議 [8]，而其封包格式如圖 4-1。

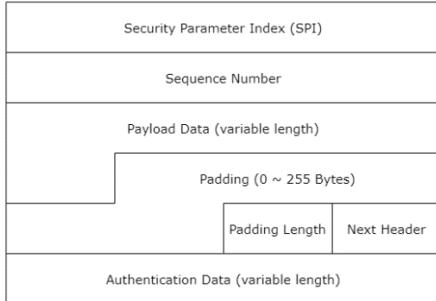


圖 4-1 ESP 封裝協議封包格式

安全參數索引 (Security Parameter Index) 為雙方安全關聯 (Security Association) 的索引值，序號 (Sequence Number) 為防止反重播攻擊，承載資料 (Payload Data) 為將原先 IP 封包所承載的資料，取決於安全關聯參數再進行對應的處理，填補(Padding)為對齊資料長度達到 32 位元整數倍，填補長度(Padding Length)為前者(Padding)長度，下一標頭(Next Header)為承載資料協定(TCP、UDP 等)，驗證資料 (Authentication Data) 則依雙方安全關聯規範範圍決定 (對規範範圍進行雜湊運算)。

### 3.3 安全關聯

IPsec 中最重要的便是安全關聯 (Security Association) [6]，因為它定義雙方要如何協商，並儲存了雙方各種參數，例如要使用何種加密演算法？要使用何種封裝協議？SA 的生命週期多久？而這些安全關聯會放在主機或路由器的安全關聯資料庫 SAD (Security Association Database) 中。倘若當 A 與 B 在進行通訊時，分作成 A 接收端、A 發送端、B 接收端、B 發送端；A 傳送訊息給 B 時，會先進入 A 的 Outbound SAD 中利用上述所提到的 SPI 找尋 Outbound SA-1，將原先的資料封裝成 SA-1 所定義的規範，將封包發送給 B；B 在接收 A 傳送的封包時，會先進入 B 的 Inbound SAD 中利用 SPI 找尋對應的 Inbound SA-1，將資料照 SA-

1 規範解密回原先的資料。

### 3.4 網際網路金鑰交換

網際網路金鑰交換 (Internet Key Exchange) [9]能夠使雙方自主協調，產生 ISAKMP SA 及 IPsec SA，ISAKMP SA 是用以確保私鑰安全，其中參數包括認證方式、加密演算法、雜湊函式等，而 IPsec SA 如 3.3 安全關聯所述為進行通訊時所規範參數。其中，IKE 又分作為 Phase 1 跟 Phase 2，Phase 1 主要目的為雙方互相認證，認證方法有預先共享金鑰(Pre-shared Key)、會議金鑰(Session Key)、公開金鑰(Public Key)，透過雙方金鑰比對進行身分認證，再經由 Diffie-Hellman 演算法(非對稱)產生出另一筆金鑰，以保護 Phase 2 通訊安全

Phase 2 是基於 ISAKMP SA，可以執行多次，產生出多組 IPsec SA，假設 IPsec SA 的生命週期為 3600 秒，代表著每一個小時就會更新一筆 IPsec SA，但這一切都是由 ISAKMP SA 所確保其安全。

### 3.5 實作模式

IPsec 會分作成 kernel space 及 user space 實作，Linux 3.6+ Kernel 的虛擬隧道介面 VTI (Virtual Tunnel Interface) [10]，或是 Linux 4.19+ Kernel 的 XFRM 框架[11]，皆屬於 kernel space，而 OpenVPN 及 IPsec 開源實作軟體 strongSwan 中的 libipsec 則是基於 user space。

雖然 OpenVPN 的技術並不是基於 IPsec 的，不過 OpenVPN 與 IPsec 同屬於 Layer 3 VPN，且運作邏輯與 libipsec 十分相似，文獻及資源相比之下也更多，因此下方會以 OpenVPN 作為類比 libipsec 的舉例。

以 OpenVPN 為例，其實作方式如圖 4-2，是在 kernel space 產生一個虛擬 tun 裝置，封包在傳出時，必須從 kernel space 進行 memory copy 至 user space，再藉由 OpenSSL engine 進行封包處理，處理完後再從 user space 進行 memory copy 回到 kernel space，因此大量的來往的處理會使 memory copy 到達瓶頸，故不適合於路由器上實作。

此外，在 strongSwan 官方文件[12]亦指出 libipsec 不適用於高流量或高突發的情況。而 user space 實作的優點在於倘若裝置的 kernel 並未支援某種加密演算法時，便可以採取 user space 實作安全通道。

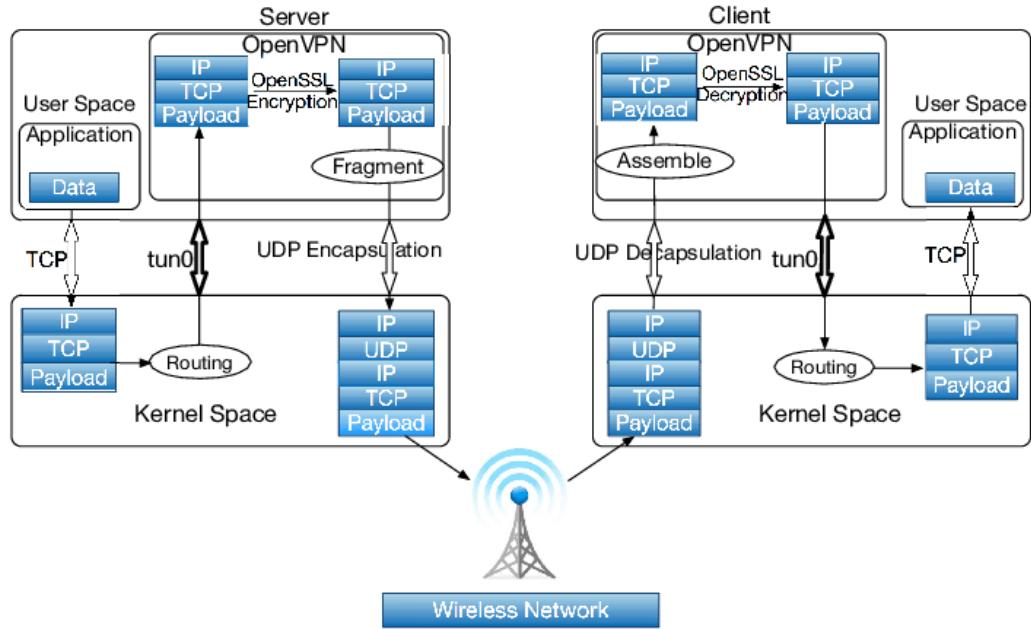


圖 4-2 OpenVPN 實作方式[13]

此外，4.19+ Kernel XFRM 相比 VTI 提出了一個更好的解決方式。首先，XFRM 不再需要設定端點位址，這解決了 wildcard 位址的問題，並避免了介於 SA 及介面的一對一映射，故可更簡易的允許 SA 與多個對點共享介面。再者，如上所述不再需要設定端點位址，這也意味著在同一介面上支援 IPv4 及 IPv6 等優勢。以下將著重介紹 XFRM 框架的流程。

### 3.6 XFRM 框架

根據圖 4-3 所示，在訊息傳送前會先經過 xfrm4\_lookup，其作用為查詢是否有對應的 Outbound SA，若是符合 Outbound SA，則進入 XFRM 框架內根據模式及協議進行處理，最後則如上執行 ip\_output，完成封包發送。

根據圖 4-4 所示，在訊息接收後會先偵測封裝類型，如果封裝類型為 AH、ESP，則會進入 XFRM 框架內查詢對應的 Inbound SA，再透過安全關聯中的模

式及協議進行處理，處理完後將原始訊息擷取出來重回到 ip\_local\_deliver，再經過 XFRM Policy 的過濾，最終將資料送上應用層。

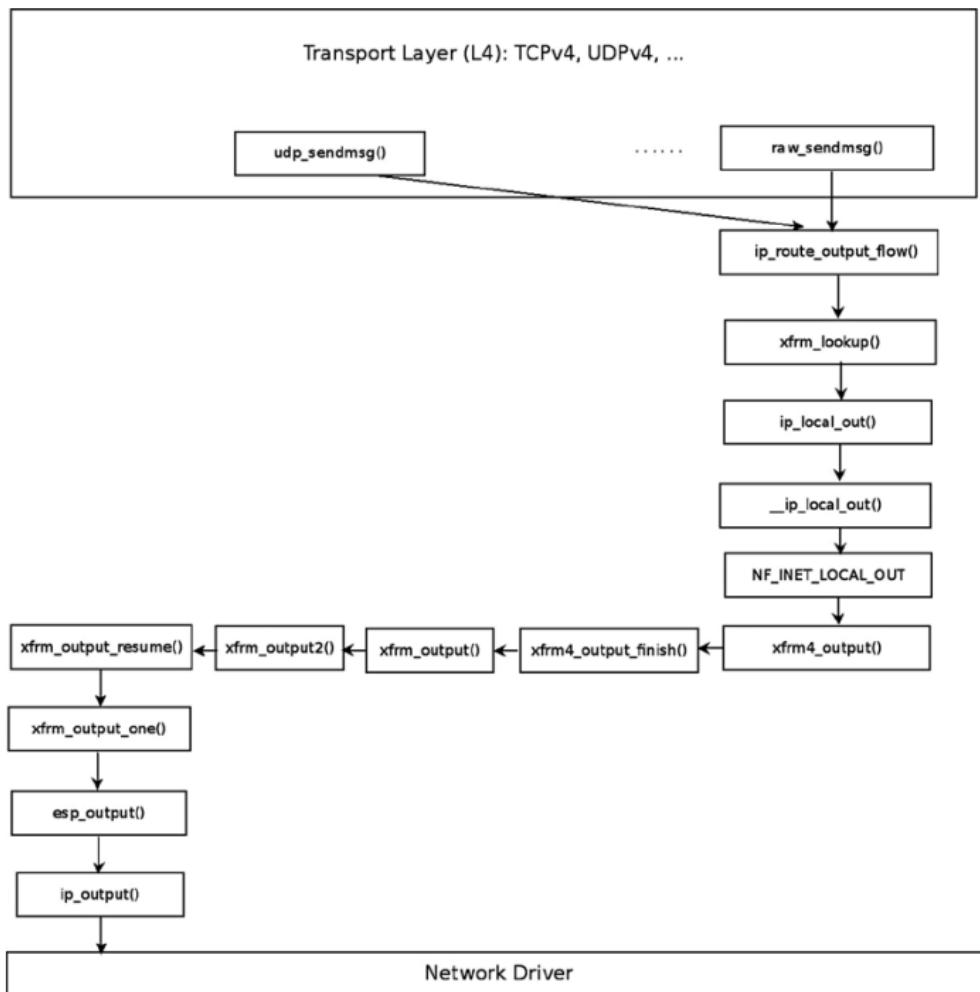


圖 4-3 XFRM 框架封包發送流程圖[14]



圖 4-4 XFRM 框架封包接收流程圖[14]

### 3.7 加密演算法、雜湊函式

上述提到 IPsec ESP 封裝協議的承載資料（Payload Data），會經過加密以確保安全。而加密演算法（Encryption Algorithms）分作為對稱（Symmetric）及非對稱（Asymmetric）兩種，對稱加密利用私鑰加密，私鑰解密，因此會存在著雙方私鑰交換的問題，然而其優點在於運算速度快，需要較少的運算資源。非對稱加密利用公鑰加密，私鑰解密，解決了對稱加密的私鑰交換的問題，但由於金鑰長度過長，所以需要更多的運算資源。

此外，IPsec ESP 封裝協議的認證資料（Authentication Data），雙方會對安全關聯規範範圍使用雜湊函式（Hash Function）進行雜湊運算，得一完整性檢查值（Integrity Check Value），以確保封包資料正確。

本研究著重於具有安全性前提下，探討高效率的加密演算法，或是硬體加密加速（Hardware Cryptographic Acceleration）作為解決方法。

首先從高效率的加密演算法切入，在加密演算法的部分，Saraiva D.A.F 等人於[15]研究結果中勝出的 ChaCha20-Poly1305，以及 D. Ma and Y. Shi 於論文中提出的 SDN\_IIOT\_EN [16]，經實驗測試後，其安全性皆與 AES256 不相上下，其中 ChaCha20-Poly1305 速度快於 AES256 將近三倍，SDN\_IIOT\_EN 則快於 AES 將近五倍。

而於雜湊函式的部分，T. Goethals 等人所述[4]，WireGuard 所採用的雜湊函式為 BLAKE2，不僅安全性優於 SHA-2，速度更快於 MD5。因此在具有安全性前提下，BLAKE 相當適合在於性能受限的路由器中實作。

#### 4. 硬體加速

在經過多方涉略之後，發現到現在在不同領域皆有做硬體加速（Hardware Acceleration）的趨勢，好比前陣子 Google 所提出的 TPU( Tensor Processing Unit ) [17]，專門用於加速機器學習的速度，抑或是安全通道、近期流行的加密貨幣、區塊鏈所需的硬體加密加速（Hardware Cryptographic Acceleration），專門用於加速加密演算法、雜湊演算法的速度。而硬體加密加速器的原理為在原先的指令架構（Instructions Set Architecture）進行擴充，如 x86 架構下的 AES-NI 及 ARM 架構下的 ARMv8 Cryptography Extensions。

在 Intel Advanced Encryption Standard New Instructions Set White Paper[18]中提到，其 AES-NI 包含了六個指令如圖 4-5，其中四個指令分別為 AESENC ( AES Encrypt Round )、AESENCLAST ( AES Encrypt Last Round )、AESDEC ( AES Decrypt

Last Round) 及 AESDECLAST (AES Decrypt Last Round) 作為資料加解密，另外兩個指令分別為 AESIMC (AES Inverse Columns) 及 AESKEYGENASSIST (AES Key Generation Assist) 作為協助 AES Key 的展開。

## Intel® AES New Instructions Architecture

The Intel AES New Instructions consists of six instructions.

Four instructions, namely AESENC, AESENCLAST, AESDEC, AESDECLAST, are provided for data encryption and decryption (the names are short for AES Encrypt Round, AES Encrypt Last Round, AES Decrypt Round AES Decrypt Last Round). These instructions have both register-register and register-memory variants.

Two other instructions, namely AESIMC and AESKEYGENASSIST are provided in order to assist with AES Key Expansion (the names are short for AES Inverse Mix Columns, and AES Key Generation Assist).

圖 4-5 Intel AES-NI 指令[18]

在 ARM Architecture Reference Manual for A-profile architecture[19] 中提到，CE (Cryptographic Extension) 包含如圖 4-6 所示指令，主要有 AES 運算處理、SHA 運算處理等指令，這些皆能夠加速加密演算法及雜湊函式。如圖 4-7，以 AESE 指令為例，我們能在 Assembler symbols 發現到 Vd 代表 SIMD&FP 來源端及目的端暫存器，Vn 則代表 SIMD&FP 來源端暫存器。

Table C3-111 shows the Armv8.0 Cryptographic Extension instructions.

Table C3-111 Cryptographic Extension instructions

Mnemonic	Instruction	See
AESD	AES single round decryption	<i>AESD</i> on page C7-2029
AESE	AES single round encryption	<i>AESE</i> on page C7-2030
AESIMC	AES inverse mix columns	<i>AESIMC</i> on page C7-2031
AESMC	AES mix columns	<i>AESMC</i> on page C7-2032
PMULL	Polynomial multiply long	<i>PMULL, PMULL2</i> on page C7-2513 <sup>a</sup>
SHA1C	SHA1 hash update (choose)	<i>SHA1C</i> on page C7-2562
SHA1H	SHA1 fixed rotate	<i>SHA1H</i> on page C7-2563
SHA1M	SHA1 hash update (majority)	<i>SHA1M</i> on page C7-2564
SHA1P	SHA1 hash update (parity)	<i>SHA1P</i> on page C7-2565
SHA1SU0	SHA1 schedule update 0	<i>SHA1SU0</i> on page C7-2566
SHA1SU1	SHA1 schedule update 1	<i>SHA1SU1</i> on page C7-2567
SHA256H	SHA256 hash update, part 1	<i>SHA256H</i> on page C7-2569
SHA256H2	SHA256 hash update, part 2	<i>SHA256H2</i> on page C7-2568
SHA256SU0	SHA256 schedule update 0	<i>SHA256SU0</i> on page C7-2570
SHA256SU1	SHA256 schedule update 1	<i>SHA256SU1</i> on page C7-2571

a. The Cryptographic Extension adds the variant of the instruction that operates on two 64-bit polynomials.

圖 4-6 ARM-CE 指令[19]

### **AESE**

AES single round encryption.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9   5 4   0	D
0 1 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 Rn Rd	

#### **Encoding**

AESE <Vd>.16B, <Vn>.16B

#### **Decode for this encoding**

```
integer d = UInt(Rd);
integer n = UInt(Rn);
if !HaveAESExt() then UNDEFINED;
```

#### **Assembler symbols**

<Vd> Is the name of the SIMD&FP source and destination register, encoded in the "Rd" field.  
 <Vn> Is the name of the second SIMD&FP source register, encoded in the "Rn" field.

#### **Operation**

```
AArch64.CheckFPAdvSIMDEnabled();

bits(128) operand1 = V[d];
bits(128) operand2 = V[n];
bits(128) result;
result = operand1 EOR operand2;
result = AESSubBytes(AESShiftRows(result));

V[d] = result;
```

圖 4-7 ARM-CE AESE 指令[19]

在 Processing Multiple Buffers in Parallel to Increase Performance on Intel® Architecture Processors[20]中提到，雖然 x86 架構下的 AES-NI 是被定義在 SSE ( Streaming SIMD Extensions ) 指令中，但它並不是 SIMD ( Single Instruction Multiple Data ) 的一種，特別是 AESENCL 這個指令在某些模式下是能夠平行處理的，像是 counter-mode，但在 CBC 模式下卻是需要序列實作的。

ARM 架構下的 ARM Cryptographic Extension Instructions 則是使用 SIMD 技術進行加密演算法、雜湊函式加速。SIMD 一詞在 Computer Organization and Design RISC-V Edition[21]一書中的定義為同樣的指令被應用於許多資料流，如同於一個向量處理器。

若以 ARM64 作為舉例的話，圖 4-8 左方為 SISD 運算，每一個 PC 只能執行一次 64 bits x 64 bits 的指令，倘若今天需要運算四組 16 bits x 16 bits 時，在 PC+1 的時候會將 16 bits 延伸至 64 bits，再進行 64 bits x 64 bits 運算，在 PC+2、PC+3、PC+4 的時候也如同 PC+1 時進行一樣的步驟，因此在每次的運算中皆會浪費掉

48 bits。圖 4-8 右方為 SIMD 運算，如上同樣的問題，倘若今天需要運算四組 16 bits x 16 bits 時，可以將四組 16 bits 併成 64 bits 於 SIMD 指令下進行同步運算，故在 PC+1 時便可以完成運算。

綜括以上所述，SISD 能夠在一個指令下運算一個資料流，而 SIMD 能夠在一個指令下運算多個資料流，而加密演算法、雜湊函式的運算流程皆符合 SIMD 的概念，因此目前普遍的硬體加密加速器使用 SIMD 技術為大宗。

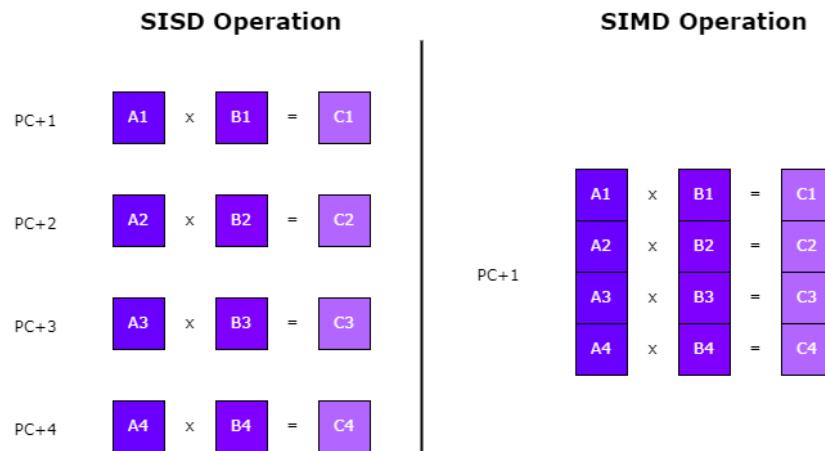


圖 4-8 SISD 運算及 SIMD 運算

比較 Linux AES-CBC 原始碼如圖 4-9，與 ARM-CE AES-CBC 原始碼如圖 4-10、圖 4-11 及圖 4-12，我們可以發現到 Linux AES-CBC 是純軟實作的，而 ARM-CE AES-CBC 則會調用 ARM-CE 的指令 aes-cbc-encrypt。

```

static int crypto_cbc_encrypt(struct skcipher_request *req)
{
    struct crypto_skcipher *skcipher = crypto_skcipher_reqtfm(req);
    struct skcipher_walk walk;
    int err;

    err = skcipher_walk_virt(&walk, req, false);

    while (walk.nbytes) {
        if (walk.src.virt.addr == walk.dst.virt.addr)
            err = crypto_cbc_encrypt_inplace(&walk, skcipher);
        else
            err = crypto_cbc_encrypt_segment(&walk, skcipher);
        err = skcipher_walk_done(&walk, err);
    }

    return err;
}

```

圖 4-9 Linux AES-CBC 原始碼[22]

```

static int crypto_cbc_encrypt_segment(struct skcipher_walk *walk,
                                      struct crypto_skcipher *skcipher)
{
    unsigned int bsize = crypto_skcipher_blocksize(skcipher);
    void (*fn)(struct crypto_tfm *, u8 *, const u8 *);
    unsigned int nbytes = walk->nbytes;
    u8 *src = walk->src.virt.addr;
    u8 *dst = walk->dst.virt.addr;
    struct crypto_cipher *cipher;
    struct crypto_tfm *tfm;
    u8 *iv = walk->iv;

    cipher = skcipher_cipher_simple(skcipher);
    tfm = crypto_cipher_tfm(cipher);
    fn = crypto_cipher_alg(cipher)->cia_encrypt;

    do {
        crypto_xor(iv, src, bsize);
        fn(tfm, dst, iv);
        memcpy(iv, dst, bsize);

        src += bsize;
        dst += bsize;
    } while ((nbytes -= bsize) >= bsize);

    return nbytes;
}

```

圖 4-10 ARM-CE AES-CBC 原始碼[23]

```

static int __maybe_unused cbc_encrypt(struct skcipher_request *req)
{
    struct skcipher_walk walk;
    int err;

    err = skcipher_walk_virt(&walk, req, false);
    if (err)
        return err;
    return cbc_encrypt_walk(req, &walk);
}

```

圖 4-11 ARM-CE AES-CBC 原始碼[23]

```

static int cbc_encrypt_walk(struct skcipher_request *req,
                           struct skcipher_walk *walk)
{
    struct crypto_skcipher *tfm = crypto_skcipher_reqtfm(req);
    struct crypto_aes_ctx *ctx = crypto_skcipher_ctx(tfm);
    int err = 0, rounds = 6 + ctx->key_length / 4;
    unsigned int blocks;

    while ((blocks = (walk->nbytes / AES_BLOCK_SIZE))) {
        kernel_neon_begin();
        aes_cbc_encrypt(walk->dst.virt.addr, walk->src.virt.addr,
                        ctx->key_enc, rounds, blocks, walk->iv);
        kernel_neon_end();
        err = skcipher_walk_done(walk, walk->nbytes % AES_BLOCK_SIZE);
    }
    return err;
}

```

圖 4-12 ARM-CE AES-CBC 原始碼[23]

## (五) 研究方法及步驟

本研究著眼於中小製造業從本地部署架構轉變為雲端架構如圖 5-1，所將遭遇到的傳輸安全問題，研究方法分成以下四步驟進行。

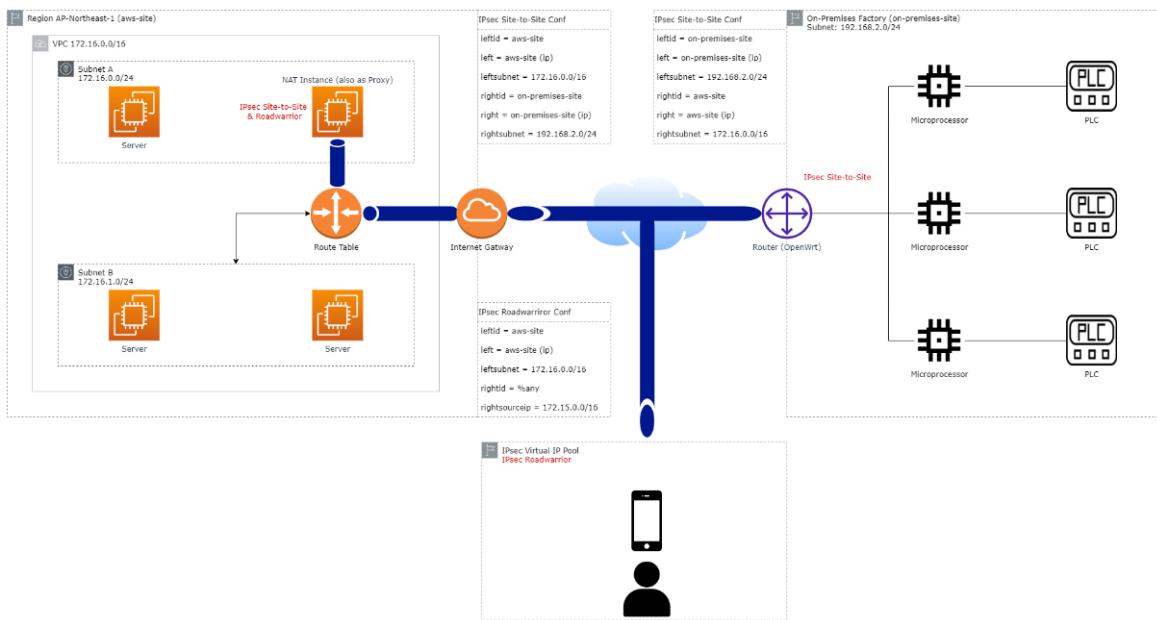


圖 5-1 中小製造業雲端架構模擬設計

### 1. 上雲架構模擬設計

#### 1.1 工業物聯網架構流程

在「工業物聯網架構」中，由於中小製造業普遍所使用的市售 CNC 及 PLC 控制器售價不菲，故本研究採用 LinkIt 7688 開發版以模擬工業控制器。之所以使用 LinkIt 7688 開發版，是因為其軟體開發環境基於 OpenWrt 作業系統，預設安裝了許多物聯網裝置開發常用的套件，包含多種程式語言的支援，例如 Python、Node.js 及 C 語言[24]。因此，本研究在 LinkIt 7688 開發版上，撰寫簡單的定時獲取系統時間程式如圖 5-2，並上傳資料到雲端服務中的資料存取伺服器，以模擬工業物聯網中將資料定時上傳模式。

在使用 LinkIt 7688 開發版時，必須先經由 AP Mode 連接至開發版，並且利用 OpenWRT CLI 介面更改網路介面，例如 WAN 的網路設定如圖 5-3、實體層設定如圖 5-4，以及 LAN 實體層如圖 5-5 設定，藉由此流程設定後，便可以於路由器設置如圖 5-6，看見 LinkIt 7688 此裝置如圖 5-7。

```
from urllib import request, parse
import time
import datetime

url = 'http://172.16.2.116'
while True:
    current_time = datetime.datetime.now()
    post_time = {'time' : str(current_time)}
    print(post_time)
    data = parse.urlencode(post_time).encode()
    req = request.Request(url, data)
    resp = request.urlopen(req)
    print(resp.read())
    time.sleep(1)
```

圖 5-2 定時獲取系統時間及上傳程式（Python）

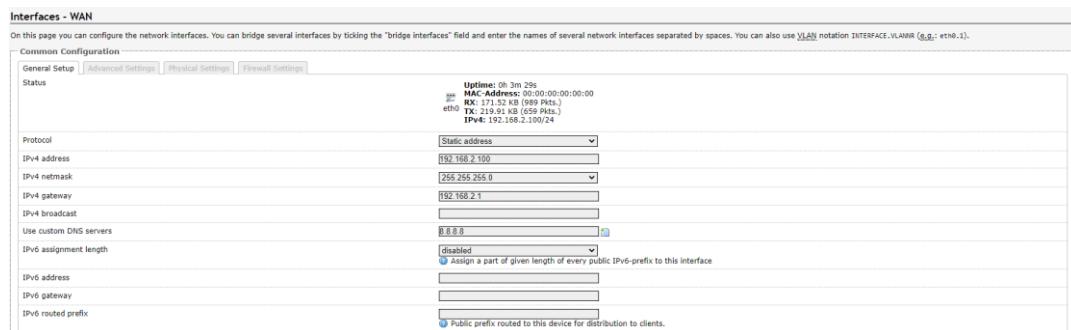


圖 5-3 LinkIt 7688 WAN 網路設定

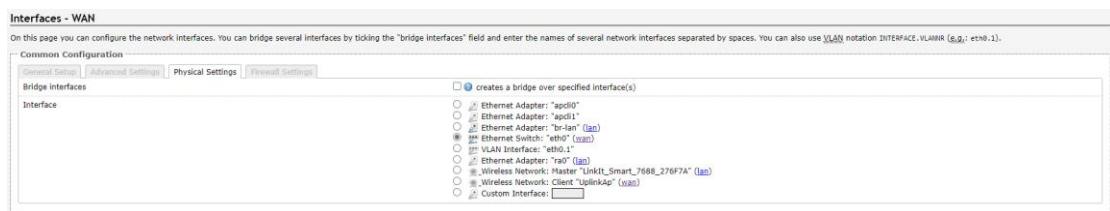


圖 5-4 LinkIt 7688 WAN 實體層設定

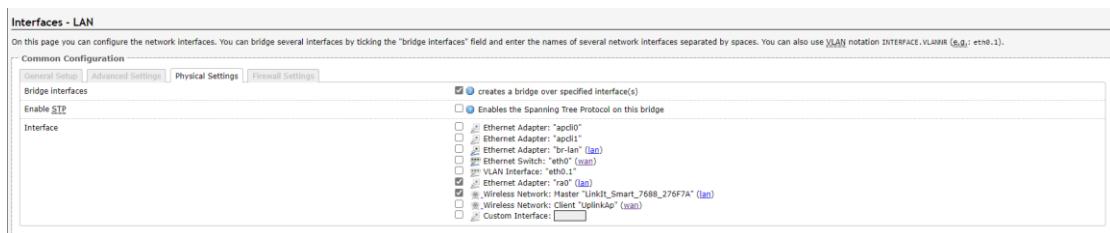


圖 5-5 LinkIt 7688 LAN 實體層設定

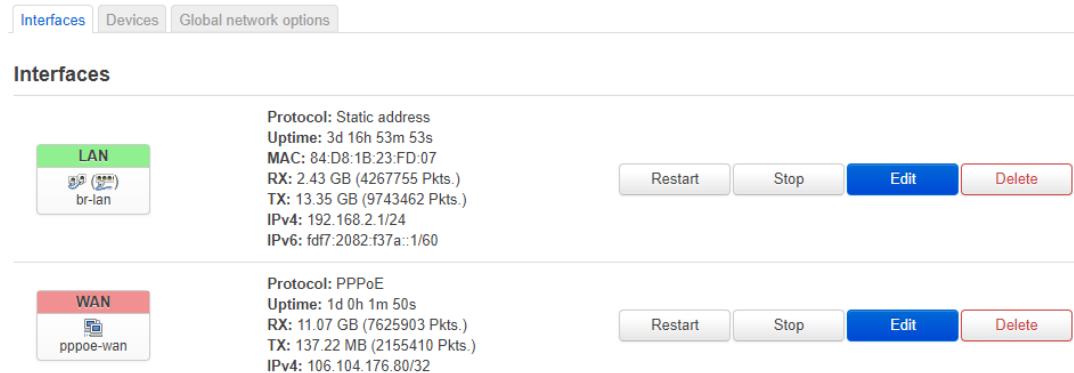


圖 5-6 OpenWRT 路由器設置

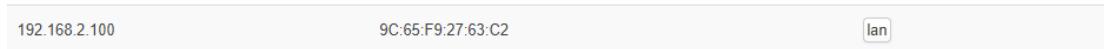


圖 5-7 路由器中的 LinkIt 7688

## 1.2 雲端服務架構流程

在「雲端服務架構」中，本研究採用 Amazon Web Services 以作為雲端服務架構的模擬。首先建立 VPC (Virtual Private Cloud) 雲端環境如圖 5-8，且在 VPC 中配置 Internet Gateway 如圖 5-9 以連通內外雙向網路，再分別於兩個 Subnet 中建立一 EC2 Instance 作為 NAT 如圖 5-10，Subnet 如圖 5-11，安全群組如圖 5-12，另一 EC2 Instance 作為資料存取伺服器如圖 5-13，Subnet 如圖 5-14，安全群組如圖 5-15。

而在 AWS 中將 EC2 Instance 作為 NAT 時，必須注意以下幾個事項，(a) 關閉來源／目的地檢查如圖 5-16，(b) iptables 設定如圖 5-17，方得使 Subnet B 連上網路。

再來將不屬於 NAT Subnet 的流量先路由至 NAT Instance 上如圖 5-17，再將 NAT Instance 路由至 Internet Gateway 上如圖 5-18，之所以需要 EC2 Instance 作為 NAT 的原因，而非 AWS 所提供的 NAT Gateway，原因在於 NAT Gateway 為 AWS 所提供的軟體即服務，故無法在上面進行分析，因此必須採用基礎設施即服務安裝作業系統，作為 NAT 進行分析。

至於資料存取伺服器，使用 Docker 如圖 5-19 及圖 5-20、Flask 如圖 5-21 及

## MongoDB 建立具有容器化的資料存取伺服器。

This screenshot shows the AWS VPC Details page for the VPC ID: vpc-09233a1526a416e86. The page includes tabs for Detailed Information, CIDR, Flow Log, and Tags. The Detailed Information section displays various network settings such as status (Available), DNS settings, and security groups.

VPC ID	状态	DNS 主机名	DNS 解析
vpc-09233a1526a416e86	Available	已停用	已启用
租用	DHCP 选项集	主路由器	主要路由器 ACL
Default	dopt-10bb5476	rtb-0576ef418a928e3e2 / most-tunnel-nat-to-ig	acl-02d26bfc64d5d8fa
所属 VPC	IPv4 CIDR	IPv6 集合	IPv6 CIDR (网路边界群组)
否	172.16.0.0/16	-	-
Route 53 Resolver DNS 防火牆规则群组	端点 ID		
-	180529907447		

圖 5-8 AWS VPC 詳細資訊

This screenshot shows the AWS Internet Gateway Details page for the Internet Gateway ID: igw-05e68ee1c524791d6. It displays basic information like the VPC it belongs to and its status.

網際網路閘道 ID	狀態	VPC ID	擁有者
igw-05e68ee1c524791d6	Attached	vpc-09233a1526a416e86   most-tunnel-vpc	180529907447

圖 5-9 AWS Internet Gateway 詳細資訊

This screenshot shows the AWS EC2 Instance (NAT) Details page for the Instance ID: i-064ed092a959917e9. It provides detailed information about the instance's network configuration, including IPv4 and IPv6 addresses, security groups, and IAM roles.

執行個體 ID	私有 IPv4 地址	私有 IPv4 地址
i-064ed092a959917e9 (most-tunnel-nat-instance)	3.114.126.71   開放地址	172.16.1.52
IPv6 地址	執行個體狀態	公有 IPv4 DNS
-	執行中	-
主機名稱類型	私有 IP DNS 名稱 (僅限 IPv4)	回音私有資源 DNS 名稱
IP 名稱: ip-172-16-1-52.ap-northeast-1.compute.internal	ip-172-16-1-52.ap-northeast-1.compute.internal	IPv4 (A)
執行個體類型	彈性 IP 地址	VPC ID
t3.medium	3.114.126.71 [公有 IP]	vpc-09233a1526a416e86 (most-tunnel-vpc)
AWS Compute Optimizer 發現項目	IAM 角色	子網路 ID
①選擇使用 AWS Compute Optimizer 活得建議。   進一步了解	-	subnet-00fe1ae5b311a1ad9 (most-tunnel-public-subnet)

圖 5-10 AWS EC2 Instance (NAT) 詳細資訊

This screenshot shows the AWS Subnet A (NAT) Details page for the Subnet ID: subnet-00fe1ae5b311a1ad9. It displays subnet-specific details like CIDR, route tables, and associated VPCs.

子網路 ID	子網路 ARN	狀態	IPv4 CIDR
subnet-00fe1ae5b311a1ad9	arn:aws:ec2:ap-northeast-1:180529907447:subnet/subnet-00fe1ae5b311a1ad9	Available	172.16.1.0/24
可用的 IPv4 地址	IPv6 CIDR	可用區域	可用區域 ID
250	-	ap-northeast-1d	ap-northeast-1az2
網路邊界群組	VPC	路由表	網路 ACL
ap-northeast-1	vpc-09233a1526a416e86   most-tunnel-vpc	rtb-0576ef418a928e3e2   most-tunnel-nat-to-ig	acl-02d26bfc64d5d8fa

圖 5-11 AWS Subnet A (NAT) 詳細資訊

傳入規則 (6)

類型	通訊協定	連接埠...	來源	描述
所有 TCP	TCP	0 - 65535	106.104.176.80/32	iPerf3/Netperf without...
SSH	TCP	22	0.0.0.0/0	SSH
自訂 UDP	UDP	500	0.0.0.0/0	ISAKMP
所有 TCP	TCP	0 - 65535	192.168.2.0/24	iPerf3/Netperf with IP...
自訂 UDP	UDP	4500	0.0.0.0/0	IPsec with UDP (NAT-T)
所有流量	全部	全部	172.16.0.0/16	VPC

圖 5-12 AWS Subnet A (NAT) 安全群組

執行個體 : i-062d87087c0bb9289 (most-tunnel-server-instance)

詳細資訊 | 安全性 | 聰網 | 儲存 | 狀態檢查 | 監控 | 標籤

▼ 執行個體摘要 | 資訊

執行個體 ID i-062d87087c0bb9289 (most-tunnel-server-instance)	公有 IPv4 地址 -	私有 IPv4 地址 172.16.2.116
IPv6 地址 -	執行個體狀態 執行中	公有 IPv4 DNS -
主機名稱類型 IP 名稱 : ip-172-16-2-116.ap-northeast-1.compute.internal	私有 IP DNS 名稱 (僅限 IPv4) ip-172-16-2-116.ap-northeast-1.compute.internal	回答私有資源 DNS 名稱 -
執行個體類型 t2.micro	彈性 IP 地址 -	VPC ID vpc-09233a1526a416e86 (most-tunnel-vpc)
AWS Compute Optimizer 發現項目 ①選擇使用 AWS Compute Optimizer 指定建議。   <a href="#">進一步了解</a>	IAM 角色 -	子網路 ID subnet-01211023d6089f153 (most-tunnel-server-subnet)

圖 5-13 AWS EC2 Instance (資料存取伺服器) 詳細資訊

subnet-01211023d6089f153 / most-tunnel-server-subnet

詳細資訊 | 流程日誌 | 路由表 | 請路 ACL | CIDR 保留 | 共用 | 標籤

詳細資訊

子網路 ID subnet-01211023d6089f153	子網路 ARN arn:aws:ec2:ap-northeast-1:180529907447:subnet/subnet-01211023d6089f153	狀態 Available	IPv4 CIDR 172.16.2.0/24
可用的 IPv4 地址 250	IPv6 CIDR -	可用區域 ap-northeast-1d	可用區域 ID ap-ne1-az2
網路邊界群組 ap-northeast-1	VPC vpc-09233a1526a416e86   most-tunnel-vpc	路由表 rtb-063fea6580517537c   most-tunnel-rt-to-nat	請路 ACL acl-02d26bfce64d5d8fa
預設子網路		自動指派 IPv6 地址	自動指派客戶擁有的 IPv4 地址

圖 5-14 AWS Subnet B (資料存取伺服器) 詳細資訊

傳入規則 (4)

類型	通訊協定	連接埠...	來源	描述
HTTP	TCP	80	192.168.2.0/24	HTTP Secure Channel ...
所有 TCP	TCP	0 - 65535	192.168.2.0/24	iPerf3/Netperf Secure ...
HTTP	TCP	80	172.15.0.0/16	HTTP Secure Channel ...
所有流量	全部	全部	172.16.0.0/16	VPC

圖 5-15 AWS Subnet B (資料存取伺服器) 安全群組



圖 5-14 關閉來源／目的地檢查

```
ubuntu@ip-172-16-1-52:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               anywhere
ACCEPT    all  --  172.16.0.0/16          anywhere
ACCEPT    all  --  anywhere             172.16.0.0/16      state RELATED,ESTABLISHED
                                         destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

ubuntu@ip-172-16-1-52:~$ sudo iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE all  --  anywhere           anywhere
MASQUERADE all  --  anywhere           anywhere
MASQUERADE all  --  172.16.0.0/16       anywhere
```

圖 5-15 iptables 設置

rtb-063fea6580517537c / most-tunnel-rt-to-nat	
詳細資訊	路由
<a href="#">編輯路由</a>	
目的地	目標
172.16.0.0/16	local
0.0.0.0/0	eni-09f2e8d5719593e0e
狀態	
172.16.0.0/16	作用中
0.0.0.0/0	作用中
已傳播	
172.16.0.0/16	否
0.0.0.0/0	否

圖 5-17 Route Table ( 路由至 NAT Instance )

rtb-0576ef418a928e3e2 / most-tunnel-nat-to-ig	
詳細資訊	路由
<a href="#">編輯路由</a>	
目的地	目標
172.16.0.0/16	local
0.0.0.0/0	igw-05e68ee1c524791d6
狀態	
172.16.0.0/16	作用中
0.0.0.0/0	作用中
已傳播	
172.16.0.0/16	否
0.0.0.0/0	否

圖 5-18 Route Table ( 路由至 Internet Gateway )

```
FROM      python:3.8
ADD      . /most
WORKDIR  /most
RUN      pip3 install -r requirements.txt
```

圖 5-19 Dockerfile

```

database:
  image: mongo:4.2
  ports:
    - "27017:27017"
environment:
  -MONGO_INITDB_ROOT_USERNAME: most
  -MONGO_INITDB_ROOT_PASSWORD: 1234
  -MONGO_INITDB_DATABASE: most
  command: mongod --auth
web:
  build: .
  command: python3 server.py
  ports:
    - "80:80"
  volumes:
    - .:/most
links:
  - database

```

圖 5-20 docker-compose.yml

```

from flask import Flask, request
from pymongo import MongoClient

uri = "mongodb://database:27017"
client = MongoClient(uri)
db = client['most']
col = db['7688']

app = Flask(__name__)

@app.route("/", methods=['POST'])
def insert_time():
    time = request.values['time']
    print(time)
    col.insert_one({'time': time})
    return "Success " + time

@app.route("/", methods=['GET'])
def get_time():
    html_str = "<h1>LinkIt 7688</h1>"
    for time in col.find():
        html_str += time['time'] + "<br>"
    return html_str

if __name__ == '__main__':
    app.debug = True
    app.run('0.0.0.0', 80)

```

圖 5-21 Flask 程式碼

## 2. 安全通道研究方法

原先提出的研究方法為依照工廠內部資料的機密程度及即時性權衡之下，實作不同的加密演算法及雜湊演算法，其中包括 None、XOR Cipher、ChaCha20-Poly1305 及 SDN\_IIOT\_EN 為加密演算法，None、BLAKE2 及 SHA3 作為雜湊演算法。

不過經過近期多方涉略及（四）文獻回顧探討之後，權衡之下，我認為將研究分析的方向從實作加密演算法、雜湊函式，轉變為分析硬體加密加速，有以下幾個原因：

### 2.1 自定義演算法為小眾

首先，中小製造業的 IT 人手大多並不足，這些 IT 人手不僅需要維護 IT 基礎設施、應用程式的穩定及網路安全等，還需要依照公司開發所需的應用程式。倘若公司需自定義演算法，必須聘請懂得自定義演算法的人員，這又是一個額外的成本開銷。

舉凡若需要高吞吐率，不見得需要從演算法的優化或實作進行切入，可以藉由硬體加密加速的方式，或許能達到高吞吐率的目標。況且，較冷門的演算法可能不會有長期支援（Long-term support）或漏洞維護，主流的演算法效能可能不

及其他研究提出的演算法，不過也就是因為主流才經得起資訊安全的考驗，以致於有足夠的安全性。

## 2.2 符合當今硬體趨勢

相對起實作其他研究提出的加密演算法、雜湊函式，我認為進行硬體加密加速更能夠符合當今研究硬體優化的趨勢。在（四）文獻回顧探討時，發現無論是 Google 的 TPU，或是硬體加密加速普遍所使用的 SIMD 技術，皆是在硬體上進行優化。

## 2.3 提供產業更好的解決方式

在中小製造業中已有上雲的趨勢，或是其他產業仍採取遠距辦公的型態，綜括以上皆需使用到 VPN 的技術。而相對於實作演算法以提高吞吐率，使用硬體加密加速的部署成本是相對低廉的。況且，市面上有著許多專門用於加速加密演算法的路由器及閘道器。

## 3. 路由器作業系統及硬體加密加速分析

除了上雲架構模擬設計及安全通道建置以外，這部分進行硬體加密加速分析，工廠本地端的路由器分別採用 TP-Link Archer C7 AC1750 作為常見的路由器，以及 Banana PI R64 作為分析非硬體加密加速及硬體加密加速的差別。在這兩台路由器上安裝 OpenWRT 作業系統[25]，並透過 OpenSSL[26]驗證硬體加密加速是否開啟及效能差異，以作為加速安全通道研究基礎。

對於 TP-Link Archer C7 AC1750 來說，僅需要至 OpenWRT 官方網站亦能找到映像檔。然而對於 Banana PI R64 而言，需要自行編譯作業系統的映像檔，若是要開啟硬體加密加速的話，需在 kernel\_menuconfig 進行設定如圖 5-22 及圖 5-23，先開啟 ARM64 Accelerated Cryptographic Algorithms，再開啟 ARMv8 Crypto Extensions 及 ARM-NEON 演算法。在開啟硬體加密加速後，可以透過 cat /proc/crypto 觀察支援的演算法，其中 priority 越高，代表 kernel 會優先使用。

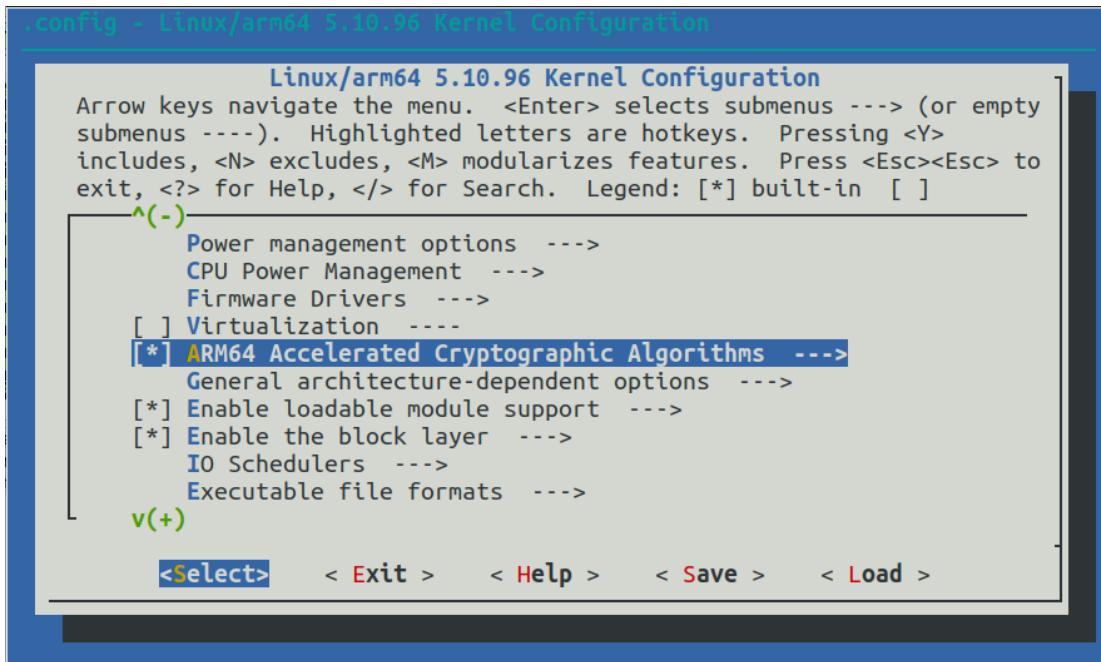


圖 5-22 OpenWRT 透過 make kernel\_menuconfig 指令所開啟的設定介面

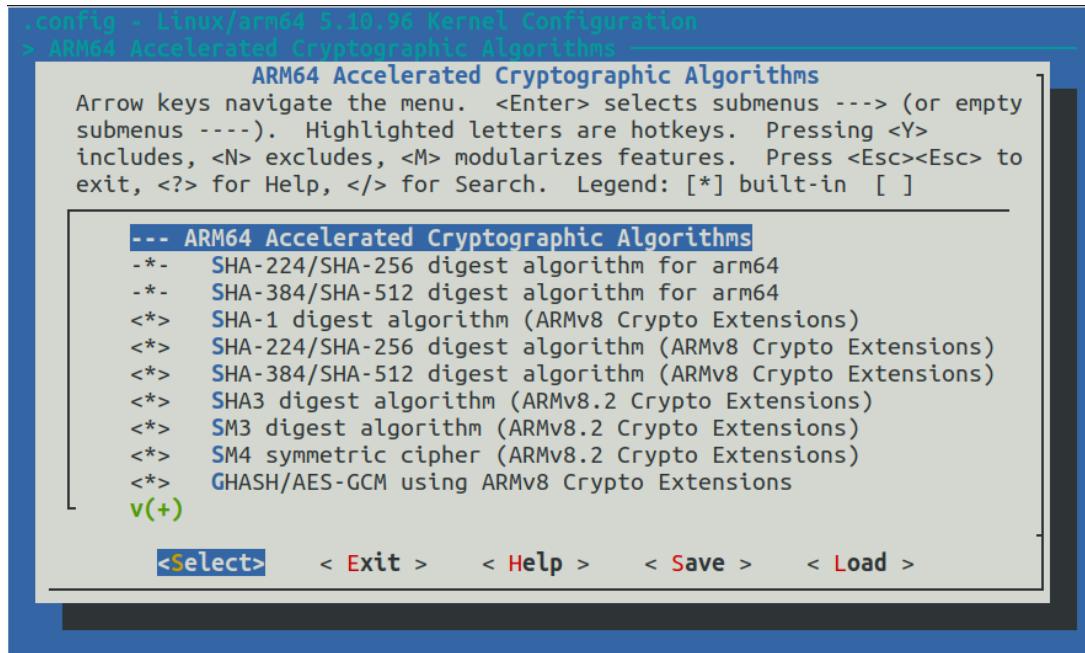


圖 5-22 OpenWRT 透過 make kernel\_menuconfig 指令所開啟的設定介面

```

name          : __cbc(aes)
driver        : __cbc-aes-ce
module        : kernel
priority      : 300
refcnt        : 1
selftest      : passed
internal      : yes
type          : skcipher
async         : no
blocksize     : 16
min keysize   : 16
max keysize   : 32
ivsize        : 16
chunksize     : 16
walksize      : 16

```

圖 5-24 透過 cat /proc/crypto 指令顯示支援的演算法（擷取 cbc-aes 片段）

#### 4. 安全通道建置及分析

本研究透過 strongSwan 配置工廠本地端至雲端服務端 site-to-site 安全通道 (IPsec)，以及雲端服務端至行動裝置端的 Roadwarrior 安全通道 (IPsec)，而安全通道分為「工廠本地端」、「雲端服務端」及「行動裝置端」實作，其拓撲如圖 5-25 所示。

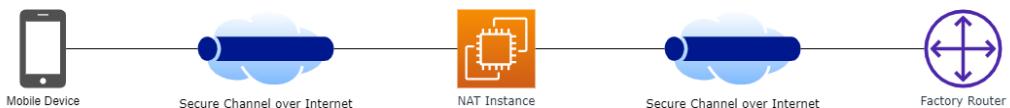


圖 5-25 安全通道網路拓樸

##### 4.1 工廠本地端

工廠本地端路由器採用的作業系統為 OpenWRT 21.02，所需的套件包含 strongswan 以及 ip-full，strongSwan 為實作 IPsec XFRM 框架的開源軟體，而 ip-full 則包含了 Linux Kernel XFRM 框架的實現。

strongSwan 的配置主要為 ipsec.conf 以及 ipsec.secrets 兩個檔案，ipsec.conf

如圖 5-26 包含網際網路金鑰交換協議、ESP 封裝參數、來源 IP、目的地 IP 等安全通道參數，而 ipsec.secrets 如圖 5-27 則是設置安全通道預共享金鑰（Pre-shared key）。

在雙方配置完成後，利用 ipsec up {id} 的方式啟動安全通道，便會把 ipsec.conf 及 ipsec.secrets 等配置編譯後注入 XFRM 底層如圖 5-28 及圖 5-29，而 ipsec statusall 表示安全通道的狀態如圖 5-30。

```
conn %default
      keyexchange=ikev2
      authby=secret

conn aws
      ike=aes-sha1-curve25519!
      esp=aes-sha1-curve25519!
      leftid=wrt
      left=192.168.2.1
      leftsubnet=192.168.2.0/24
      rightid=aws
      right=3.114.126.71
      rightsubnet=172.16.0.0/16
      auto=route
```

圖 5-26 IPsec 參數配置檔（工廠本地端）

```
wrt aws : PSK "simplepsk"
# /etc/ipsec.secrets - strongSwan IPsec secrets file
```

圖 5-27 IPsec Pre-shared key 配置檔（工廠本地端）

```
root@OpenWrt:~# ip xfrm state list
src 192.168.2.1 dst 3.114.126.71
    proto esp spi 0xcc816d50 reqid 1 mode tunnel
    replay-window 0 flag af-unspec
    auth-trunc hmac(sha1) 0xbc018874442e7b765c48d1e759b5a92aeffa9de8 96
    enc cbc(aes) 0x2f94c26926489e109c56a4a0447ab305
    encapsulate type espinudp sport 4500 dport 4500 addr 0.0.0.0
    anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
src 3.114.126.71 dst 192.168.2.1
    proto esp spi 0xc1c13b87 reqid 1 mode tunnel
    replay-window 32 flag af-unspec
    auth-trunc hmac(sha1) 0xde1f6fe58abec19ee683296956601b2d4529fe97 96
    enc cbc(aes) 0x5a6c104361283aad0832e00851430a7f
    encapsulate type espinudp sport 4500 dport 4500 addr 0.0.0.0
    anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
```

圖 5-28 XFRM 框架安全狀態（IPsec 參數狀態）

```

root@OpenWrt:~# ip xfrm policy list
src 192.168.2.0/24 dst 172.16.0.0/16
    dir out priority 379519
        tmpl src 192.168.2.1 dst 3.114.126.71
            proto esp spi 0xcc816d50 reqid 1 mode tunnel
src 172.16.0.0/16 dst 192.168.2.0/24
    dir fwd priority 379519
        tmpl src 3.114.126.71 dst 192.168.2.1
            proto esp reqid 1 mode tunnel
src 172.16.0.0/16 dst 192.168.2.0/24
    dir in priority 379519
        tmpl src 3.114.126.71 dst 192.168.2.1
            proto esp reqid 1 mode tunnel
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0
src ::/0 dst ::/0
    socket in priority 0
src ::/0 dst ::/0
    socket out priority 0
src ::/0 dst ::/0
    socket in priority 0
src ::/0 dst ::/0
    socket out priority 0

```

圖 5-29 XFRM 框架安全政策 (IPsec 安全政策)

```

Connections:
aws: 192.168.2.1...3.114.126.71 IKEv2
aws: local: [wrt] uses pre-shared key authentication
aws: remote: [aws] uses pre-shared key authentication
aws: child: 192.168.2.0/24 === 172.16.0.0/16 TUNNEL
Routed Connections:
aws{1}: ROUTED, TUNNEL, reqid 1
aws{1}: 192.168.2.0/24 === 172.16.0.0/16
Security Associations (1 up, 0 connecting):
aws[20]: ESTABLISHED 17 minutes ago, 192.168.2.1[wrt]...3.114.126.71[aws]
aws[20]: IKEv2 SPIs: e45a3cfcf521e64_i* 221610ac773f5e64_r, pre-shared key reauthentication in 2 hours
aws[20]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/CURVE_25519
aws{57}: INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: ceb749d0_i ce63f1ef_o
aws{57}: AES_CBC_128/HMAC_SHA1_96, 168 bytes_i, 168 bytes_o (2 pkts, 1008s ago), rekeying in 25 minutes
aws{57}: 192.168.2.0/24 === 172.16.0.0/16
aws{58}: INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: c449dc7_i c84811bd_o
aws{58}: AES_CBC_128/HMAC_SHA1_96/CURVE_25519, 0 bytes_i, 0 bytes_o, rekeying in 42 minutes
aws{58}: 192.168.2.0/24 === 172.16.0.0/16

```

圖 5-30 IPsec 安全通道狀態

不過通常路由器具有 SNAT，這一舉動會使符合 XFRM Policy 的封包，由於 Source IP 改變，故在 xfrm lookup 如圖 5-31 中無法匹配，導致無法封裝成 ESP 封包。因此必須額外設置 iptables 如圖 5-32，若封包符合 XFRM Policy 時，跳過 NAT Table 中的 Postrouting 鏈。

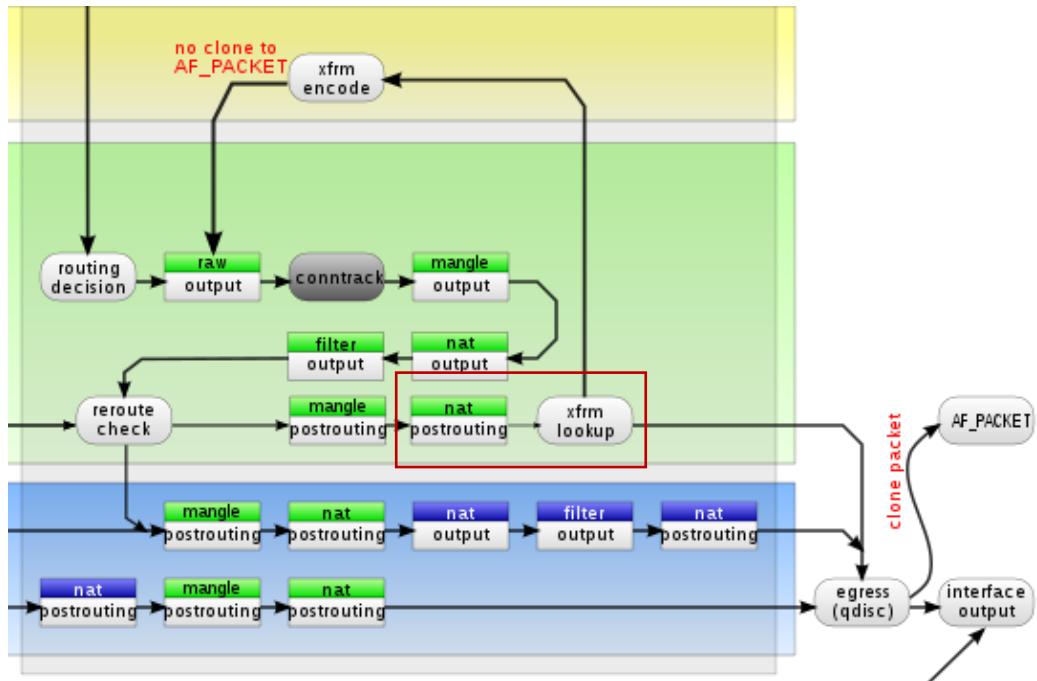


圖 5-31 Netfilter 流程[27]

```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
ACCEPT    all   --  anywhere             anywhere            policy match dir out pol ipsec
```

圖 5-32 iptables 設置以略過 Postrouting 之設置

## 4.2 雲端架構端

雲端架構端所使用的作業系統為 Ubuntu Server 20.04，所需的套件僅需 strongSwan，而設置與工廠本地端大抵相同如圖 5-33 及圖 5-34，唯一不同的便是多了 Roadwarrior 的設置，Roadwarrior 設置為圖 5-33 中 mobile 部分。

由於 iPhone 在 IPsec VPN 若使用 PSK 認證時，diffie-hellman 僅支援 Group 2 如圖 5-35，因此雲端服務端可為行動裝置端所支援的演算法一一配置。

```

conn %default
keyexchange=ikev2
authby=secret

conn mobile
ike=aes-sha1-modp1024!
esp=aes-sha1-modp2048!
leftid=aws
left=172.16.1.52           # instance private IP of local system
leftsubnet=172.16.0.0/16
leftfirewall=yes
right=%any
rightsourceip=172.15.0.0/16
auto=route

conn wrt
ike=aes-sha1-curve25519!
esp=aes-sha1-curve25519!
leftid=aws
left=172.16.1.52
leftsubnet=172.16.0.0/16
leftfirewall=yes
rightid=wrt
right=106.104.176.80
rightsubnet=192.168.1.0/24
auto=route

conn wrt-2
ike=aes-sha1-curve25519!
esp=aes-sha1-curve25519!
leftid=aws
left=172.16.1.52
leftsubnet=172.16.0.0/16
leftfirewall=yes
rightid=wrt
right=106.104.176.80
rightsubnet=192.168.2.0/24
auto=route

```

圖 5-33 IPsec 參數配置檔（雲端）

```

aws wrt : PSK "simplepsk"

# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

```

圖 5-34 IPsec Pre-shared key 配置檔（雲端）

## IPsec 設定與描述

您可以指定這些設定來定義 IPsec 的實作方式：

- 模式：通道模式。
- IKE 交換模式：預先共享密鑰與混合認證適用「嚴格」模式，或憑證認證適用「主要」模式。
- 加密演算法：3DES、AES-128 或 AES256。
- 認證演算法：HMAC-MD5 或 HMAC-SHA1。
- Diffie-Hellman 群組：預先共享密鑰和混合認證需有 Group 2、Group 2 (使用 3DES 與 AES-128) 用於憑證認證，以及 Group 2 或 5 (使用 AES-256)。

圖 5-35 iPhone IPsec VPN 支援的演算法[28]

### 4.3 行動裝置端

在 iPhone (iOS 15.3) 中設置 IPsec 十分簡單，只需進到 Settings > General 如圖 5-36 > VPN & Device Management 如圖 5-37 > VPN 如圖 5-38 > Add VPN Configuration 如圖 5-39，便能進行設定如圖 5-40。在 Description 欄位描述此 IPsec VPN 的用途，Server 為雲端伺服器的 Public IP，Server ID 則為 ipsec.conf 的 leftid，Local ID 則任意，再點開 Secret 便能填寫 PSK，連接成功便會顯示 Connected 如圖 5-41。

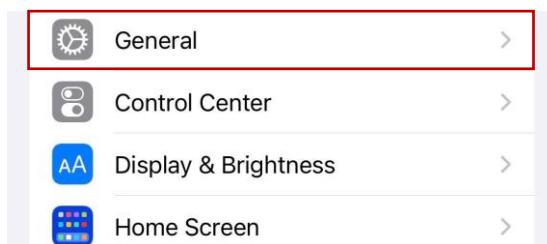


圖 5-36 iPhone 設置 IPsec VPN 流程

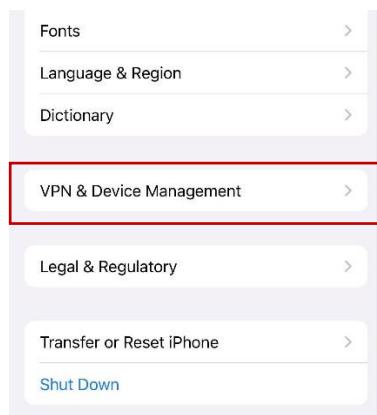


圖 5-37 iPhone 設置 IPsec VPN 流程



圖 5-38 iPhone 設置 IPsec VPN 流程



圖 5-39 iPhone 設置 IPsec VPN 流程



圖 5-40 iPhone 設置 IPsec VPN 流程

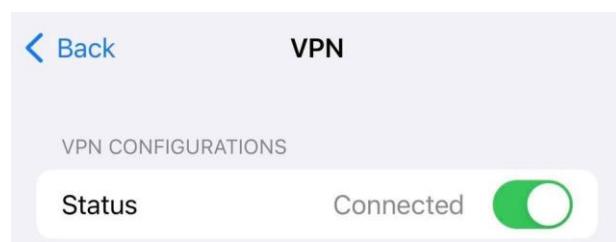


圖 5-41 iPhone IPsec VPN 連接成功

## (六) 結果與討論

### 1. 上雲架構模擬設計

在「工業物聯網架構」中，本研究成功地利用 LinkIt 7688 模擬工業控制器，以上傳資料至雲端服務中的資料存取伺服器如圖 6-1。

```
root@mylinkit:~# python3 upload.py
[{"time": "2022-02-11 06:20:22.656573"}]
b'Success 2022-02-11 06:20:22.656573'
[{"time": "2022-02-11 06:20:24.065098"}]
b'Success 2022-02-11 06:20:24.065098'
[{"time": "2022-02-11 06:20:25.169293"}]
b'Success 2022-02-11 06:20:25.169293'
[{"time": "2022-02-11 06:20:26.273191"}]
b'Success 2022-02-11 06:20:26.273191'
[{"time": "2022-02-11 06:20:27.376082"}]
b'Success 2022-02-11 06:20:27.376082'
[{"time": "2022-02-11 06:20:28.481142"}]
b'Success 2022-02-11 06:20:28.481142'
[{"time": "2022-02-11 06:20:29.584981"}]
b'Success 2022-02-11 06:20:29.584981'
[{"time": "2022-02-11 06:20:30.689573"}]
b'Success 2022-02-11 06:20:30.689573'
[{"time": "2022-02-11 06:20:31.792977"}]
b'Success 2022-02-11 06:20:31.792977'
[{"time": "2022-02-11 06:20:32.896082"}]
b'Success 2022-02-11 06:20:32.896082'
[{"time": "2022-02-11 06:20:34.001079"}]
b'Success 2022-02-11 06:20:34.001079'
[{"time": "2022-02-11 06:20:35.104877"}]
b'Success 2022-02-11 06:20:35.104877'
```

圖 6-1 LinkIt 7688 將系統時間上傳至資料存取伺服器

在「雲端服務架構」中，本研究成功地利用容器化建置資料存取伺服器如圖 6-2，接受工廠本地端子網路中的 LinkIt 7688 所上傳的資料，抑或是因應行動裝置端的要求回傳資料。而在行動裝置端開啟 IPsec VPN 時，便能夠經由安全通道連接上資料存取伺服器如圖 6-3，而在未開啟 IPsec VPN 時便無法連接上資料存取伺服器如圖 6-4。

```
web_1 | 172.15.0.1 - - [11/Feb/2022 06:49:32] "GET / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:35] "POST / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:36] "POST / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:37] "POST / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:38] "POST / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:39] "POST / HTTP/1.1" 200 -
web_1 | 172.15.0.1 - - [11/Feb/2022 06:49:39] "GET / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:40] "POST / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:41] "POST / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:42] "POST / HTTP/1.1" 200 -
web_1 | 192.168.2.100 - - [11/Feb/2022 06:49:43] "POST / HTTP/1.1" 200 -
```

圖 6-2 容器化資料存取伺服器

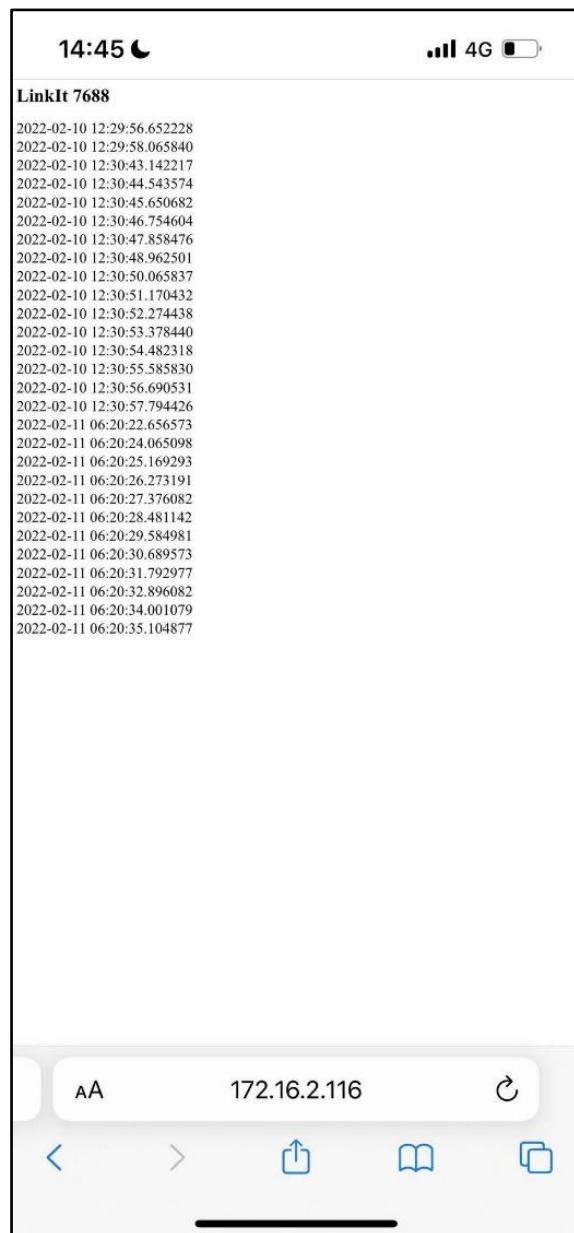


圖 6-3 成功連接資料存取伺服器



圖 6-4 未成功連接資料存取伺服器

## 2. 路由器作業系統及硬體加密加速

本研究成功地分別在 TP-Link Archer C7 AC1750 及 Banana PI R64 上安裝 OpenWRT 作業系統，並於 Banana PI R64 上開啟 ARM-CE 及 ARM-NEON 驅動如圖 6-5，以利硬體加密加速安全通道實作。此外，首先利用 OpenSSL 分析兩台不同路由器裝置的加密速度。OpenSSL 提供了 EVP 介面，開啟 EVP 介面的好處在於若是硬體支援硬體加密加速的話，OpenSSL 會自動使用硬體加密加速的實

作方式，因此在測試硬體加速必須使用 evp flag，而若不需要硬體加速時便不用 evp flag。

```
driver      : xts-aes-neonbs
driver      : ctr-aes-neonbs
driver      : cbc-aes-neonbs
driver      : ecb-aes-neonbs
driver      : __xts-aes-neonbs
driver      : ctr-aes-neonbs
driver      : __ctr-aes-neonbs
driver      : __cbc-aes-neonbs
driver      : __ecb-aes-neonbs
driver      : nhpoly1305-neon
driver      : poly1305-neon
driver      : xchacha12-neon
driver      : xchacha20-neon
driver      : chacha20-neon
driver      : sha224-arm64-neon
driver      : sha256-arm64-neon
driver      : essiv-cbc-aes-sha256-neon
driver      : cts-cbc-aes-neon
driver      : cbcmac-aes-neon
driver      : xcbc-aes-neon
driver      : cmac-aes-neon
driver      : __essiv-cbc-aes-sha256-neon
driver      : __cts-cbc-aes-neon
driver      : essiv-cbc-aes-sha256-ce
driver      : cts-cbc-aes-ce
driver      : xts-aes-ce
driver      : ctr-aes-ce
driver      : cbc-aes-ce
driver      : ecb-aes-ce
driver      : cbcmac-aes-ce
driver      : xcbc-aes-ce
driver      : cmac-aes-ce
driver      : __essiv-cbc-aes-sha256-ce
driver      : __cts-cbc-aes-ce
driver      : __xts-aes-ce
driver      : ctr-aes-ce
driver      : __ctr-aes-ce
driver      : __cbc-aes-ce
driver      : __ecb-aes-ce
driver      : ccm-aes-ce
driver      : aes-ce
driver      : gcm-aes-ce
driver      : sha256-ce
driver      : sha224-ce
driver      : sha1-ce
```

圖 6-5 ARM-CE、ARM-NEON 驅動程式

對於加密演算法 AES-128-CBC 如圖 6-6 及 AES-256-CBC 如圖 6-7 來說，有支援硬體加密加速 (ARMv8-CE) 的晶片，如 Banana PI R64 (MT7622)，開啟 evp

相比起沒有 evp 的情況下有明顯的效能提升。然而，對於無支援硬體加密加速的晶片，如 TP-Link Archer C7 AC1750 (QCA956X)，開啟 evp 相比起沒有 evp 的情況下效能十分相近。

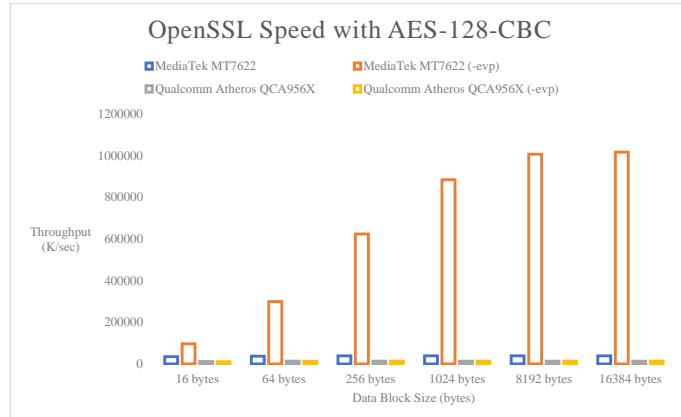


圖 6-6 AES-128-CBC OpenSSL 效能差異

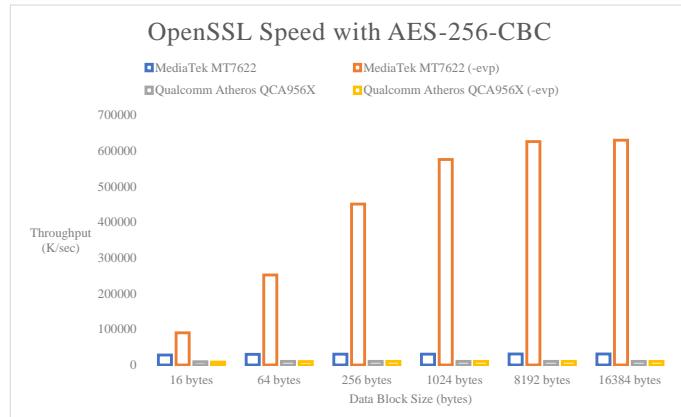


圖 6-7 AES-256-CBC OpenSSL 效能差異

### 3. 安全通道建置及分析

#### 3.1 TP-Link Archer C7 AC1750 至雲端架構安全通道

本研究成功地建立 TP-Link Archer C7 AC1750 與雲端架構端的安全通道如圖 6-8，並且透過 iPerf3[29]實測不同時間（早、中、晚）下的廣域網路、無安全通道站到站 (Site-to-Site)、安全通道站到站 (Site-to-Site) 及點到點 (End-to-End)、LinkIt 7688 乙太網路介面 30 秒內平均吞吐率。

```

Connections:
aws: 192.168.2.1...3.114.126.71 IKEv2
aws: local: [wrt] uses pre-shared key authentication
aws: remote: [aws] uses pre-shared key authentication
aws: child: 192.168.2.0/24 === 172.16.0.0/16 TUNNEL
Routed Connections:
aws[1]: ROUTED, TUNNEL, reqid 1
aws[1]: 192.168.2.0/24 === 172.16.0.0/16
Security Associations (1 up, 0 connecting):
aws[20]: ESTABLISHED 17 minutes ago, 192.168.2.1[wrt]...3.114.126.71[aws]
aws[20]: IKEv2 SPIs: e45a3cfcf521e64_i* 221610ac773f5e64_r, pre-shared key reauthentication in 2 hours
aws[20]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/CURVE_25519
aws[57]: INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: ceb749d0_i ce63fief_o
aws[57]: AES_CBC_128/HMAC_SHA1_96, 168 bytes_i, 168 bytes_o (2 pkts, 1008s ago), rekeying in 25 minutes
aws[57]: 192.168.2.0/24 === 172.16.0.0/16
aws[58]: INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: c449dc7_i c84811bd_o
aws[58]: AES_CBC_128/HMAC_SHA1_96/CURVE_25519, 0 bytes_i, 0 bytes_o, rekeying in 42 minutes
aws[58]: 192.168.2.0/24 === 172.16.0.0/16

```

圖 6-8 成功建立 TP-Link Archer C7 AC1750 與雲端架構端的安全通道

而實驗的環境設置如表 5-1 所示，實驗數據採用 iPerf3 Server 端所接受到的 intervals\_sum\_bits\_per\_second 為基準，實驗進行截圖為 Client 端。由於 Client 端與 Server 端存在著傳輸延遲，而 iPerf3 所設定的時間間距為 1 秒，故傳送的吞吐率在起初有些許落差，不過在第 1 秒過後便會趨於穩定。

表 5-1 iPerf 環境設置

	iPerf3 Server	iPerf3 Client
廣域網路(中華電信數據機不同 LAN 孔，一為靜態 IP、另一為動態 IP)	112.105.153.74	106.104.176.80
無安全通道站到站	3.114.126.71	106.104.176.80
安全通道站到站	172.16.1.52	192.168.2.1
安全通道端對端	172.16.2.116	192.168.2.100
LinkIt 7688 乙太網路至路由器	192.168.2.1	192.168.2.100

在這個實驗中觀察到廣域網路的吞吐率在 20 Mbits/s 上下如圖 6-9，而無安全通道站到站也在 20 Mbits/s 上下如圖 6-10，故確保在無安全通道的情況下，從台灣至東京可達到 ISP 所提供的頻寬。

```

root@OpenWrt:~# iperf3 -c 112.105.153.74 -t 30
Connecting to host 112.105.153.74, port 5201
[ 5] local 106.104.176.80 port 49220 connected to 112.105.153.74 port 5201
[ ID] Interval           Transfer     Bitrate      Retr  Cwnd
[ 5]  0.00-1.00   sec  2.65 MBytes  22.2 Mbits/sec  0  177 KBytes
[ 5]  1.00-2.00   sec  2.51 MBytes  21.1 Mbits/sec  0  209 KBytes
[ 5]  2.00-3.00   sec  2.33 MBytes  19.5 Mbits/sec  0  209 KBytes
[ 5]  3.00-4.00   sec  2.45 MBytes  20.5 Mbits/sec  0  209 KBytes
[ 5]  4.00-5.00   sec  2.51 MBytes  21.1 Mbits/sec  0  220 KBytes
[ 5]  5.00-6.00   sec  2.45 MBytes  20.6 Mbits/sec  0  220 KBytes
[ 5]  6.00-7.00   sec  2.45 MBytes  20.6 Mbits/sec  0  220 KBytes
[ 5]  7.00-8.00   sec  2.27 MBytes  19.0 Mbits/sec  0  220 KBytes
[ 5]  8.00-9.00   sec  2.51 MBytes  21.1 Mbits/sec  0  220 KBytes
[ 5]  9.00-10.00  sec  2.45 MBytes  20.6 Mbits/sec  0  220 KBytes
[ 5] 10.00-11.00  sec  2.51 MBytes  21.1 Mbits/sec  0  220 KBytes
[ 5] 11.00-12.00  sec  2.45 MBytes  20.6 Mbits/sec  0  220 KBytes
[ 5] 12.00-13.00  sec  2.33 MBytes  19.5 Mbits/sec  0  220 KBytes
[ 5] 13.00-14.00  sec  2.51 MBytes  21.1 Mbits/sec  0  220 KBytes
[ 5] 14.00-15.00  sec  2.45 MBytes  20.6 Mbits/sec  0  220 KBytes
[ 5] 15.00-16.00  sec  2.45 MBytes  20.5 Mbits/sec  0  220 KBytes
[ 5] 16.00-17.00  sec  2.21 MBytes  18.6 Mbits/sec  0  220 KBytes
[ 5] 17.00-18.00  sec  2.45 MBytes  20.6 Mbits/sec  0  220 KBytes
[ 5] 18.00-19.00  sec  2.51 MBytes  21.1 Mbits/sec  0  220 KBytes
[ 5] 19.00-20.00  sec  2.45 MBytes  20.6 Mbits/sec  0  220 KBytes

```

圖 6-9 廣域網路吞吐率

```

root@OpenWrt:~# iperf3 -c 3.114.126.71 -t 30
Connecting to host 3.114.126.71, port 5201
[ 5] local 106.104.176.80 port 36034 connected to 3.114.126.71 port 5201
[ ID] Interval           Transfer     Bitrate      Retr  Cwnd
[ 5]  0.00-1.00   sec  2.72 MBytes  22.9 Mbits/sec  0  285 KBytes
[ 5]  1.00-2.00   sec  2.44 MBytes  20.5 Mbits/sec  0  413 KBytes
[ 5]  2.00-3.00   sec  2.31 MBytes  19.4 Mbits/sec  0  541 KBytes
[ 5]  3.00-4.00   sec  2.62 MBytes  22.0 Mbits/sec  0  669 KBytes
[ 5]  4.00-5.00   sec  2.40 MBytes  20.1 Mbits/sec  0  792 KBytes
[ 5]  5.00-6.00   sec  2.86 MBytes  24.0 Mbits/sec  0  830 KBytes
[ 5]  6.00-7.00   sec  2.14 MBytes  17.9 Mbits/sec  0  885 KBytes
[ 5]  7.00-8.00   sec  2.70 MBytes  22.7 Mbits/sec  0  942 KBytes
[ 5]  8.00-9.00   sec  2.45 MBytes  20.5 Mbits/sec  0  942 KBytes
[ 5]  9.00-10.00  sec  2.43 MBytes  20.4 Mbits/sec  0  942 KBytes
[ 5] 10.00-11.00  sec  2.59 MBytes  21.8 Mbits/sec  0  942 KBytes
[ 5] 11.00-12.00  sec  2.49 MBytes  20.9 Mbits/sec  0  942 KBytes
[ 5] 12.00-13.00  sec  2.33 MBytes  19.6 Mbits/sec  0  1.08 MBytes
[ 5] 13.00-14.00  sec  2.61 MBytes  21.9 Mbits/sec  0  1.08 MBytes
[ 5] 14.00-15.00  sec  2.47 MBytes  20.7 Mbits/sec  0  1.08 MBytes
[ 5] 15.00-16.00  sec  2.68 MBytes  22.5 Mbits/sec  0  1.08 MBytes
[ 5] 16.00-17.00  sec  2.24 MBytes  18.8 Mbits/sec  0  1.08 MBytes
[ 5] 17.00-18.02  sec  2.87 MBytes  23.5 Mbits/sec  0  1.08 MBytes
[ 5] 18.02-19.00  sec  2.15 MBytes  18.5 Mbits/sec  0  1.12 MBytes

```

圖 6-10 無安全通道站到站吞吐率

8261...	7568..716958	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19937190 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..716979	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19938630 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..716990	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[PSH, ACK] Seq=19940070 Ack=1 Win=65344 Len=1440...
8261...	7568..717106	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19941510 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717139	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19942950 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717153	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19944390 Ack=1 Win=65344 Len=1440...
8261...	7568..717257	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19945830 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717280	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19947270 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717291	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[PSH, ACK] Seq=19948710 Ack=1 Win=65344 Len=1440...
8261...	7568..717410	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19950150 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717441	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19951590 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717453	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[PSH, ACK] Seq=19953830 Ack=1 Win=65344 Len=1440...
8261...	7568..717545	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19954470 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717568	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[ACK] Seq=19955910 Ack=1 Win=65344 Len=1440 TSva...
8261...	7568..717579	106.104.176.80	3.114.126.71	TCP	1508 36034 - 5201	[PSH, ACK] Seq=19957350 Ack=1 Win=65344 Len=1440...

圖 6-11 無安全通道下的封包截取

而安全通道站到站如圖 6-12 與無安全通道站到站的吞吐率大抵相同，因此得出加解密的吞吐率（XFRM\_lookup、XFRM\_encode）在頻寬 20 Mbits/s 的情況下，仍不會造成吞吐率下降或成為網路吞吐率瓶頸結論。

```
root@OpenWrt:~# iperf3 -c 172.16.1.52 -t 30
Connecting to host 172.16.1.52, port 5201
[ 5] local 192.168.2.1 port 53168 connected to 172.16.1.52 port 5201
[ ID] Interval Transfer Bitrate Retr Cwnd
[ 5] 0.00-1.00 sec 2.26 MBytes 19.0 Mbits/sec 0 347 KBytes
[ 5] 1.00-2.01 sec 2.49 MBytes 20.8 Mbits/sec 0 407 KBytes
[ 5] 2.01-3.02 sec 2.41 MBytes 20.0 Mbits/sec 0 407 KBytes
[ 5] 3.02-4.00 sec 2.33 MBytes 19.8 Mbits/sec 0 407 KBytes
[ 5] 4.00-5.00 sec 2.38 MBytes 20.0 Mbits/sec 0 427 KBytes
[ 5] 5.00-6.01 sec 2.41 MBytes 20.1 Mbits/sec 0 427 KBytes
[ 5] 6.01-7.04 sec 2.46 MBytes 20.0 Mbits/sec 0 427 KBytes
[ 5] 7.04-8.01 sec 2.25 MBytes 19.4 Mbits/sec 0 458 KBytes
[ 5] 8.01-9.00 sec 2.28 MBytes 19.4 Mbits/sec 0 458 KBytes
[ 5] 9.00-10.06 sec 2.66 MBytes 21.1 Mbits/sec 0 458 KBytes
[ 5] 10.06-11.01 sec 2.11 MBytes 18.6 Mbits/sec 0 458 KBytes
[ 5] 11.01-12.00 sec 2.41 MBytes 20.4 Mbits/sec 0 458 KBytes
[ 5] 12.00-13.00 sec 2.49 MBytes 20.9 Mbits/sec 0 458 KBytes
[ 5] 13.00-14.00 sec 2.35 MBytes 19.7 Mbits/sec 0 458 KBytes
[ 5] 14.00-15.00 sec 2.37 MBytes 19.9 Mbits/sec 0 458 KBytes
[ 5] 15.00-16.00 sec 2.38 MBytes 20.0 Mbits/sec 0 458 KBytes
[ 5] 16.00-17.00 sec 2.35 MBytes 19.7 Mbits/sec 0 458 KBytes
[ 5] 17.00-18.00 sec 2.35 MBytes 19.7 Mbits/sec 0 458 KBytes
[ 5] 18.00-19.11 sec 2.74 MBytes 20.8 Mbits/sec 0 458 KBytes
```

圖 6-12 安全通道站到站吞吐率

8799.. 7792.041797	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.040279	172.16.1.52	192.168.2.1	TCP	68 5201 → 53168 [ACK] Seq=1 Ack=23059542 Win=1071872 Len=0 TSval...	
8799.. 7792.042344	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.042654	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.043029	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.043344	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.043664	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.043969	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.044319	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.044623	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.044930	3.114.126.71	106.104.176.80	ESP	144 ESP (SPI=0xc2aaceb0)	
8799.. 7792.045136	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.045464	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	
8799.. 7792.044030	172.16.1.52	192.168.2.1	TCP	68 5201 → 53168 [ACK] Seq=1 Ack=23069132 Win=1071872 Len=0 TSval...	
8799.. 7792.046060	106.104.176.80	3.114.126.71	ESP	1504 ESP (SPI=0xc156eaae)	

圖 6-13 安全通道下的封包截取

本實驗限制 LinkIt 7688 乙太網路介面頻寬為 5 Mbits/s 如圖 6-14、10 Mbits/s 如圖 6-15、20 Mbits/s 如圖 6-16 情況下端對端吞吐率測試，得到 LinkIt 7688 能夠達到安全通道站到站最高吞吐率 20 Mbits/s。

```

root@mylinkit:~# iperf3 -c 172.16.2.116 -t 30 -b 5M
Connecting to host 172.16.2.116, port 5201
[ 4] local 192.168.2.124 port 40977 connected to 172.16.2.116 port 5201
[ ID] Interval      Transfer     Bandwidth    Retr  Cwnd
[ 4]  0.00-1.00  sec   653 KBytes  5.34 Mbits/sec  20   116 KBytes
[ 4]  1.00-2.00  sec   512 KBytes  4.19 Mbits/sec   0   116 KBytes
[ 4]  2.00-3.00  sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4]  3.00-4.00  sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4]  4.00-5.00  sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4]  5.00-6.00  sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4]  6.00-7.00  sec   512 KBytes  4.19 Mbits/sec   0   116 KBytes
[ 4]  7.00-8.00  sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4]  8.00-9.00  sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4]  9.00-10.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4] 10.00-11.00 sec   512 KBytes  4.19 Mbits/sec   0   116 KBytes
[ 4] 11.00-12.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4] 12.00-13.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4] 13.00-14.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4] 14.00-15.00 sec   512 KBytes  4.19 Mbits/sec   0   116 KBytes
[ 4] 15.00-16.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4] 16.00-17.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4] 17.00-18.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes
[ 4] 18.00-19.00 sec   640 KBytes  5.24 Mbits/sec   0   116 KBytes

```

圖 6-14 LinkIt 7688 乙太網路介面頻寬為 5 Mbits/s

```

root@mylinkit:~# iperf3 -c 172.16.2.116 -t 30 -b 40M
Connecting to host 172.16.2.116, port 5201
[ 4] local 192.168.2.124 port 40985 connected to 172.16.2.116 port 5201
[ ID] Interval      Transfer     Bandwidth    Retr  Cwnd
[ 4]  0.00-1.00  sec   1.99 MBytes  16.7 Mbits/sec   0   191 KBytes
[ 4]  1.00-2.00  sec   2.37 MBytes  19.9 Mbits/sec   0   273 KBytes
[ 4]  2.00-3.00  sec   2.50 MBytes  21.0 Mbits/sec   0   361 KBytes
[ 4]  3.00-4.00  sec   2.38 MBytes  19.9 Mbits/sec   0   405 KBytes
[ 4]  4.00-5.00  sec   2.38 MBytes  19.9 Mbits/sec   0   425 KBytes
[ 4]  5.00-6.00  sec   2.38 MBytes  19.9 Mbits/sec   0   452 KBytes
[ 4]  6.00-7.00  sec   2.38 MBytes  19.9 Mbits/sec   0   452 KBytes
[ 4]  7.00-8.00  sec   2.38 MBytes  19.9 Mbits/sec   0   452 KBytes
[ 4]  8.00-9.00  sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4]  9.00-10.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4] 10.00-11.00 sec   2.50 MBytes  21.0 Mbits/sec   0   458 KBytes
[ 4] 11.00-12.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4] 12.00-13.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4] 13.00-14.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4] 14.00-15.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4] 15.00-16.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4] 16.00-17.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes
[ 4] 17.00-18.00 sec   2.38 MBytes  19.9 Mbits/sec   0   458 KBytes

```

圖 6-15 LinkIt 7688 乙太網路介面頻寬為 10 Mbits/s

```

root@mylinkit:~# iperf3 -c 172.16.2.116 -t 30 -b 20M
Connecting to host 172.16.2.116, port 5201
[ 4] local 192.168.2.124 port 40983 connected to 172.16.2.116 port 5201
[ ID] Interval      Transfer     Bandwidth      Retr  Cwnd
[ 4]  0.00-1.00  sec   1.27 MBytes   10.6 Mbits/sec    0   92.3 KBytes
[ 4]  1.00-2.00  sec   2.23 MBytes   18.7 Mbits/sec    0   185 KBytes
[ 4]  2.00-3.00  sec   2.50 MBytes   21.0 Mbits/sec    0   278 KBytes
[ 4]  3.00-4.00  sec   2.38 MBytes   19.9 Mbits/sec    0   356 KBytes
[ 4]  4.00-5.00  sec   2.38 MBytes   19.9 Mbits/sec    0   404 KBytes
[ 4]  5.00-6.00  sec   2.38 MBytes   19.9 Mbits/sec    0   423 KBytes
[ 4]  6.00-7.00  sec   2.38 MBytes   19.9 Mbits/sec    0   442 KBytes
[ 4]  7.00-8.00  sec   2.38 MBytes   19.9 Mbits/sec    0   450 KBytes
[ 4]  8.00-9.00  sec   2.50 MBytes   21.0 Mbits/sec    0   456 KBytes
[ 4]  9.00-10.00 sec   2.25 MBytes   18.9 Mbits/sec    0   456 KBytes
[ 4] 10.00-11.00 sec   2.13 MBytes   17.8 Mbits/sec    0   456 KBytes
[ 4] 11.00-12.00 sec   2.50 MBytes   21.0 Mbits/sec    0   506 KBytes
[ 4] 12.00-13.00 sec   2.38 MBytes   19.9 Mbits/sec    0   570 KBytes
[ 4] 13.00-14.00 sec   2.38 MBytes   19.9 Mbits/sec    0   570 KBytes
[ 4] 14.00-15.00 sec   2.38 MBytes   19.9 Mbits/sec    0   570 KBytes
[ 4] 15.00-16.00 sec   2.38 MBytes   19.9 Mbits/sec    0   570 KBytes
[ 4] 16.00-17.00 sec   2.38 MBytes   19.9 Mbits/sec    0   570 KBytes
[ 4] 17.00-18.00 sec   2.38 MBytes   19.9 Mbits/sec    0   570 KBytes
[ 4] 18.00-19.00 sec   2.25 MBytes   18.9 Mbits/sec    0   570 KBytes

```

圖 6-16 LinkIt 7688 乙太網路介面頻寬為 20 Mbits/s

最後，本研究亦測試 LinkIt 7688 的乙太網路吞吐率如圖 6-17，實驗結果與官方網站上的理論頻寬 100 MBits/s 相去不遠，其吞吐率最低為 86 Mbits/s，最高為 90 Mbits/s。

```

root@mylinkit:~# iperf3 -c 192.168.2.1 -t 30
Connecting to host 192.168.2.1, port 5201
[ 4] local 192.168.2.100 port 45263 connected to 192.168.2.1 port 5201
[ ID] Interval      Transfer     Bandwidth      Retr  Cwnd
[ 4]  0.00-1.00  sec   11.7 MBytes   98.0 Mbits/sec   20   106 KBytes
[ 4]  1.00-2.00  sec   11.1 MBytes   93.4 Mbits/sec    0   134 KBytes
[ 4]  2.00-3.00  sec   11.3 MBytes   94.4 Mbits/sec    9   122 KBytes
[ 4]  3.00-4.00  sec   11.2 MBytes   94.1 Mbits/sec   8   107 KBytes
[ 4]  4.00-5.00  sec   11.2 MBytes   94.0 Mbits/sec    0   136 KBytes
[ 4]  5.00-6.00  sec   11.3 MBytes   94.6 Mbits/sec   5   126 KBytes
[ 4]  6.00-7.00  sec   11.2 MBytes   93.9 Mbits/sec   1   116 KBytes
[ 4]  7.00-8.00  sec   11.3 MBytes   94.7 Mbits/sec    0   143 KBytes
[ 4]  8.00-9.00  sec   11.2 MBytes   93.9 Mbits/sec   6   127 KBytes
[ 4]  9.00-10.00 sec   11.2 MBytes   93.6 Mbits/sec   8   119 KBytes
[ 4] 10.00-11.00 sec   11.2 MBytes   94.3 Mbits/sec   7   106 KBytes
[ 4] 11.00-12.00 sec   11.3 MBytes   94.7 Mbits/sec   8   94.7 KBytes
[ 4] 12.00-13.00 sec   11.2 MBytes   93.5 Mbits/sec    0   127 KBytes
[ 4] 13.00-14.00 sec   11.4 MBytes   95.0 Mbits/sec   9   110 KBytes
[ 4] 14.00-15.00 sec   11.1 MBytes   93.2 Mbits/sec    0   137 KBytes
[ 4] 15.00-16.00 sec   11.3 MBytes   94.4 Mbits/sec   7   126 KBytes
[ 4] 16.00-17.00 sec   11.2 MBytes   93.9 Mbits/sec   2   110 KBytes
[ 4] 17.00-18.00 sec   11.3 MBytes   94.4 Mbits/sec    0   137 KBytes
[ 4] 18.00-19.00 sec   11.2 MBytes   94.1 Mbits/sec   3   120 KBytes
[ 4] 19.00-20.00 sec   11.2 MBytes   94.2 Mbits/sec   7   105 KBytes

```

圖 6-17 LinkIt 7688 乙太網路吞吐率

因此，礙於實驗環境之網路頻寬限制，綜括以上實驗數據而言如圖 6-18，在頻寬超過 20 MBits/s 時，首先必須探討廣域網路是否能夠達到 ISP 的理論頻寬，再探討在此頻寬下安全通道的吞吐率是否成為瓶頸，最後在頻寬超過 90 MBits/s

時，LinkIt 7688 的乙太網路吞吐率則為瓶頸，需考慮更換控制器或網路實體介面。

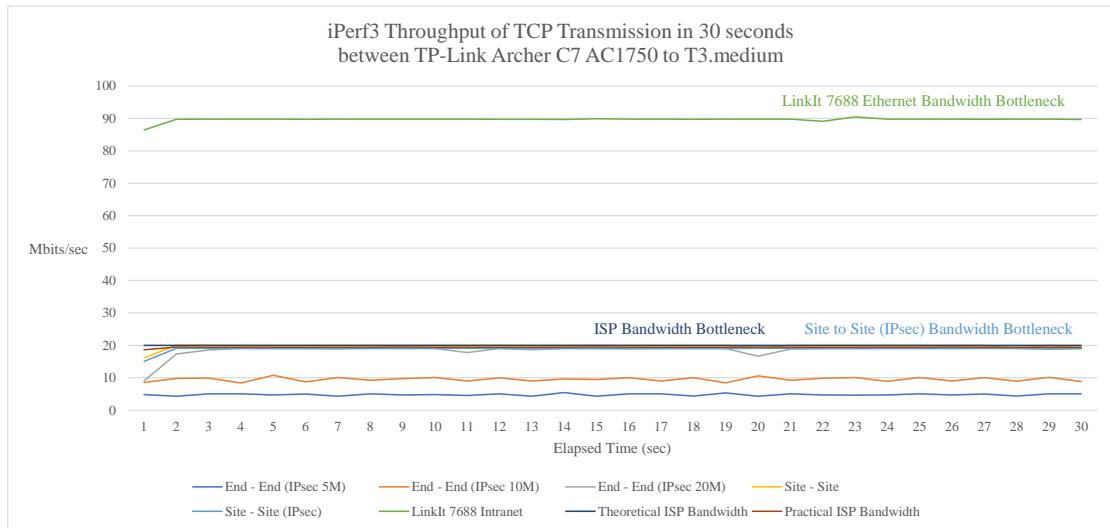


圖 6-18 各情境下吞吐率比較

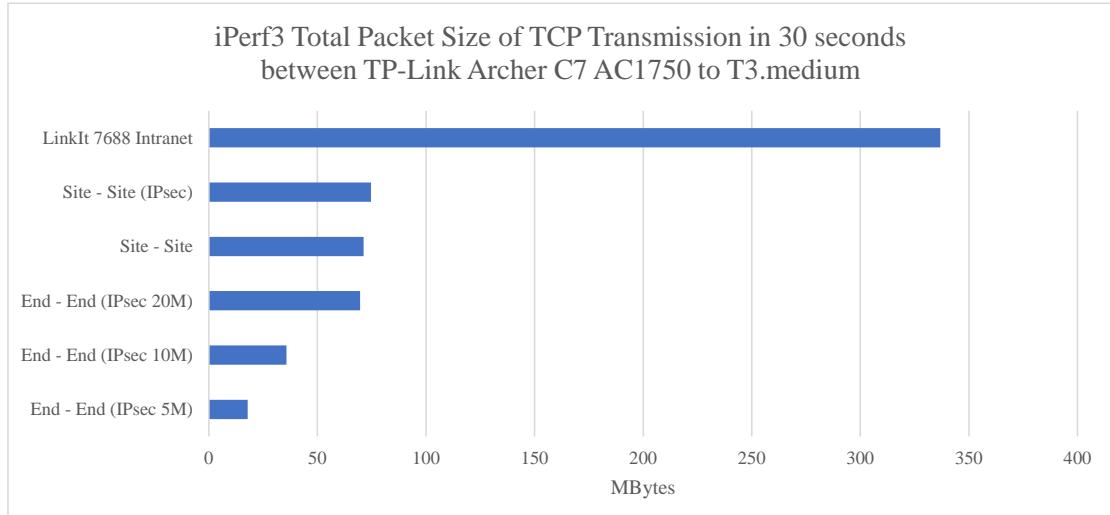


圖 6-19 總傳送的位元組

此外，本實驗數據能夠推論出加解密吞吐率高於網路傳輸吞吐率，因此安全通道加解密的過程並不會成為網路傳輸之瓶頸，以降低其吞吐率，如圖 6-20。



圖 6-20 吞吐率瓶頸示意圖

本研究亦透過 Netperf[30]實測無安全通道站到站 (Site-to-Site)、安全通道站到站 (Site-to-Site) 及點到點 (End-to-End) 延遲時間。

如表 5-2 所示，這個實驗觀察到無安全通道站到站的平均延遲時間如圖 6-21 為 40.17 毫秒，安全通道站到站的平均延遲時間如圖 6-22 為 43.30 毫秒，而點到點的平均延遲時間如圖 6-23 為 44.14 毫秒，意味著安全通道的處理時間約莫 3 毫秒（包含加解密），如圖 6-24。

表 5-2 Netperf 延遲時間數據

	最小值	平均值	最大值	標準差
無安全通道平均延遲時間	39.32 毫秒	40.17 毫秒	83.20 毫秒	27.50 毫秒
安全通道平均延遲時間	40.10 毫秒	43.30 毫秒	87.38 毫秒	32.77 毫秒
點到點平均延遲時間	40.97 毫秒	744.14 毫秒	87.61 毫秒	29.24 毫秒

```
root@OpenWrt:~# netperf -H 3.114.126.71 -t TCP_RR -- -0 min_latency,mean_latency,max_latency,stddev_latency,transaction_rate
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0) port 0 AF_INET to 3.114.126.71 (3.1) port 0 AF_INET : demo : first burst 0
Minimum      Mean      Maximum      Stddev      Transaction
Latency     Latency    Latency     Latency     Rate
Microseconds Microseconds Microseconds Microseconds Tran/s
39325        40173.64     83201       2750.83     24.783
```

圖 6-21 無安全通道站到站平均延遲時間

```
root@OpenWrt:~# netperf -H 172.16.1.52 -t TCP_RR -- -0 min_latency,mean_latency,max_latency,stddev_latency,transaction_rate
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0) port 0 AF_INET to 172.16.1.52 (172) port 0 AF_INET : demo : first burst 0
Minimum      Mean      Maximum      Stddev      Transaction
Latency     Latency    Latency     Latency     Rate
Microseconds Microseconds Microseconds Microseconds Tran/s
40103        43300.87     87380       3277.80     22.984
```

圖 6-22 安全通道站到站平均延遲時間

```
root@mylinkit:~# netperf -H 172.16.2.116 -t TCP_RR -- -0 min_latency,mean_latency,max_latency,stddev_latency,transaction_rate
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 () port 0 AF_INET to 172.16.2.116 () port 0 AF_INET : demo : first burst 0
Minimum      Mean      Maximum      Stddev      Transaction
Latency     Latency    Latency     Latency     Rate
Microseconds Microseconds Microseconds Microseconds Tran/s
40973        44144.90     87618       2924.87     22.538
```

圖 6-23 點到點平均延遲時間



圖 6-24 加解密處理時間示意圖

### 3.2 iPhone 13 (iOS 15.3) 至雲端架構安全通道

本研究亦成功地於 iPhone 13 上使用 iNetTools 進行端對端 ICMP 測試如圖 6-25，並且在雲端架構端透過 Wireshark 截取封包如圖 6-26，觀察到封包確實為 ESP，以完成傳輸安全目的。

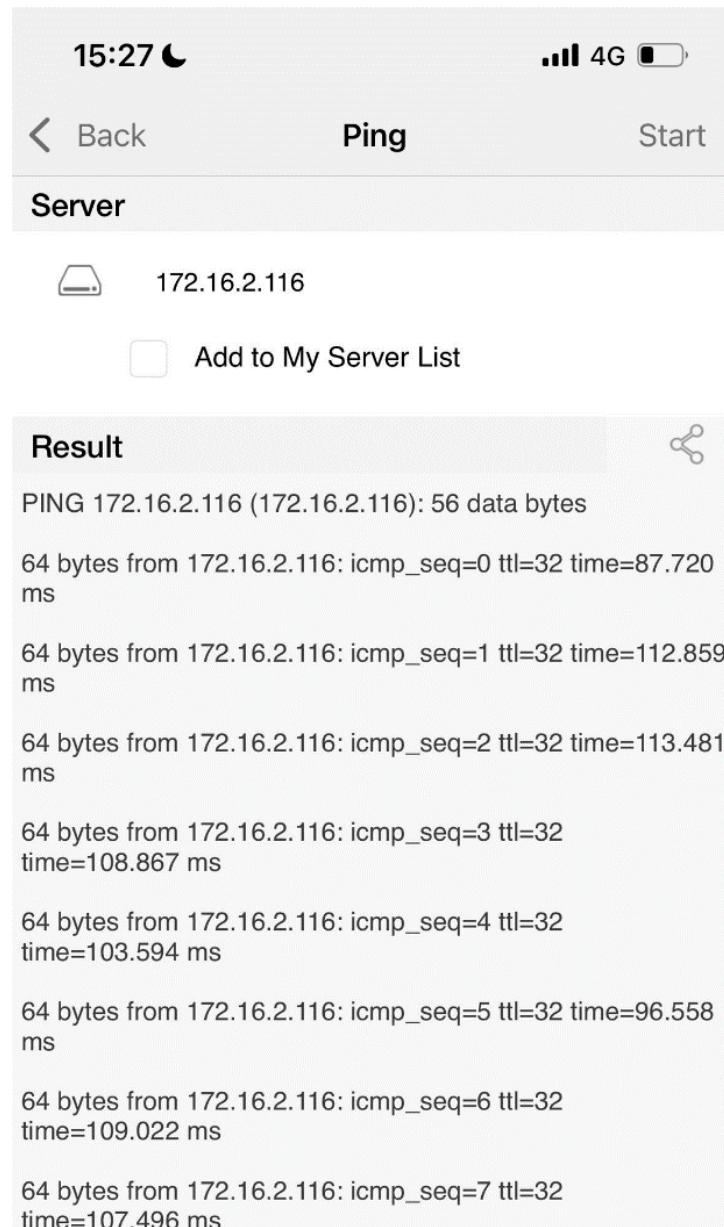


圖 6-25 iPhone 13 安全通道端對端 ICMP 測試

249 28.476529	172.16.1.52	49.217.141.249	ESP	174 ESP (SPI=0x049c46ba)
269 29.493975	49.217.141.249	172.16.1.52	ESP	174 ESP (SPI=0xcd5e87d9)
270 29.493975	172.15.0.1	172.16.2.116	ICMP	98 Echo (ping) request id=0xeb44, seq=1/256, ttl=64 (no respons...
271 29.494024	172.15.0.1	172.16.2.116	ICMP	98 Echo (ping) request id=0xeb44, seq=1/256, ttl=63 (reply in 2...
272 29.494438	172.16.2.116	172.15.0.1	ICMP	98 Echo (ping) reply id=0xeb44, seq=1/256, ttl=64 (request in...
273 29.494476	172.16.1.52	49.217.141.249	ESP	174 ESP (SPI=0x049c46ba)
293 30.499320	49.217.141.249	172.16.1.52	ESP	174 ESP (SPI=0xcd5e87d9)
294 30.499320	172.15.0.1	172.16.2.116	ICMP	98 Echo (ping) request id=0xeb44, seq=2/512, ttl=64 (no respons...
295 30.499371	172.15.0.1	172.16.2.116	ICMP	98 Echo (ping) request id=0xeb44, seq=2/512, ttl=63 (reply in 2...
296 30.499783	172.16.2.116	172.15.0.1	ICMP	98 Echo (ping) reply id=0xeb44, seq=2/512, ttl=64 (request in...

圖 6-26 Wireshark 截取封包

## 4. 執行計畫過程遇到之困難或阻礙

### 4.1 EC2 Instance 作為 NAT (已解決)

在實作將 EC2 Instance 作為 NAT 時，遇到許多問題，舉凡子網路如何路由至 NAT 上，又或是 NAT 的 iptables 要如何設定，以及最重要的便是要關閉來源／目的端檢查，這些問題皆於（五）研究方法 1.2 雲端服務架構流程中提出解決方法。

### 4.2 安全通道防火牆規則處理 (已解決)

在實工作站到站安全通道時，最主要遇到的問題便是工廠本地端子網路無法與雲端通訊，在透過 Wireshark 截取封包後發現凡是符合 XFRM 規則的封包未進行 XFRM 的處理，反而會被 SNAT，因此必須設定 iptables 中的 NAT Postrouting Chain，此問題於（五）研究方法 4.1 工廠本地端中提出解決方法

### 4.3 路由器作業系統及硬體加密加速 (未解決)

不過，對於雜湊函式 SHA1 如圖 6-27 及 SHA256 如圖 6-28 來說，有支援硬體加密加速的晶片，如 Banana PI R64，開啟 evp 反而相比起沒有 evp 的情況下有明顯的效能落後，直到 16384 bytes 時，效能才約莫追平。然而，對於無支援硬體加密加速的晶片，如 TP-Link Archer C7 AC1750，開啟 evp 相比起沒有 evp 的情況下也是有明顯的效能落後，直到 16384 bytes 時，效能約莫追平。

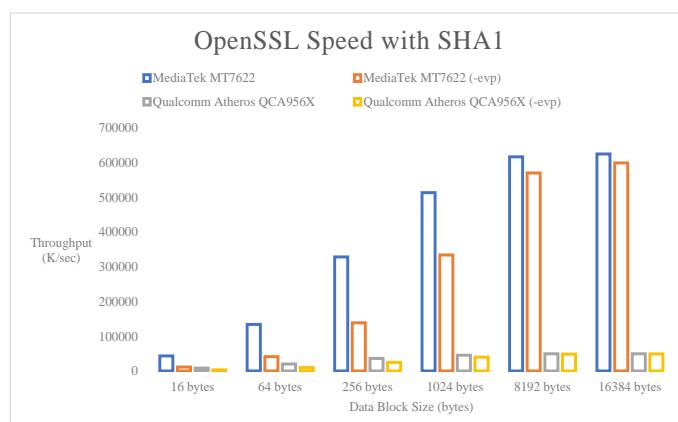


圖 6-27 SHA1 OpenSSL 效能差異

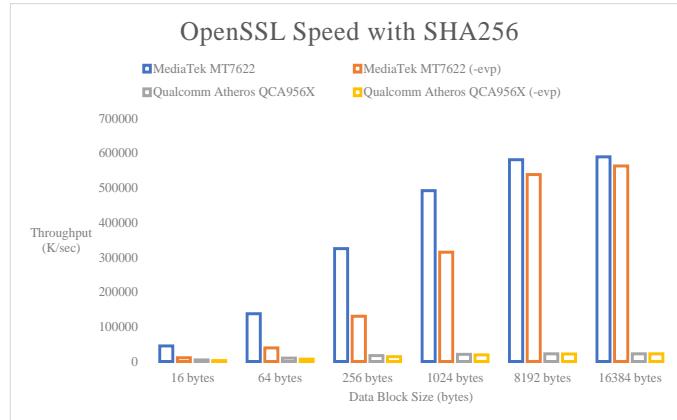


圖 6-28 SHA256 OpenSSL 效能差異

至於雜湊函式的效能差異其中的原因，我猜測雜湊函式的實作模式現今或許並不是藉由 evp 所控制的，而非如同加密演算法是藉由 evp 所控制一樣，不過本研究礙於時間上的限制，無法進行後續驗證。

#### 4.4 Banana PI R64 至雲端架構安全通道建置（未解決）

在 Banana PI R64 上，暫時無法完全建置站到站安全通道，封包僅能單向傳輸，即雲端服務子網路至工廠本地路由器端，其餘的部分皆無法正常運作。

我猜測原因出在如果採用這塊開發版的話，必須編譯並使用 OpenWRT 官方的 snapshot 版本。然而自從 2021 年 10 月開始，OpenWRT 的開發人員於 Github Issue 上提到下一個版本後的防火牆將使用 nftables 取代掉 iptables 如圖 6-29。而實驗的當下為 2022 年 1 月，使用的版本其防火牆為 nftables，因此 strongSwan 尚未支援。

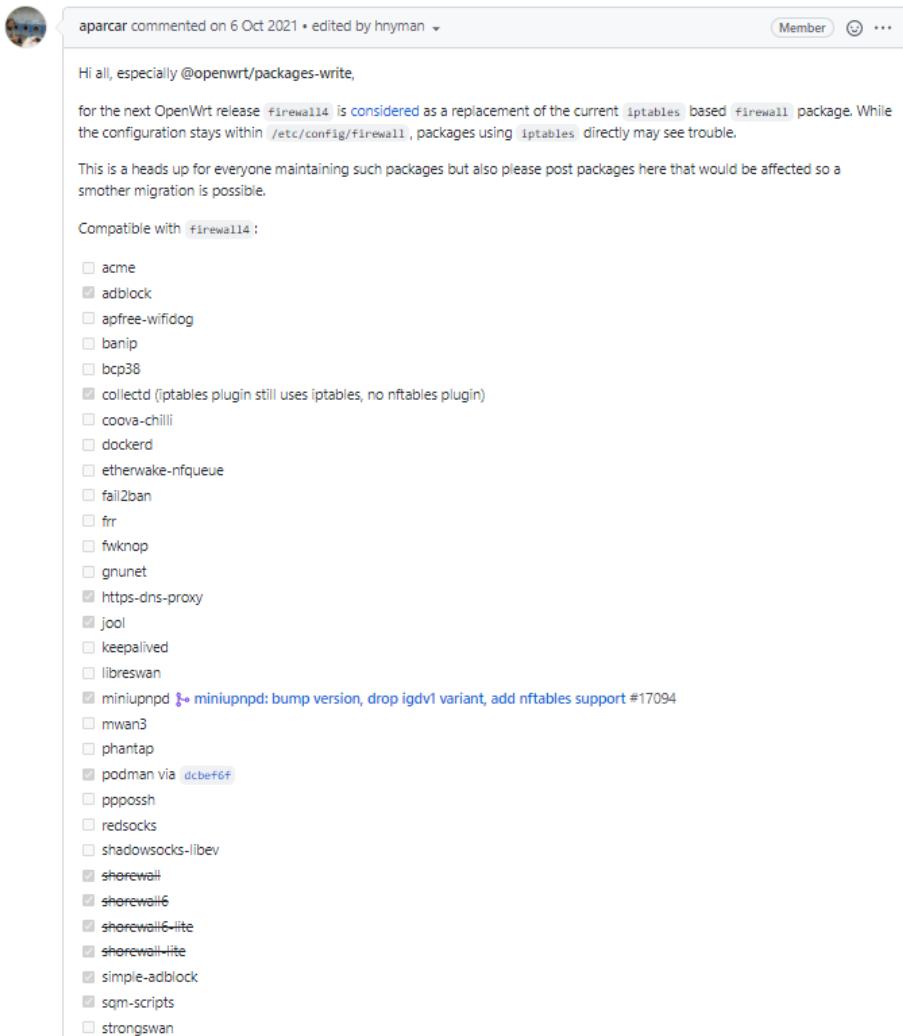


圖 6-29 OpenWRT Github Issue[31]於 2021 年 10 月提到 nftables 將在下一版本取代掉 iptables

不過，我仍有嘗試使用對應的 XFRM 框架如圖 6-30 及圖 6-31 實作 IPsec，然而封包如上所述僅能單向傳輸如圖 6-32，若從工廠本地端子網路與雲端通訊的話，以 Wireshark 截取封包分析，會發現到符合 XFRM 規則的封包如圖 6-33 都被 SNAT，導致無法進行 XFRM\_lookup 及 XFRM\_encode。

```

root@OpenWrt:~# ip xfrm state ls
src 192.168.1.1 dst 3.114.126.71
    proto esp spi 0xc2b51ea3 reqid 1 mode tunnel
    replay-window 0 flag af-unspec
    auth-trunc hmac(sha1) 0xfeef882fb9d002405b4577d4f886ff696692bcc0 96
    enc cbc(aes) 0x018ed1ee630a66938361a8bbcd82638a
    encaps type espinudp sport 4500 dport 4500 addr 0.0.0.0
    anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
src 3.114.126.71 dst 192.168.1.1
    proto esp spi 0xc864b3f1 reqid 1 mode tunnel
    replay-window 32 flag af-unspec
    auth-trunc hmac(sha1) 0x4c0b631d491c27b15d18f2d80bb79b4e6864d3a9 96
    enc cbc(aes) 0xac6a247d3e279f74d949b28577f58aaaf
    encaps type espinudp sport 4500 dport 4500 addr 0.0.0.0
    anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000

```

圖 6-30 XFRM 框架安全狀態 (IPsec 參數狀態)

```

root@OpenWrt:~# ip xfrm policy ls
src 192.168.1.0/24 dst 172.16.0.0/16
    dir out priority 379519
    tmpl src 192.168.1.1 dst 3.114.126.71
        proto esp spi 0xc2b51ea3 reqid 1 mode tunnel
src 172.16.0.0/16 dst 192.168.1.0/24
    dir fwd priority 379519
    tmpl src 3.114.126.71 dst 192.168.1.1
        proto esp reqid 1 mode tunnel
src 172.16.0.0/16 dst 192.168.1.0/24
    dir in priority 379519
    tmpl src 3.114.126.71 dst 192.168.1.1
        proto esp reqid 1 mode tunnel
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0
src ::/0 dst ::/0
    socket in priority 0
src ::/0 dst ::/0
    socket out priority 0
src ::/0 dst ::/0
    socket in priority 0
src ::/0 dst ::/0
    socket out priority 0

```

圖 6-31 XFRM 框架安全政策 (IPsec 安全政策)

526 57.163898	106.104.176.80	3.114.126.71	ESP	176 ESP (SPI=0xc2b51ea3)
532 58.010904	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=58/14848, ttl=64 (no resp...
533 58.165568	3.114.126.71	106.104.176.80	ESP	176 ESP (SPI=0xc864b3f1)
534 58.165671	172.16.1.52	192.168.1.1	ICMP	100 Echo (ping) request id=0xae6b, seq=7/1792, ttl=64 (no resp...
535 58.165791	106.104.176.80	3.114.126.71	ESP	176 ESP (SPI=0xc2b51ea3)
538 59.010169	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=59/15104, ttl=64 (no resp...
539 59.167520	3.114.126.71	106.104.176.80	ESP	176 ESP (SPI=0xc864b3f1)
540 59.167626	172.16.1.52	192.168.1.1	ICMP	100 Echo (ping) request id=0x4e6b, seq=8/2048, ttl=64 (no resp...
541 59.167743	106.104.176.80	3.114.126.71	ESP	176 ESP (SPI=0xc2b51ea3)
544 60.010336	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=60/15360, ttl=64 (no resp...
545 60.168560	3.114.126.71	106.104.176.80	ESP	176 ESP (SPI=0xc864b3f1)
546 60.168663	172.16.1.52	192.168.1.1	ICMP	100 Echo (ping) request id=0x4e6b, seq=9/2304, ttl=64 (no resp...

圖 6-32 封包僅能單向傳輸

2 1.000209	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=1/256, ttl=64 (no respons...
3 2.000386	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=2/512, ttl=64 (no respons...
4 3.000570	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=3/768, ttl=64 (no respons...
5 4.000752	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=4/1024, ttl=64 (no respons...
6 5.000918	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=5/1280, ttl=64 (no respons...
7 6.001084	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=6/1536, ttl=64 (no respons...
8 7.001264	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=7/1792, ttl=64 (no respons...
9 8.001446	106.104.176.80	172.16.1.52	ICMP	100 Echo (ping) request id=0x26df, seq=8/2048, ttl=64 (no respons...

圖 6-33 符合 XFRM 規則的封包皆被 SNAT

此問題曾在 TP-Link Archer C7 AC1750 上發生過，不過其作業系統的防火牆仍使用 iptables 實作，因此只需要將 NAT Postrouting Chain 中符合 IPsec 封包 Accept，便能夠完成站到站安全通道。

不過，在 nftables 上卻無相對應的規則，如 match ipsec、--dir out 之類，此外由於是 snapshot 版本，故穩定性較低，或許等 release 版本出來後便能解決此問題。

## 5. 結論與未來展望

由於時間上的限制，此研究實作基於雲端環境下智慧製造系統之安全通道，包含了工廠本地端、雲端架構端以及行動裝置端，並分析了轉型至雲端架構下安全通道可行性、效能及安全性。至於硬體加密加速部分礙於防火牆環境尚未穩定，此刻尚未能做出完整分析。

往後可藉由更大的網路頻寬環境測試安全通道中的加解密過程是否會成為網路傳輸吞吐量之瓶頸，及透過多次且隨機抽樣測試吞吐率、延遲等指標，以得出更嚴謹的實驗推論。抑或是探討以近期熱門的軟體定義網路（Software Defined Network）作為本地端網路架構，並於軟體定義網路中的控制層實作安全通道及服務品質（Quality of Service），以提供更高的擴展性及網路品質。

## (七) 參考文獻

- [1] A. Sun, J. Zhou, T. Ji and Q. Yue, "CSB: Cloud service bus based public SaaS platform for small and median enterprises," 2011 International Conference on Cloud and Service Computing, Hong Kong, 2011, pp. 309-314, doi: 10.1109/CSC.2011.6138539.
- [2] M. Rao, T. Newe, I. Grout, E. Lewis and A. Mathur, "FPGA Based Reconfigurable IPsec AH Core Suitable for IoT Applications", *13th IEEE International Conference on Pervasive Intelligence and computing (PICoM-2015)*.
- [3] J. Kirupakar and S. M. Shalinie, "Situation Aware Intrusion Detection System Design for Industrial IoT Gateways," 2019 International Conference on Computational Intelligence in Data Science (ICCIDIS), Chennai, India, 2019, pp. 1-6, doi: 10.1109/ICCIDIS.2019.8862038.
- [4] T. Goethals, D. Kerkhove, B. Volckaert and F. D. Turck, "Scalability evaluation of VPN technologies for secure container networking," 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 2019, pp. 1-7, doi: 10.23919/CNSM46954.2019.9012673.
- [5] S. Jahan, M. S. Rahman and S. Saha, "Application specific tunneling protocol selection for Virtual Private Networks," 2017 International Conference on Networking, Systems and Security (NSysS), Dhaka, 2017, pp. 39-44, doi: 10.1109/NSysS.2017.7885799.
- [6] S. Kent and R. Atkinson, "Security Architecture for Internet Protocol", RFC 2401, November 1998.
- [7] S. Kent and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.
- [8] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

- [9] D. Harkins and D. Carrel, "The Internet Key Exchange", RFC 2409, November 1998.
- [10] Liu, H. (2019, October 18). *An introduction to linux virtual interfaces: Tunnels*. Red Hat Developer. Retrieved February 16, 2022, from  
[https://developers.redhat.com/blog/2019/05/17/an-introduction-to-linux-virtual-interfaces-tunnels#sit\\_tunnel](https://developers.redhat.com/blog/2019/05/17/an-introduction-to-linux-virtual-interfaces-tunnels#sit_tunnel)
- [11] XFRM - The Linux Kernel documentation. Retrieved February 16, 2022, from  
[https://www.kernel.org/doc/html/latest/networking/xfrm\\_sync.html](https://www.kernel.org/doc/html/latest/networking/xfrm_sync.html)
- [12] *kernel-libipsec plugin*. Strongswan. Retrieved February 16, 2022, from  
<https://wiki.strongswan.org/projects/strongswan/wiki/Kernel-libipsec>
- [13] Hoque, Mohammad. (2015). Poster : VPN Tunnels for Energy Efficient Multimedia Streaming. 10.1145/2789168.2795168.
- [14] Rosen, R. (2014). IPsec. In *Linux kernel networking implementation and theory* (pp. 279–304). essay, Apress.
- [15] Saraiva D.A.F., Leithardt V.R.Q., de Paula D., Mendes A.S., Gonz G.V., Crocker P. PRISEC: Comparison of Symmetric Key Algorithms for IoT Devices. *Sensors*. 2019;19:4312. doi: 10.3390/s19194312.
- [16] D. Ma and Y. Shi, "A Lightweight Encryption Algorithm for Edge Networks in Software-Defined Industrial Internet of Things," 2019 IEEE 5th International Conference on Computer and Communications (ICCC), Chengdu, China, 2019, pp. 1489-1493, doi: 10.1109/ICCC47050.2019.9064352.
- [17] Google. *Introduction to cloud TPU / google cloud*. Google. Retrieved February 16, 2022, from <https://cloud.google.com/tpu/docs/intro-to-tpu>
- [18] Intel. Intel® Advanced Encryption Standard (AES) New Instructions Set White Paper. Retrieved February 16, 2022, from

- <https://www.intel.in/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf>
- [19] Arm Architecture Reference Manual for A-profile architecture. (n.d.). Retrieved February 16, 2022, from  
<https://developer.arm.com/documentation/ddi0487/latest>
- [20] *Processing Multiple Buffers in Parallel to Increase Performance on Intel® Architecture Processors*. Retrieved February 16, 2022, from  
<https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ia-32-ia-64-benchmark-code-execution-paper.pdf>
- [21] Patterson, D., & Hennessy, J. L. (2018). SISD, MIMD, SIMD, SPMD, and Vector. In *Computer Organization and Design RISC-V edition* (pp. 994–994). Elsevier.
- [22] Torvalds. (2020, December 11). *Linux/cbc.c at master · torvalds/linux*. GitHub. Retrieved February 16, 2022, from  
<https://github.com/torvalds/linux/blob/master/crypto/cbc.c>
- [23] Torvalds. *Linux/aes-glue.c at master · torvalds/linux*. GitHub. Retrieved February 16, 2022, from  
<https://github.com/torvalds/linux/blob/master/arch/arm64/crypto/aes-glue.c>
- [24] *LinkIt™ smart 7688 物聯網開發平台*. 聯發科技創意實驗室. Retrieved February 16, 2022, from <https://labs..com/zh-tw/platform/linkit-smart-7688>
- [25] *OpenWRT*. OpenWrt Wiki. (2021, September 4). Retrieved February 16, 2022, from <https://openwrt.org/>
- [26] OpenSSL Foundation, I. *OpenSSL*. /index.html. Retrieved February 16, 2022, from <https://www.openssl.org/>
- [27] *The netfilter.org project*. netfilter/iptables project homepage - The netfilter.org project. Retrieved February 16, 2022, from <https://www.netfilter.org/>

- [28] *iPhone 和 iPad 的 Cisco IPsec VPN 設定*. Apple Support. Retrieved February 16, 2022, from <https://support.apple.com/zh-tw/guide/deployment/depdf31db478/web>
- [29] GUEANT, V. *Iperf - the ultimate speed test tool for TCP, UDP and SCTP*. iPerf.fr. Retrieved February 16, 2022, from <https://iperf.fr/>
- [30] *Netperf(1) - linux man page*. netperf(1): network performance benchmark - Linux man page. (n.d.). Retrieved February 16, 2022, from <https://linux.die.net/man/1/netperf>
- [31] Openwrt. *Certain upstream switch to `Firewall4` aka `nftables` instead of `iptables` · issue #16818 · openwrt/packages*. GitHub. Retrieved February 16, 2022, from <https://github.com/openwrt/packages/issues/16818>