國 立 臺 灣 科 技 大 學

資 訊 管 理 系

碩 士 學 位 論 文

學號：M10409106

基於 MQTT 協定架構之
安全物聯網韌體更新機制

A Secure IoT Firmware Update Mechanism based on MQTT Protocol

研 究 生：許勝翔

指導教授：羅乃維　博士

中 華 民 國 一 〇 六 年 六 月

M10409106

碩士學位論文指導教授推薦書

本校　資訊管理系　　許勝翔(HSU, SHENG-HSIANG)　君

所提之論文：

基於MQTT協定架構之安全物聯網韌體更新機制

係由本人指導撰述，同意提付審查。　　指導教授：羅乃維

指導教授 羅乃維

106年 6月 30日

碩士學位考試委員審定書

M10409106

指導教授：羅乃維

本校　資訊管理系　　許勝翔　君

所提之論文：

基於MQTT協定架構之安全物聯網韌體更新機制

經本委員會審定通過，特此證明。

學校考試委員會

委　　員：

指導教授：

學程主任：

系(學程)主任、所長：

中華民國 106 年 6 月 30 日

摘要

隨著物聯網的快速發展，目前在家庭、醫院以及工廠等環境皆有可能看見被大量部署的感測裝置，藉由這些感測裝置以蒐集環境中的各種數據，我們能夠分析這些數據作後續利用。然而，這些大量部署於環境中且具有連線能力的感測裝置，已經成為有心人士的攻擊目標，雖然感測裝置製造商會定期發布新版韌體以修補漏洞，但若感測裝置之管理者疏於更新，則仍可能造成讓攻擊者造成危害。

然而，目前感測裝置之韌體更新機制未臻完善，因此我們必須確保在安裝新版本韌體前，以可靠方法驗證韌體完整性，以及韌體是否確實由其製造商所提供。否則，一旦感測裝置安裝了由攻擊者所提供的惡意韌體，可能會導致感測裝置功能停擺，甚至是使得感測裝置成為攻擊者發動後續網路攻擊的跳板。

綜合上述，本論文發展了一套基於訊息佇列遙測傳輸(Message Queuing Telemetry Transport, MQTT)協定架構之安全物聯網韌體更新機制，以確保感測裝置製造商提供的新版韌體能夠有效率地推送至目標型號的感測裝置。另外，在協定中利用橢圓曲線迪菲-赫爾曼金鑰交換(Elliptic Curve Diffie-Hellman Key Exchange, ECDH)、數位簽章(Digital Signature)及金鑰雜湊訊息鑑別碼(Keyed-hash Message Authentication Code, HMAC)等方法，以完成裝置對裝置的身分鑑別。若韌體在推送的過程中遭人竄改，或是韌體的發布者並非原感測裝置製造商，透過本論文設計的協定皆可檢驗出來，因此確保了攻擊者所提供的惡意韌體不會被安裝於感測裝置之上。最後，本論文亦對所設計的協定進行安全性分析，驗證協定可抵擋常見的竊聽攻擊、中間人攻擊、重送攻擊、偽冒攻擊等常見手法。

關鍵字：物聯網、韌體更新、身分鑑別、感測裝置、訊息佇列遙測傳輸

Abstract

With the rapid advancements in Internet of Things (IoT), there are a lot of sensors deployed in each environment such as home, hospitals, and factories. By using these sensors, we can collect different kinds of data from the environment and analyze it later. However, these sensors which have functionality to connect Internet have become attacked targets by malicious hacker. Although the manufacturers release new version of firmware to resolve vulnerability for specific sensors, the administrator of sensors may ignore the importance of firmware update. As a result, sensors are still under threat of attacks.

Nevertheless, the firmware update mechanism for sensors is not perfect nowadays. We must assure that a reliable method to verify the integrity and provider of firmware. Otherwise, if sensors install malicious firmware, they will be out of order or controlled by attackers to launch attacks in the future.

To sum up, this thesis designs a secure IoT firmware update mechanism based on Message Queuing Telemetry Transport(MQTT) protocol. It assures that new version of firmware provided by manufacturers can be pushed to corresponding sensors efficiently. We use Elliptic Curve Diffie-Hellman Key Exchange(ECDH), Digital Signature, and Keyed-hash Message Authentication Code(HMAC) algorithms in the protocol to accomplish machine-to-machine authentication. If firmware is modified or provided by attackers, our proposed protocol will detect it. Consequently, we promise that malicious firmware won't be installed on sensors. Finally, we adopt a security analysis for our protocol, and confirm that our proposed protocol can defend common attacks such as Eavesdropping Attack, Man-in-the-middle Attack, Replay Attack, and Impersonation Attack.

Keywords: Internet of Things, Firmware Update, Authentication, Sensors, Message Queuing Telemetry Transport

誌謝

在碩士班的兩年生活中，非常感謝我的指導教授羅乃維老師，老師平常除了傳授我學術上的知識，也教導我為人處事的道理。當研究或是計畫上遇到阻礙時，老師總是能給予我明確的建議與方向，讓我可以解決問題並學習相關的技術與經驗。在撰寫論文的過程中，羅老師也給予我非常多的指導，讓學生得以順利完成論文。同時，十分感謝口試委員吳宗成老師、查士朝老師對論文提出的珍貴建議，讓我的論文能夠更加完備。

再來要感謝整合數位服務實驗室的學長姐(莊祐軒、Alexander Yohan、唐偲瑋、譚學勇、林子鈞、陳伶貞)一路帶領著我持續學習、同學(劉佳琳、Doni Winata、廖建銘)是一起修課以及撰寫論文的好夥伴、學弟妹們(郭峻鴻、黎哲宇、張守群、賴柏融、李宇正、林靖倫、Suttawee Achawapong、Ryannathan Setiawan)在我撰寫論文時提供一些幫忙與協助。而平常實驗室成員的各種互動，也讓我的碩士生活額外增添了幾分光彩與回憶，感謝實驗室的大家。

最後，感謝我的父親許明男先生、母親林素珍女士、弟弟許睿勳、妹妹許婷瑄，一直支持著我當初選擇念研究所的決定，讓我能夠安心的完成碩士學業而無後顧之憂。謹以本文獻給所有關心我的老師、家人、朋友及同學，願將這份喜悅與你們一同分享。

許勝翔 謹誌於

國立台灣科技大學資訊管理研究所

中華民國 一〇六 年 七 月

目錄

圖目錄

表目錄

第一章 緒論

1.1 研究背景

近年來物聯網(Internet of Things, IoT)的應用快速地蓬勃發展，不管是在智慧家庭[1]、個人醫療[2]等情境，常可以發現大量的感測裝置被部署於空間中，透過這些感測裝置蒐集環境中的各種資料，將這些資料透過無線傳輸技術傳送至閘道器，再由閘道器轉傳至雲端伺服器，更進一步的運用。然而，物聯網確實替現代社會帶來了許多便利性，但同時也存在著風險。目前許多企業正積極投入大量資金以建置物聯網相關技術與應用，其注重的是物聯網能夠為企業帶來的利潤，但是，卻忽略了物聯網的裝置具有連線能力，就代表著這些裝置隨時都有被惡意人士鎖定而成為攻擊目標的可能。根據 2015 年美國電信公司 AT&T 曾對 5000 家企業進行調查[3]，當中高達 85%的企業，希望未來能發展物聯網的相關應用，然而，只有 12%的企業有信心以抵抗駭客攻擊，此現象說明了企業對於如何迎戰物聯網的安全威脅是沒有概念的。

另外，依據美國 Gartner 顧問機構[4]於 2016 年所發布的文章，其預測了在 2017、2018 年的前 10 大物聯網技術，在文章中提及物聯網的裝置管理問題在未來將是一大挑戰，因為物聯網的感測裝置是長期被部署在特定場域中，所以應針對裝置持續進行管理及監控，例如：進行軟體、韌體更新、裝置錯誤分析及回報等。由於物聯網本身具有可擴充性，可能有龐大數量的裝置存在於物聯網之中，隨著駭客針對物聯網裝置或是協定的漏洞進行攻擊，這些長期部署於物聯網的裝置必須要持續進行軟體或韌體更新以因應這些攻擊，因此設計出一套適用於物聯網的安全軟、韌體更新機制是必要的。

1.2 研究動機與目標

相較於車聯網中有 Tesla 公司運用所謂的空中下載(Over-the-Air, OTA)方式，針對電子控制單元(Electronic Control Unit, ECU)進行韌體更新以修補弱點[5]；目前物聯網中缺乏即時針對裝置進行安全自動韌體更新的機制。一般而言，若要對物聯網裝置進行韌體更新，目前較常見的作法是要由裝置的管理者主動至製造商提供的平台下載新版的韌體，待下載完成後將新版韌體安裝到對應的裝置上。而這種方法使得攻擊者有機可乘，因為即使裝置的製造商已針對特定型號之裝置其漏洞作修補後，而提供新版的韌體至平台上，卻由於裝置管理者不了解韌體更新的重要性，或是不曉得製造商已有新版的韌體發布，而沒有即時下載新版的韌體進行安裝，則會造成惡意人士仍有機會針對仍運行舊版韌體的裝置進行攻擊。

其次，目前許多物聯網裝置在進行更新時，並無提供韌體完整性檢查(Firmware Integrity Check)[6]，即沒有一套機制驗證即將安裝於特定型號之物聯網裝置上的韌體是否遭受惡意人士作竄改、韌體是否真正由原裝置製造商所提供。一旦裝置安裝了由攻擊者所竄改的韌體，可能造成裝置之正常功能停擺，抑或是將從環境中偵測的資料外洩給第三方，更嚴重的是這些安裝了被竄改韌體的感測裝置將可能成為攻擊者後續發動攻擊的跳板，而裝置管理者卻渾然不知。

根據趨勢科技團隊的研究[6]，一般 IoT 裝置存在有五大類安全弱點，其中包含了 IoT 裝置本身已存在潛在漏洞、裝置提供不安全的軟體及韌體更新機制等，因此容易遭受駭客的利用，所以生產 IoT 裝置之製造商必須設下最後一道防線，就是為這些裝置提供安全的軟體、韌體更新機制。Choi 等人[1]指出最近較嚴重的資訊安全攻擊其主要目標是消費者電子裝置(Consumer Electronics Devices)上的韌體，而非針對應用程式或是作業系統，因此要透過韌體驗證的方式，以偵測韌體是否已被竄改，且裝置須提供遠端更新韌體的機制，藉由韌體更新以修補漏洞。Mohan[2]指出個人醫療裝置(Personal Medical Devices, PMD)普遍缺乏鑑別機制，使得個人醫療裝置可能遭受攻擊者執行惡意韌體更新，使得病人的敏感資料外洩。

再者，由於物聯網當中所部署的感測裝置數量十分龐大，而且這些感測裝置可能分別屬於不同廠商，甚至是由同一廠商的不同型號裝置所構成，因此若由裝置管理者逐一手動進行裝置的韌體更新是非常沒有效率的做法，所以該如何達成在感測裝置製造商發布韌體時，就自動地快速將新版韌體推送至目標型號的裝置上，即是一大重要議題。因此，本論文利用 MQTT 協定以解決此議題，之所以採用 MQTT 協定，是因為 MQTT 協定屬於發布/訂閱架構，原先被設計以應用於頻寬有限以及高延遲的網路環境中。一旦韌體提供者(製造商伺服器)發布了新版本韌體，經由訊息中繼站轉傳後，若接收者(感測裝置)先前已訂閱了其製造商的韌體推送主題，則感測裝置即可接收製造商伺服器所發布的新版韌體，所以 MQTT 協定非常適合應用在由一個製造商伺服器同時對多個感測裝置進行韌體推送的情況。另外，MQTT 協定已定義了三種服務品質(Quality of Service, QoS)等級，MQTT 協定有一套機制以處理韌體在傳輸的過程中遺失的情況，以確保感測裝置能夠接收到新版韌體。

最後，綜合上述議題，本論文設計了一套針對物聯網之感測裝置進行韌體更新的安全機制。為了將感測裝置製造商所提供的韌體有效率地推送至對應型號之感測裝置上，本論文將以 MQTT 協定架構為基礎，並利用橢圓曲線迪菲-赫爾曼(Elliptic Curve Diffie-Hellman Key Exchange, ECDH)金鑰協商、數位簽章(Digital Signature)，以及金鑰雜湊訊息鑑別碼(Keyed-hash Message Authentication Code, HMAC)[7]等方法，以確保韌體由製造商傳送至感測裝置的途中，即使韌體遭受

了攻擊者的竄改，依照本論文所提出的協定將可檢驗出來。等待感測裝置接收到推送的韌體後，亦會針對韌體進行驗證，以檢驗韌體之資料完整性、韌體來源之合法性，以及感測裝置韌體適用性，若成功驗證了韌體符合前述三者特性，感測裝置才會安裝新版韌體，避免感測裝置安裝了惡意韌體而造成的資訊安全風險。

1.3 章節介紹

本論文共分為七個章節：首先，第一章描述本論文的研究背景、研究動機與目標；第二章則是針對本論文設計的協定所使用之密碼學相關理論與技術作簡短闡述；在第三章則是對於本論文所採用的 MQTT 協定架構作介紹，以期讀者能夠快速了解與本論文密切相關的 MQTT 協定，作為後續介紹本論文所設計協定的基礎；第四章則蒐集並整理與 MQTT 協定之適用性、物聯網之裝置對裝置(Machine-to-Machine, M2M)身分鑑別機制，以及物聯網韌體更新機制及情境的相關文獻；第五章將對本論文所設計的協定進行詳細說明；第六章則是針對本論文所設計協定進行安全性分析，以確保協定能夠抵抗竊聽攻擊、中間人攻擊、重送攻擊、偽冒攻擊手法，並達到金鑰安全性、前向安全性；最後，第七章則是對於整個論文作個總結與未來展望。

第二章 密碼學相關理論與技術

本章將簡短介紹本論文所設計之協定採用的密碼學相關理論與技術。如：橢圓曲線迪菲-赫爾曼金鑰交換(Elliptic Curve Diffie-Hellman Key Exchange, ECDH)，以及數位簽章(Digital Signature)。

2.1 橢圓曲線迪菲-赫爾曼金鑰交換

橢圓曲線迪菲-赫爾曼金鑰交換是一種讓通訊雙方可以在不安全通道(Insecure Channel)之中進行金鑰協商(Key Agreement)的方法，雙方將各自擁有一組由橢圓曲線密碼學而產生的公、私金鑰對，以產生共享的金鑰(Shared Secret)。ECDH 進行流程如下：

(1) 欲進行通訊之 A、B 雙方皆選擇相同之橢圓曲線參數與基點G，且G的級數 n 要夠大，在本論文中採用之橢圓曲線以滿足$y^2 = x^3 + ax + b(mod\ p)$且$4a^3+27b^2 \neq 0$方程式為主。關於橢圓曲線密碼學的完整詳細規範，可參考 Certicom 公司所提供的橢圓曲線密碼學標準文件[8]。

(2) A 方選擇一個正整數d_A，作為其私鑰，1 <d_A< n，並計算$Q_A = d_A G$，此Q_A即為 A 方之公鑰，隨後將此Q_A傳送給 B 方。

(3) B 方選擇一個正整數d_B，作為其私鑰，1 <d_B< n，並計算$Q_B = d_B G$，此Q_B即為 B 方之公鑰，隨後將此Q_B傳送給 A 方。

(4) A 方利用自身的私鑰d_A與 B 方所傳送過來的公鑰Q_B，計算得到$SK_{AB} = d_A Q_B = d_A(d_B G)$。

(5) B 方利用自身的私鑰d_B與 A 方所傳送過來的公鑰Q_A，計算得到$SK_{AB}' = d_B Q_A = d_B(d_A G)$。

(6) 又$SK_{AB} = d_A Q_B = d_A(d_B G) = d_B(d_A G) = d_B Q_A = SK_{AB}'$，故$SK_{AB}$與$SK_{AB}'$相等。

(7) 經由上述步驟，A、B 雙方可計算出相同的共享金鑰SK_{AB}

若網路中有攻擊者欲進行竊聽，雖然其可以取得公開的橢圓曲線參數、基點G、A 方公鑰Q_A、B 方公鑰Q_B，但是若要由基點G以及公鑰Q_A，要得知d_A是極其困難的；同樣地，若要由基點G以及公鑰Q_B，要得知d_B是極其困難的，即是所謂的橢圓曲線離散對數問題。因此，攻擊者並無法利用竊聽訊息的方式，而計算出 A、B 雙方所共享的金鑰SK_{AB}，於是可確保此把共享金鑰僅由 A、B 雙方知悉。

2.2 數位簽章

數位簽章(Digital Signature)其使用非對稱式密碼學(Asymmetric Cryptography)，簽署者會以自己的私鑰對訊息摘要(Message Digest)產生簽章，並提供自身的公鑰給訊息接收者，隨後訊息的接收者會利用簽署者的公鑰以驗證簽章。若驗證簽章成功，則代表已鑑別簽署者的身分合法(真實性)、原始資料在傳輸的過程中沒有被竄改(完整性)，且簽署者確實發送過此訊息(不可否認性)。

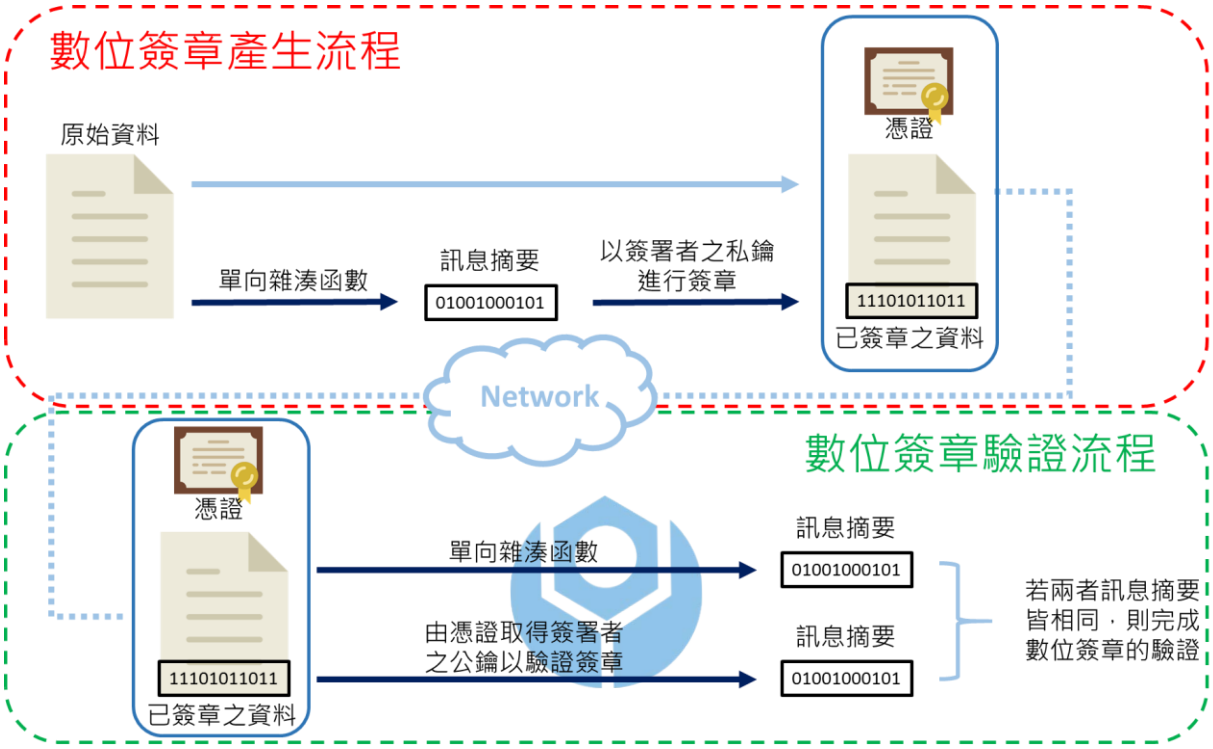

圖 1、 數位簽章產生與驗證之流程圖

- **數位簽章產生流程：**利用單向雜湊函數對原始資料產生訊息摘要後，再運用簽署者之私鑰對訊息摘要進行簽章後，得到數位簽章。最後，將原始資料、數位簽章，以及訊息簽署者之憑證，一同封裝後透過網路傳送給訊息接收者，流程如圖 1 所示。

- **數位簽章驗證流程：**訊息接收者由憑證取得訊息簽署者之公鑰。隨後利用訊息簽署者之公鑰以驗證簽章，得到一個訊息摘要。另外，訊息接收者利用單向雜湊函數對原始資料產生一個訊息摘要，並比較兩個訊息摘要值是否相等，若相等則代表資料在傳送的過程中訊息確實由該合法簽署者所發送、訊息未被竄改，且簽署者確實發送過此訊息，流程如圖 1 所示。

第三章 MQTT 協定

由於本論文所設計的協定是以訊息佇列遙測傳輸 (Message Queuing Telemetry Transport, MQTT)協定為基礎而發展的，而 MQTT 協定之細節無法盡在本論文中作敘述，故以下僅針對與本論文密切相關的 MQTT 協定之部分章節作摘要，所參考的標準文件為 MQTT Version 3.1.1 Plus Errata 01[9]之規格書。同時，亦參考由 HiveMQ 公司所提供的一系列 MQTT 相關技術文件[10]，HiveMQ 是一間提供企業級的 MQTT 訊息中繼站服務的公司。

3.1 MQTT 協定介紹

依據 MQTT Version 3.1.1 Plus Errata 01 文件，MQTT 是一個採用發布/訂閱(Publish/Subscribe)機制的訊息傳輸協定，其具有輕量化(Lightweight)、開放(Open)、精簡(Simple)、容易實作(Easy to Implement)等特性，上述特性讓 MQTT 協定適合被應用在資源有限的(Resource-Constrained)環境中，像是裝置對裝置(Machine-to-Machine, M2M)與物聯網中的通訊。MQTT 協定是在西元 1999 年由 IBM 公司的 Andy Stanford-Clark 博士以及 Arcom 公司的 Arlen Nipper 先生共同提出。在西元 2013 年由結構化資訊標準推動組織(Organization for the Advancement of Structured Information Standards, OASIS)執行標準化，在西元 2014 年 10 月正式成為 OASIS 標準。

在 MQTT 協定中有兩個角色，如客戶端(Client)與伺服器端(Server)。客戶端包含了訊息發布者(Publisher)與訊息訂閱者(Subscriber)；伺服器端則是指訊息中繼站(Broker)。訊息發布者所發布的訊息並不是直接傳送給訊息訂閱者，而是須經由訊息中繼站作訊息轉送。所有由訊息發布者傳送的訊息必須要有主題(Topic)以作為識別，而訊息中繼站會依據訊息訂閱者其訂閱的主題，將匹配主題的訊息傳送給訊息訂閱者。訊息訂閱者可向訊息中繼站註冊，以訂閱一個或多個主題。

以圖 2 為例，訊息發布者向訊息中繼站發布了主題為 temp 的訊息，若訊息訂閱者向訊息中繼站訂閱了相同主題為 temp 的訊息，則訊息中繼站便會把來自訊息發布者的訊息轉傳給所有已訂閱 temp 主題的訊息訂閱者。

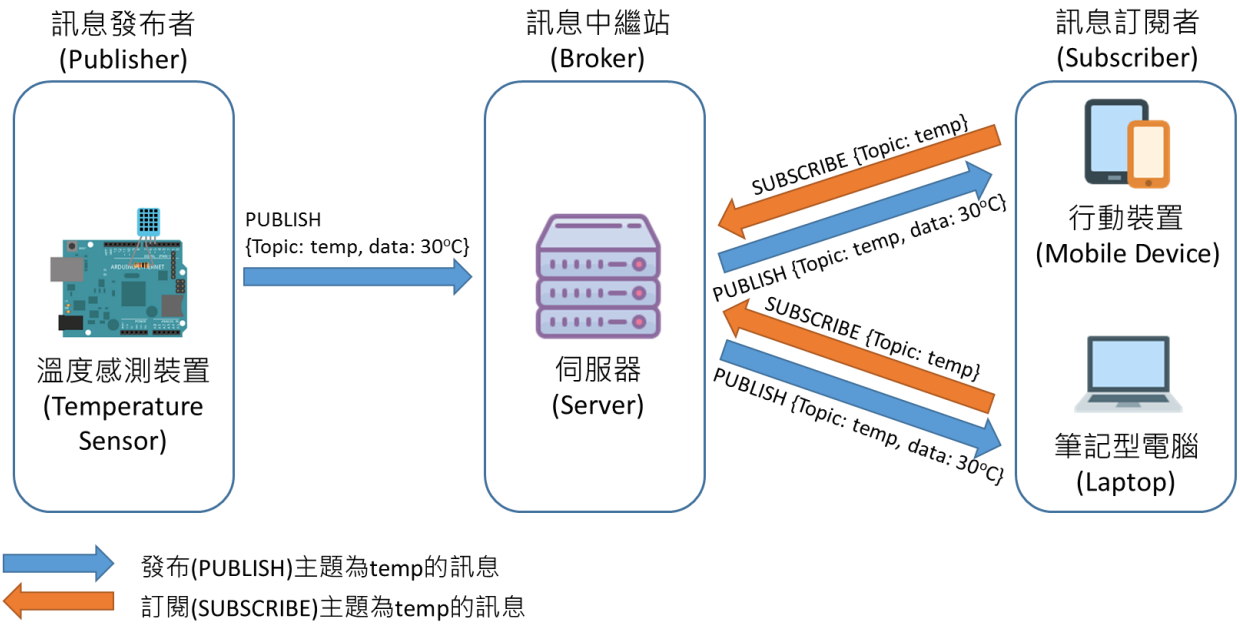

圖 2、　MQTT 協定之架構圖

3.2 MQTT 之控制封包結構

MQTT 控制封包的結構由三個部分構成，分別為：固定標頭(Fixed Header)、可變標頭(Variable Header)、酬載(Payload)，如圖 3 所示，以下分別對三者作說明：

- **固定標頭**：所有 MQTT 控制封包皆有固定標頭。
- **可變標頭**：只有部分的 MQTT 控制封包才有可變標頭。
- **酬載**：只有部分的 MQTT 控制封包才有此標頭，關於控制封包是否有酬載欄位，請參考表 1。

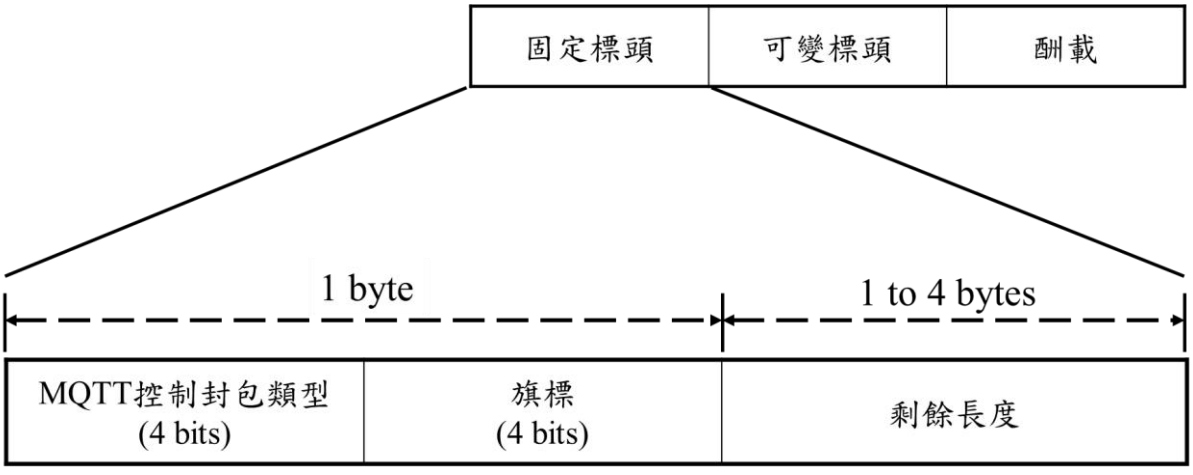

圖 3、　MQTT 之控制封包結構圖

根據圖 3，固定標頭共包含 3 個欄位，以下分別對三者作說明：

- **MQTT 控制封包類型**：MQTT 協定是藉由交換一系列已定義的 MQTT 控制封包而進行的，MQTT 控制封包類型(MQTT Control Packet Type)由 4 個位元構成，因此共有 16 種組合。其中，MQTT 協定已定義 14 種控制封包的用途，剩餘的 2 種欄位是保留欄位，如表 2 所示。
- **旗標**：每種 MQTT 控制封包皆有對應的旗標(Flags)欄位，其由 4 個位元構成，細節如表 3 所示。
- **剩餘長度**：剩餘長度(Remaining Length)此欄位最少為 1 byte，最多則是 4 bytes。此欄位用來記錄可變標頭與酬載的總共長度，其最大長度為 268,435,455 bytes(256 MB)。

表1、　MQTT 協定之控制封包的酬載狀況

控制封包	酬載(Payload)
CONNECT	Required
CONNACK	None
PUBLISH	Optional
PUBACK	None
PUBREC	None
PUBREL	None
PUBCOMP	None
SUBSCRIBE	Required
SUBACK	Required
UNSUBSCRIBE	Required
UNSUBACK	None
PINGREQ	None
PINGRESP	None
DISCONNECT	None

表2、 MQTT 之控制封包類型

控制封包	值	流動方向 (Direction of flow)	描述
Reserved	0	Forbidden	被保留的欄位
CONNECT	1	Client to Server	客戶端請求連接至伺服器端
CONNACK	2	Server to Client	連接確認
PUBLISH	3	Client to Server or Server to Client	發布訊息
PUBACK	4	Client to Server or Server to Client	發布確認
PUBREC	5	Client to Server or Server to Client	發布接收，為 QoS 2 的第一個回應
PUBREL	6	Client to Server or Server to Client	發布釋放，為 QoS 2 的第二個回應
PUBCOMP	7	Client to Server or Server to Client	發布完成，為 QoS 2 的第三個回應
SUBSCRIBE	8	Client to Server	客戶端請求訂閱
SUBACK	9	Server to Client	訂閱確認
UNSUBSCRIBE	10	Client to Server	請求取消訂閱
UNSUBACK	11	Server to Client	確認取消訂閱
PINGREQ	12	Client to Server	PING 請求
PINGRESP	13	Server to Client	PING 回應
DISCONNECT	14	Client to Server	客戶端向伺服器端進行斷線
Reserved	15	Forbidden	被保留的欄位

表3、 MQTT 控制封包之旗標對照表

控制封包	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	0	0	0	0
CONNACK	0	0	0	0
PUBLISH	DUP	QoS	QoS	RETAIN
PUBACK	0	0	0	0
PUBREC	0	0	0	0
PUBREL	0	0	1	0
PUBCOMP	0	0	0	0
SUBSCRIBE	0	0	1	0
SUBACK	0	0	0	0
UNSUBSCRIBE	0	0	1	0
UNSUBACK	0	0	0	0
PINGREQ	0	0	0	0
PINGRESP	0	0	0	0
DISCONNECT	0	0	0	0

3.3 MQTT 之服務品質等級

MQTT 協定之 PUBLISH 控制封包的固定標頭中，已定義三種服務品質(Quality of Service, QoS)等級欄位，其由 2 個位元表示，詳情如表 3 所示。MQTT 協定其下層的傳輸層協定是採用傳輸控制協定(Transmission Control Protocol, TCP)，一般來說在正常的連線情況下，TCP 協定能夠確保封包訊息確實送達目的地；但是 MQTT 協定本身屬於應用層的協定，並無法得知網路是否有壅塞或是有斷線的情形發生，所以 MQTT 協定定義了三種服務品質等級，以進行訊息的傳送。

- **QoS 0 ---最多送達一次(At most once delivery)：**

  發送者(Sender)只以 PUBLISH 控制封包發布訊息一次，訊息有可能在傳輸的過程中遺失，接收者(Receiver)最多只會收到一次訊息。

- **QoS 1 ---至少送達一次(At least once delivery)：**

  發送者將訊息以 PUBLISH 控制封包發布後會等待一段時間，確認是否有 PUBACK 控制封包自接收者回傳，若在特定時間內沒有收到來

自接收者回傳的 PUBACK 控制封包，則發送者會再次以 PUBLISH 控制封包發布訊息給接收者。因此，接收者至少會收到一次來自發送者所發布的訊息，所以接收者有可能重複收到相同的訊息。

- **QoS 2 ---確實送達一次(Exactly once delivery)：**

  發送者以 PUBLISH 控制封包發布訊息後，接收者必能接收到此次發布的訊息一次，而不會出現訊息遺失或是重複被接收的情況。然而，採用 QoS 2 將需要較多的網路負載量，完成整個 QoS 2 的流程也會需要較長的時間。

3.4 MQTT 之保留旗標

MQTT 協定之 PUBLISH 控制封包的固定標頭中，已定義保留旗標(RETAIN flag)欄位，其由 1 個位元表示，詳情如表 3 所示。保留旗標欄位可分別由訊息發布者或是訊息中繼站作設置，說明如下。

- 由訊息發布者設置：
  - 若將 RETAIN flag 設置為 0，則訊息中繼站接收此訊息後，絕對不會暫存此特定主題的訊息以及 QoS 等級。
  - 若將 RETAIN flag 設置為 1，則訊息中繼站收到此訊息後，必須暫存此特定主題的訊息以及 QoS 等級。用途在於讓已經錯過特定主題之訊息發布的新訊息訂閱者，可以即時向訊息中繼站取得暫存之最新發布的訊息，而不用等待至下次訊息發布者推送訊息。
- 若訊息發布者已設置 RETAIN flag 為 1，則訊息中繼站在發布訊息給訊息訂閱者前，會依據下列兩種不同情況，分別將 RETAIN flag 設置為 0 或 1：
  - 若特定主題的訊息經由訊息中繼站轉送前，訊息訂閱者已向訊息中繼站完成該主題的訂閱流程，此時，所有由訊息中繼站發布的 PUBLISH 控制封包之 RETAIN flag 欄位都將設置為 0。
  - 若是在兩次發布特定主題訊息之期間有動態新增的訊息訂閱者向訊息中繼站作訂閱，則訊息中繼站則會將先前其暫存的訊息發布給訊息訂閱者，並將 RETAIN flag 欄位設置為 1。當訊息訂閱者收到 PUBLISH 控制封包且 RETAIN flag 欄位設置為 1，即可得知其先前已錯過特定主題的發布訊息，此時訊息訂閱者所取得的訊息是來自於先前訊息中繼站所暫存的訊息。

3.5 MQTT 之清除會談旗標

MQTT 之 CONNECT 控制封包中，於其可變標頭中之連接旗標(Connect Flags)內，包含了清除會談旗標(Clean Session flag)，其由 1 個位元表示，以告知伺服器(Server)是否要與正在通訊的客戶端(Client)建立持續的會談(Persistent Session)。關於清除會談旗標說明如下：

- 若 Clean Session flag 設置為 0，則代表當客戶端向伺服器發送 DISCONNECT 控制封包進行斷線後，原先的會談紀錄將會保存在伺服器與客戶端上。因此，若客戶端從離線(Offline)狀態欲重新與訊息中繼站建立連線，則客戶端會向伺服器發送 CONNECT 控制封包，而伺服器會依據 CONNECT 控制封包之酬載中所包含的客戶端識別碼(Client Identifier)，找尋伺服器上是否有對應的客戶端識別碼之會談(Session)紀錄，若識別碼相符則會繼續延續此會談。

  因此，若訊息訂閱者先前已向訊息中繼站完成訂閱了某些主題(Topic)，由訊息訂閱者向訊息中繼站進行斷線後而重新連線，此時，訊息訂閱者若欲接收到特定主題的訊息，則訊息訂閱者不必向訊息中繼站以 SUBSCRIBE 控制封包重新進行訂閱，因為訊息中繼站會根據訊息訂閱者所發送的 CONNECT 控制封包之客戶端識別碼，而延續先前的會談並得知訊息訂閱者先前已訂閱哪些主題，隨後訊息中繼站將回傳 CONNACK 控制封包至訊息訂閱者，若此時 CONNACK 控制封包之可變標頭的會談表現旗標(Session Present Flag)值為 1，則訊息訂閱者即可得知其先前曾經與訊息中繼站建立會談。隨後，訊息中繼站將會推送該訊息訂閱者在離線狀態時而未接收的特定主題訊息給訊息訂閱者。

- 若 Clean Session flag 設置為 1，則代表當客戶端向伺服器發送 DISCONNECT 控制封包進行斷線後，先前的會談紀錄將會被清除，並不會保留在伺服器上。因此，若訊息訂閱者先前已向訊息中繼站完成訂閱了某些主題，隨後訊息訂閱者向訊息中繼站進行斷線後而重新連線，此時，訊息訂閱者若欲接收到特定主題的訊息，則訊息訂閱者必須重新向訊息中繼站以 SUBSCRIBE 控制封包完成訂閱流程。

第四章 文獻探討

本論文必須考量感測裝置製造商之伺服器該如何安全且有效率地將韌體推送至目標感測裝置上，因此，於 4.1 小節中整理了多位學者對於 MQTT 協定的相關文獻，以說明為何採用 MQTT 協定架構進行韌體推送。另外，由於新版本韌體是在不安全的通道中傳送，韌體隨時可能會有遭受攻擊者竄改而推送之風險，故必須要有一套機制以檢驗通訊裝置的身分合法性以及韌體完整性。故本論文於 4.2 小節中蒐集現有裝置對裝置(Machine-to-Machine, M2M)之身分鑑別機制，以作為設計協定的參考。最後於 4.3 小節中，則統整目前物聯網常見的韌體更新機制，以及韌體更新的情境作介紹。

4.1 MQTT 協定適用性

本論文採用 MQTT 協定作為韌體推送的架構，故先針對為何採用 MQTT 協定作簡短說明。Lee 等人[11]提到 MQTT 協定是由 IBM 所設計的公開協定，起初被設計的目的是為了在頻寬有限以及高延遲的網路中進行訊息傳輸。MQTT 協定為了確保訊息傳送的可靠性，其具有三種等級的服務品質(Quality of Service, QoS)，而現今 Facebook Messenger 即採用 MQTT 協定進行訊息的傳送與接收。Thota 等人[12]指出 MQTT 協定相較於限制性應用通訊協定(Constrained Application Protocol, CoAP)[13]，MQTT 協定更適合應用在由一個客戶端同時對多個節點進行訊息傳送的情況。因此，以本論文的情境而言，由於製造商伺服器必須同時對於多個感測裝置進行韌體推送，且要確保韌體在傳輸過程中的可靠性，而 MQTT 擁有完善的三種 QoS 等級，可依照實際需求作調整，所以本論文以 MQTT 協定架構為基礎，而設計出一套協定。

Poulter 等人[14]描述 MQTT 是一個輕量級的發布-訂閱協定，訊息發布者傳送的訊息必須經由訊息中繼站(Broker)作轉送，並提出 MQTT 協定在預設情況下，屬於不安全的協定。Niruntasukrat 等人[15]提出 MQTT 除了可利用使用者帳號密碼(user-password)進行身分鑑別以外，幾乎沒有其他的安全機制。因此 Niruntasukrat 等人改良原有 OAuth 1.0a 框架，於 MQTT 中進行授權(Authorization)機制。Singh 等人[16]則對於原先屬於應用層的 MQTT 協定、MQTT-SN(MQTT for Sensor Networks)協定進行安全性的改良，將兩者分別發展成為 SMQTT、SMQTT-SN，上述兩個協定皆採屬性加密系統(Attribute Based Encryption, ABE)的方式對傳輸資料加密。作者運用 MQTT 協定其原先被保留的控制封包欄位，將其定義為 Spublish 控制封包，使用此種控制封包將傳輸資料以屬性加密系統作加密。綜合前述學者的說法，皆認為 MQTT 在預設情況下並非一個安全的協定，故本論文為了防範在多重通訊實體(角色)中可能的攻擊手段，在韌體傳輸的過程

中採用橢圓曲線迪菲-赫爾曼金鑰交換、數位簽章，以及金鑰雜湊訊息鑑別碼等方式，以驗證韌體在傳輸過程的安全性。

4.2 裝置對裝置的身分鑑別機制

Ranjan 等人[17]設計一套終端身分鑑別協定(Terminal Authentication Protocol, TAP)進行裝置間的身分鑑別，其採用非對稱式密碼學的數位簽章(Digital Signature)與盲簽章(Blind Signature)方法，由於攻擊者必須同時取得兩把金鑰，才能成功取得訊息明文，因此降低了攻擊者取得訊息的風險。Lavanya 等人[18]設計一套無憑證的網路金鑰交換(Certificate-less Internet Key Exchange) 協定，其主要與傳統的網路金鑰交換協定的差異在於其利用橢圓曲線密碼學，進行金鑰協商與數位簽章。在金鑰協商時，以橢圓曲線迪菲-赫爾曼金鑰交換取代原始的迪菲-赫爾曼演算法；在數位簽章時，以橢圓曲線數位簽章演算法(Elliptic Curve Digital Signature Algorithm, ECDSA)取代原先的 RSA 演算法。使用橢圓曲線密碼學(Elliptic Curve Cryptography, ECC)的好處在於縮短金鑰長度，但仍可維持與 RSA 演算法相同等級的安全強度，且能夠降低於裝置的運算負荷量及電量消耗。Bamasag 等人[19]設計一套適用於物聯網裝置對裝置的輕量級連續性身分鑑別(Continuous Authentication)機制，裝置在固定的短時間內將大量資料傳輸給另一個裝置，並採用符記(Token)來建立祕密分享機制，針對裝置進行身分鑑別以達到快速驗證裝置身分的目的。

Butun 等人[20]提出一個可應用於公共安全裝置網路中的 CMULA(Cloud-centric, Multi-level Authentication as a Service)架構，其定義由雲端服務提供者(Cloud Service Provider, CSP)負責較複雜的運算，CSP 扮演的即是憑證管理中心(Certificate Authority, CA)的角色，其負責憑證的發放與驗證；另外，也負責產生每個穿戴裝置的公私金鑰對。在感測裝置與閘道器間運用訊息鑑別碼(Message Authentication Code, MAC)進行身分鑑別並確保資料傳輸過程的完整性，採用訊息鑑別碼可避免裝置間耗費過多的運算資源，同樣也可以完成輕量化的身分鑑別機制。Hernandez-Ramos 等人[21]於 2015 年提出可應用於智慧物件的輕量級身分鑑別與授權框架，其在身分鑑別階段採用 IEEE 802.1X 標準。IEEE 802.1X 由三個元件構成，分別為：申請者(Supplicant)、身分鑑別者(Authenticator)、身分鑑別伺服器(Authentication Server)。在申請者與身分鑑別者之間採用由 Hernandez-Ramos 等人提出的 SEAPOL(Slim Extensible Authentication Protocol over LAN) 協定以進行資料傳輸。SEAPOL 是以 EAPOL (Extensible Authentication Protocol over LAN)為基礎發展而來，利用 SEAPOL 相較於 EAPOL 可以節省智慧物件的記憶體使用量、網路傳輸量，因此能夠達成較輕量化的身分鑑別機制。Kumar 等人[22]設計了一個輕量且安全的金鑰協商辦法以應用在智慧家庭的情境下，其運用智慧裝置(Smart Device)其獨特且無法作異動的矽識別碼(Silicon ID)，以及利用對稱式

密碼學的進階加密標準(Advanced Encryption Standard, AES)演算法、訊息鑑別碼、金鑰雜湊訊息鑑別碼以對裝置進行相互的身分鑑別。

綜合上述，當感測裝置接收到新版韌體時，感測裝置該如何確保韌體的完整性(Integrity)及韌體提供者的身分真實性(Authenticity)，是本論文的探討重點。且感測裝置具有資源有限的特性，無法負荷過於複雜的運算，故本論文蒐集並整理了裝置對裝置間的身分鑑別機制，以作為本論文的參考。

4.3 物聯網韌體更新機制及情境

Chandra 等人[23]指出物聯網裝置的韌體更新目的在於改善裝置目前的功能或是修復已發現的漏洞，其提出一套應用於網狀網路(Mesh Network)的遠端韌體更新架構。Hassan 等人[24]提出基本上主要有兩種進行韌體更新的方法，第一種方法是拉動(Pull)，當製造商釋出新版本的韌體時，消費者並不會直接得知，而是要由消費者自行向伺服器詢問是否有新版本的更新可利用，比如：顯示卡的驅動程式更新；第二種方法則是推送(Push)，更新的訊息將直接傳送至訂閱者或是消費者。

同樣由 Hassan 等人[24]，將韌體更新的情境分為三種，分別為有技術性的(Skilled)、非技術性的(Non-skilled)，以及非介入的(Non-intervention)。Skilled 指的是一般使用者無法自行更新韌體的情況，比如：BMW 出產的車輛若要更新車上的韌體，必須將車輛送回給經銷商，由專業技術人員才能進行韌體更新；Non-skilled 則是指一般使用者即可進行更新，而不需要任何技術協助，如：Android 手機的版本更新；Non-intervention，則是指使用者已預定在某個時間點進行更新，到時進行更新就不再詢問使用者，如：Windows Update。Nilsson 等人[25]提出未來的趨勢是汽車製造商將提供空中下載韌體更新服務，且由於韌體關係著汽車能否正常運作與人身安全，因此韌體安全性問題十分重要。因此作者所設計的流程，電子控制單元(Electronic Control Unit, ECU)將針對韌體進行完整性、真實性驗證。

因此，本論文所設計的協定將採用前述推送的方式，將韌體主動由製造商伺服器推送至目標感測裝置上，使得目標感測裝置能夠儘速取得韌體更新的訊息以及檔案，以降低攻擊者可以針對尚未安裝新版韌體的感測裝置之漏洞而成功發動攻擊的機率。

第五章 物聯網韌體更新協定設計

本章將介紹本論文所設計之協定，以適用於物聯網中的安全韌體更新推送機制，並與 MQTT 協定架構作結合；然而，MQTT 協定在預設情況下，其缺乏安全性相關的機制，故本研究修改了部分 MQTT 協定，以確保當韌體由製造商伺服器推送至感測裝置時，感測裝置所安裝的韌體是安全無虞的。

本論文提出的協定其設計目標在於檢驗由製造商伺服器所推送的新版本韌體，在其傳輸至感測裝置的過程中，韌體是否遭受攻擊者的竄改、並驗證韌體是否確實由感測裝置之製造商所提供，以及確認感測裝置接收的韌體是否適用於該裝置型號。為了方便說明，於本章節中將製造商伺服器(Manufacturer Server)簡稱為 MF，訊息中繼站(Broker)稱作 BK，閘道器(Gateway)稱作 GW，感測裝置(Sensor)稱作 SN。

5.1 設計概念

本論文將製造商伺服器 MF 與訊息中繼站 BK 間的通訊定義為第一階層、訊息中繼站 BK 與閘道器 GW 間的通訊定義為第二階層、閘道器 GW 與感測裝置 SN 間的通訊定義為第三階層。於第一、第二階層採用 MQTT 協定架構以進行韌體推送；然而在第三階層中並不採用 MQTT 協定。韌體推送流程如圖 4 所示。

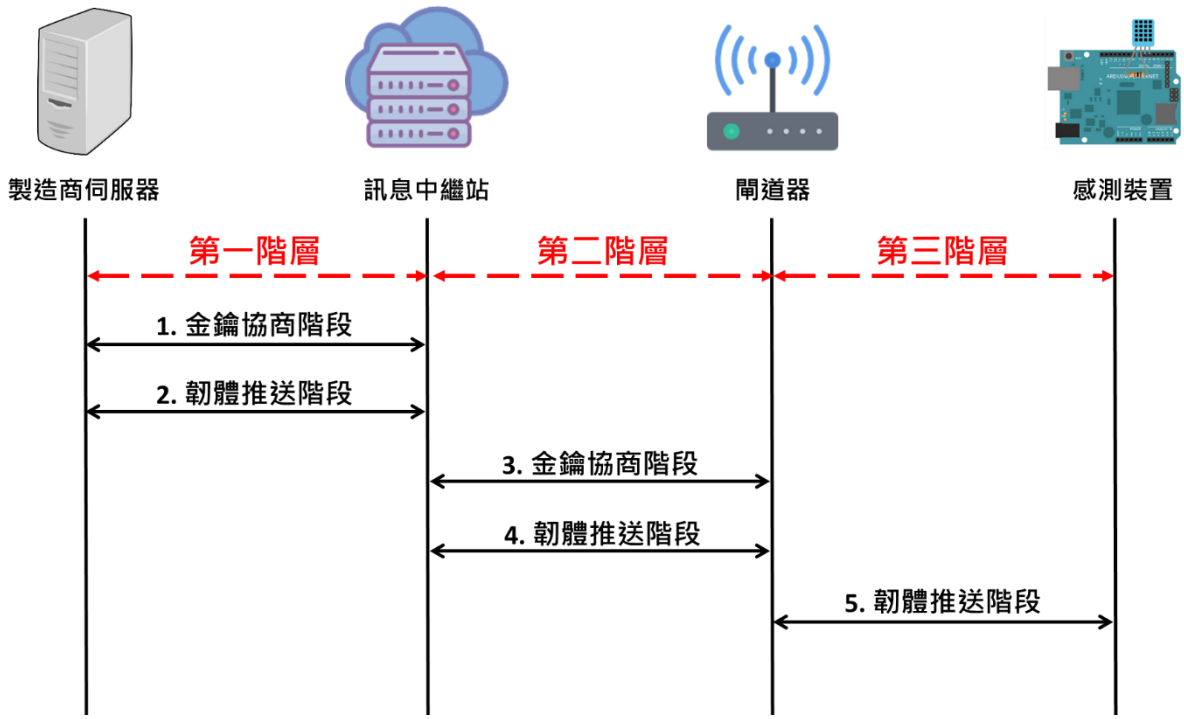

圖 4、 韌體推送流程圖

在第一階層中，製造商伺服器 MF 與訊息中繼站 BK 之間會進行金鑰協商階段、韌體推送階段；而在第二階層中，訊息中繼站 BK 與閘道器 GW 之間會進行金鑰協商階段、韌體推送階段；最後於第三階層中，閘道器 GW 與感測裝置 SN 之間僅進行韌體推送階段。

本論文採用的 MQTT 協定之 QoS 等級為 1，是基於考量到要確保製造商伺服器 MF 所發送的韌體至少會送達感測裝置一次；若採用 QoS 等級為 0，則韌體有可能在推送的過程中遺失，而未送達製造商伺服器 MF 欲進行更新之目標感測裝置上；若採用 QoS 等級為 2，則會耗費大量的網路資源以及處理時間。綜合上述，故本論文採用 QoS 等級為 1。再者，本論文設計的協定之中，由訊息發布者(製造商伺服器 MF)設置的保留旗標(RETAIN flag)，皆被設置為 1，即代表訊息中繼站 BK 必須暫存由製造商伺服器 MF 發布的韌體，使得已經錯過了特定主題之訊息發布的新訊息訂閱者，仍可即時向訊息中繼站取得先前暫存之最新發布的訊息。

另外，本論文採用的 MQTT 之 CONNECT 控制封包其可變標頭中的清除會談旗標(Clean Session flag)，皆被設置為 0，即代表在閘道器 GW 向訊息中繼站 BK 發出 DISCONNECT 控制封包進行斷線後，訊息中繼站 BK 仍會保留此次會談(Session)的紀錄。等待下次訊息訂閱者(閘道器 GW)再次與訊息中繼站 BK 建立連線時，訊息訂閱者就不必再次重新訂閱主題，以節省通訊時間以及網路流量。

5.2 協定角色

本論文所設計之協定共由 6 個角色組成，分別為製造商伺服器 MF、訊息中繼站 BK、閘道器 GW、感測裝置 SN、訊息中繼站找尋服務(Broker Discovery Service, BDS)、可信任的第三方金鑰產生中心(Trusted Third Party Key Generation Center, TTPKGC)。製造商伺服器 MF 即為 MQTT 協定之訊息發布者(Publisher)，閘道器 GW 為訊息訂閱者(Subscriber)。本論文對於 6 個角色功能之定義如下：

- **製造商伺服器 MF：**是 MQTT 協定中的訊息發布者，為韌體推送過程之起點。每當製造商對於其生產之特定型號的感測裝置提供新版本的韌體時，製造商伺服器 MF 會先呼叫訊息中繼站找尋服務，以取得所有與製造商伺服器 MF 具備合作關係之訊息中繼站 BK 其統一資源定位符(Uniform Resource Locator, URL)位址與埠號(port)清單，以建立製造商伺服器 MF 與訊息中繼站 BK 之間的連線。

- **訊息中繼站 BK：**負責轉送來自製造商伺服器 MF 的韌體推送訊息給閘道器 GW。提供訊息轉送服務的訊息中繼站 BK 可以分別屬於不同的公司，而單一公司架設之訊息中繼站 BK 可以轉送多間有合作關係之製造商的韌體更新相關訊息。因此訊息中繼站 BK 必須有充沛的儲存空間以及運算資源，以將多家製造商所推送的韌體於訊息中繼站 BK 上作暫存並轉送至閘道器 GW。

- **閘道器 GW：**是 MQTT 協定中的訊息訂閱者，閘道器 GW 可以與多個感測裝置 SN 連接，且具有依據感測裝置 SN 發生新增或是移除的情況時，而動態調整閘道器 GW 所連接的訊息中繼站 BK 清單的能力。因此，閘道器 GW 在向訊息中繼站 BK 請求建立連線前，其必須先呼叫訊息中繼站找尋服務以取得欲連線之訊息中繼站 BK 其 URL 位址與埠號清單。在閘道器 GW 與訊息中繼站 BK 建立連線之後，閘道器 GW 必須向訊息中繼站 BK 完成訂閱特定主題的流程以後，才能接收來自訊息中繼站 BK 的韌體，並把韌體轉送給對應型號的感測裝置 SN。

- **感測裝置 SN：**為韌體推送流程的終點，每個感測裝置 SN 只與單一閘道器 GW 連接。若感測裝置 SN 接收到推送的韌體後，則在感測裝置 SN 安裝此次接收的韌體之前，其須檢驗韌體完整性、韌體來源合法性，以及韌體適用性，以確保感測裝置 SN 不會安裝惡意韌體。

- **訊息中繼站找尋服務(Broker Discovery Service, BDS)**：負責提供訊息中繼站 BK 之 URL 位址與埠號清單給製造商伺服器 MF、閘道器 GW。製造商伺服器 MF 須呼叫此訊息中繼站找尋服務，使製造商伺服器 MF 取得有合作關係的訊息中繼站 BK 其 URL 位址與埠號清單，以建立製造商伺服器 MF 與訊息中繼站 BK 間的連線。步驟如圖 5 所示：

  (1) 製造商伺服器 MF 呼叫訊息中繼站找尋服務。

  (2) 訊息中繼站找尋服務回傳與製造商伺服器 MF 有合作關係的訊息中繼站 BK 其 URL 位址與埠號清單至製造商伺服器 MF。

  (3) 製造商伺服器 MF 依據(2)所提供的清單，以向訊息中繼站 BK 請求建立連線。

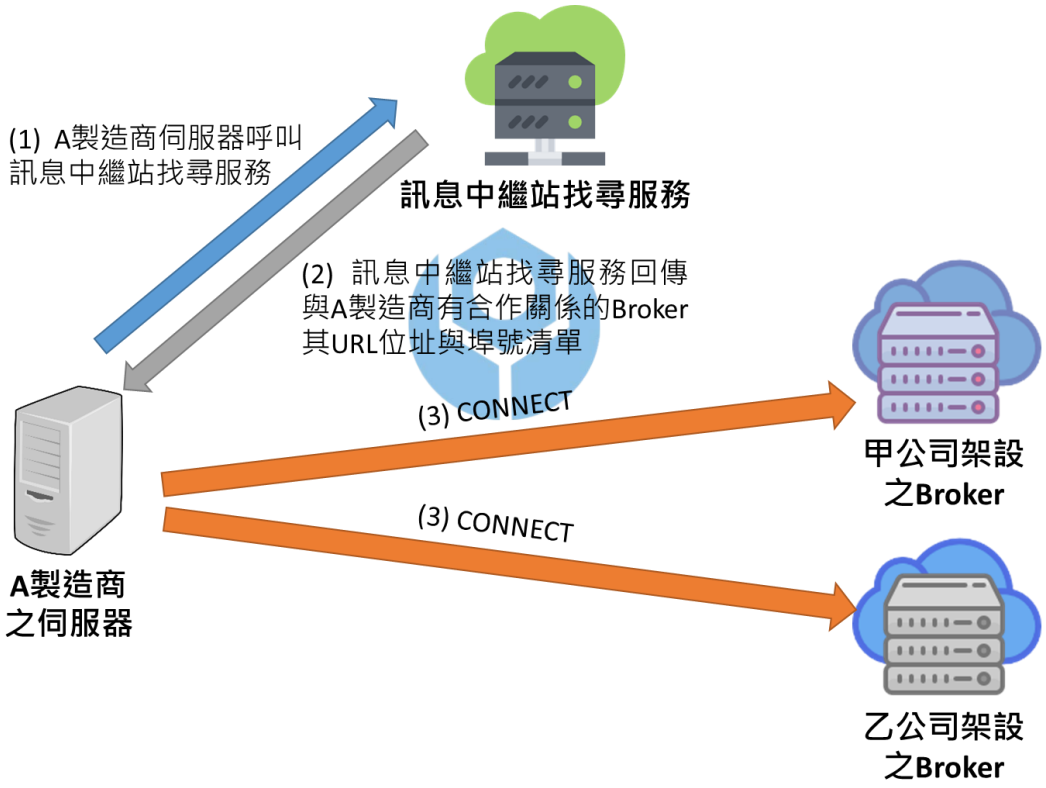

圖 5、 製造商伺服器與訊息中繼站找尋服務間的通訊流程

另一方面，閘道器 GW 亦須呼叫此訊息中繼站找尋服務，閘道器 GW 須提供其所連接之感測裝置 SN 的製造商清單給訊息中繼站找尋服務，訊息中繼站找尋服務即可計算出閘道器 GW 須向哪些訊息中繼站 BK 進行訂閱，回傳這些訊息中繼站 BK 之 URL 位址與埠號清單至閘道器 GW，以建立訊息中繼站 BK 與閘道器 GW 間的連線。步驟如圖 6 所示：

(1) 感測裝置 SN 與閘道器 GW 之間建立連線。

(2) 閘道器 GW 提供已建立連線之感測裝置 SN 其製造商清單至訊息中繼站找尋服務。

(3) 訊息中繼站找尋服務根據閘道器 GW 提交的製造商清單，自動計算出閘道器 GW 應向哪些訊息中繼站 BK 建立連線，並將對應的訊息中繼站 BK 其 URL 位址與埠號清單回傳給閘道器 GW。

(4) 閘道器 GW 根據(3)所提供的清單，以向訊息中繼站 BK 請求建立連線。

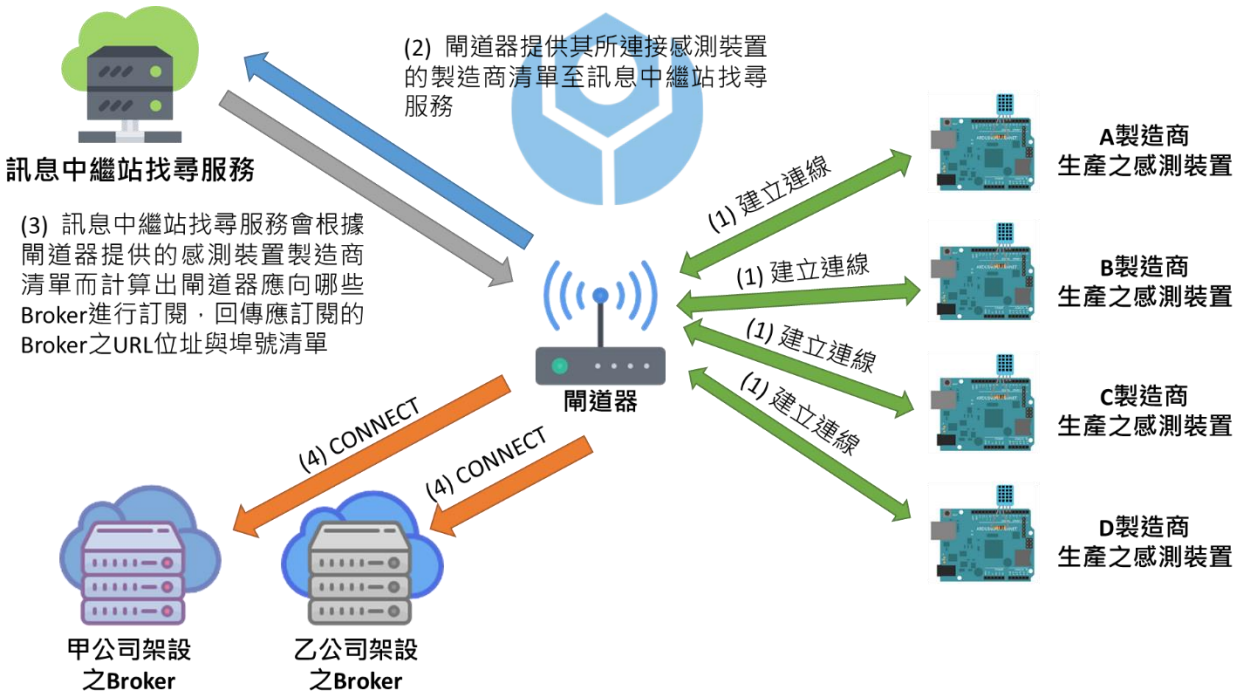

圖 6、 閘道器與訊息中繼站找尋服務間的通訊流程

- **可信任的第三方金鑰產生中心(Trusted Third Party Key Generation Center, TTPKGC)**，其負責下列四項事項：
  - 選擇一組橢圓曲線方程式相關參數與基點，並分配給製造商伺服器 MF 與訊息中繼站 BK；另外，負責選擇一組橢圓曲線方程式相關參數與基點並分配給訊息中繼站 BK 與閘道器 GW。
  - 負責產生製造商伺服器 MF、訊息中繼站 BK 以及閘道器 GW 的長期公、私金鑰對，並以安全通道將私鑰分別回傳至製造商伺服器 MF、訊息中繼站 BK 以及閘道器 GW。
  - 負責將自身產生的製造商伺服器 MF、訊息中繼站 BK 以及閘道器 GW 之長期公鑰分別作註冊以產生憑證。而且 TTPKGGC 所簽發的憑證皆會預載於製造商伺服器 MF、訊息中繼站 BK 以及閘道器 GW 的儲存空間。
  - 負責提供自身公鑰，使得此公鑰內部儲存於製造商伺服器 MF、訊息中繼站 BK 以及閘道器 GW 上，使得三者在進行通訊時，可以快速驗證接收的憑證是否由可信任的第三方金鑰產生中心核發。

5.3 前提假設

- 製造商伺服器 MF 所提供給感測裝置 SN 的韌體版本更新服務是免費的。
- 韌體本身是公開於網路上的，以提供對應型號的感測裝置進行更新，達成修補漏洞的目的，因此不需要對韌體進行任何的加密，但仍須針對韌體進行完整性驗證。
- 在感測裝置出廠前，由感測裝置製造商將祕密值、預載金鑰寫入至感測裝置的硬體安全模組(Hardware Secure Module)內，而且祕密值及預載金鑰會隨著感測裝置型號而改變。
- 製造商伺服器、訊息中繼站、閘道器的硬體安全模組內須儲存協定進行過程中所使用的長期、短期私鑰。
- 韌體推送的過程必須依序完成第一階層、第二階層以及第三階層。
- 不考量製造商伺服器 MF、訊息中繼站 BK 已被攻擊者成功控制的情況。
- 在第一、第二階層中，其服務品質等級(QoS Level)皆設定為 1。
- 所有由製造商伺服器 MF 推送至訊息中繼站 BK 的 PUBLISH 控制封包其 RETAIN flag 皆設置為 1，使韌體皆會暫存於訊息中繼站 BK 上。

- 訊息發布者、訊息訂閱者發送給訊息中繼站的清除會談旗標(Clean Session flag)皆設定為 0。
- 每個製造商伺服器 MF 所發布的韌體更新主題(Topic)是獨特且唯一的，每間製造商伺服器 MF 只會配有一個主題，且主題不會有重複的情況。
- 由不同製造商伺服器 MF 所生產的感測裝置 SN 型號，不會有重複的情況發生。例如：A 製造商有生產 MD-01 型號的裝置，則其他製造商就不會有另一款型號同為 MD-01 的裝置。
- 訊息中繼站 BK 必須具有足夠的儲存空間以暫存由製造商伺服器 MF 所發布的韌體，並具有足夠的運算能力，以將韌體轉傳至閘道器 GW。
- 閘道器 GW 必須能由其管理者分別設定兩個適當之時間參數T_R、T_I：
  - T_R：當 GW 主動向 BK 進行斷線後，每隔一固定時間會再次與 BK 進行連線。
  - T_I：當 GW 向 BK 重新建立連線後，每隔一固定時間若發現沒有任何新的訊息由 BK 傳送給 GW，則 GW 會主動向 BK 進行斷線。

5.4 修改 MQTT 協定

由於本論文在第一階層、第二階層的金鑰協商階段中，皆利用 ECDH 演算法進行金鑰協商，但是考量到單純僅採用 ECDH 作金鑰協商並無法確保通訊雙方之身分的真實性，有可能存在遭受中間人攻擊的風險，故本論文透過使用數位簽章與驗證憑證的方式，以確保進行金鑰協商的雙方其身分皆屬實，以避免中間人攻擊。然而，若要在進行 ECDH 時鑑別通訊雙方身分是否皆屬實，代表通訊雙方之間必須要以 MQTT 控制封包的酬載欄位傳送參數。

然而，根據前述 3.2 小節的表 1 可得知 MQTT 協定僅有五種控制封包可擁有酬載(Payload)，分別為：CONNECT、PUBLISH、SUBSCRIBE、SUBACK、UNSUBSCRIBE，其中 PUBLISH 控制封包的酬載是可有可無的，而另外 4 種控制封包必然具有酬載。但是，在原有的 MQTT 協定中皆已明確定義上述 5 種控制封包之用途，其餘的 9 種控制封包沒有酬載欄位，顯然 MQTT 協定已經定義的 14 種控制封包皆無法被利用以達成身分鑑別的目的。

根據 3.2 小節的表 2，可以得知 MQTT 協定有兩個控制封包的欄位是被保留的，因此本論文將利用這兩個被保留(Reserved)的欄位，將對應數值為 0 與 15 的保留欄位分別定義為 PUSH、RESPONSE 控制封包，並令兩者皆擁有酬載欄位，進而達成在進行 ECDH 金鑰協商時，能夠將參數放入酬載欄位，才能完成雙方

的身分鑑別。

5.5 符號定義

本論文設計的協定所使用符號之意義如表 4 所示。

表4、 符號定義表

符號	意義
MF, BK, GW, SN	分別為製造商伺服器 MF、訊息中繼站 BK、閘道器 GW、感測裝置 SN 的對應縮寫
$ID_{MF}, ID_{BK}, ID_{GW}, ID_{SN}$	分別為製造商伺服器 MF、訊息中繼站 BK、閘道器 GW、感測裝置 SN 的對應識別碼
MD_{SN}	經由網路傳輸的感測裝置 SN 之裝置型號，代表韌體是針對特定型號的感測裝置所發布
MD_{SN}^{L}	儲存在感測裝置 SN 記憶體的裝置型號
FW	由製造商伺服器 MF 發起推送之韌體檔案
V_{FW}	韌體推送時的版本號碼，以小數方式訂定編號
V_{FW}^{C}	正在感測裝置 SN 上執行的韌體版本號碼
S_1, S_2	將控制封包內容，經 HMAC 運算所得之值
SC	製造商在感測裝置 SN 出廠前，預先寫入至感測裝置 SN 的祕密值，會依照裝置型號而有變動
$Token$	製造商在韌體推送過程中，由製造商伺服器 MF 傳送給感測裝置 SN 的符記，用途是讓感測裝置 SN 可驗證SC
$Topic_{MF}$	製造商伺服器 MF 發布的韌體推送主題
PK_{SN}	製造商在感測裝置 SN 出廠前，預先寫入至感測裝置 SN 的金鑰，會依照裝置型號而有變動
K_{TTP}	可信任的第三方金鑰產生中心之公鑰
b_{MF}, b_{BK}, b_{GW}	MF、BK、GW 之長期私鑰
K_{MF}, K_{BK}, K_{GW}	MF、BK、GW 之長期公鑰
$Cert_{MF}, Cert_{BK}, Cert_{GW}$	MF、BK、GW 之憑證

d_{MF}, d_{BK}, d_{GW}	MF、BK、GW 經由橢圓曲線密碼學產生之短期私鑰
Q_{MF}, Q_{BK}, Q_{GW}	MF、BK、GW 經由橢圓曲線密碼學產生之短期公鑰
SK_{MF-BK}, SK_{BK-GW}	通訊雙方經由 ECDH 方法，所協商出的會談金鑰
G_{MF-BK}, G_{BK-GW}	橢圓曲線之基點
F	經由雜湊運算所得之值
$Sign_{Key}(\cdot)$	使用長期私鑰進行數位簽章的運算，$Key \in \{b_{MF}, b_{BK}, b_{GW}\}$
M_1, M_2, M_3, M_4	使用長期私鑰產生之數位簽章值
$HMAC_{SK}(\cdot)$	以會談金鑰SK進行金鑰雜湊訊息鑑別碼之運算，$SK \in \{SK_{MF-BK}, SK_{BK-GW}\}$
$H(\cdot)$	進行雜湊運算
$\oplus$	進行互斥或運算
$VerF_{Key}(\cdot)$	進行驗證憑證或是數位簽章的流程，$Key \in \{K_{TTP}, K_{MF}, K_{BK}, K_{GW}\}$

5.6 韌體更新協定

本節將針對所設計之協定進行詳細的說明，本論文設計的協定是採漸進式的將韌體由製造商伺服器 MF 推送至感測裝置 SN 上，韌體推送的順序為：製造商伺服器 MF→訊息中繼站 BK→閘道器 GW→感測裝置 SN。必須完成初始化階段以及訂閱階段後，以進行第一、第二以及第三階層間的韌體推送。

5.6.1 初始化階段

在本論文所設計協定的初始化階段，製造商、訊息中繼站找尋服務、可信任的第三方金鑰產生中心，三者會分別進行下列事項。

- 製造商：
  - 負責在感測裝置 SN 出廠前，寫入祕密值SC、預載金鑰PK_{SN}至感測裝置 SN 的硬體安全模組內。SC、PK_{SN}會隨著感測裝置 SN 型號而改變。

- 訊息中繼站找尋服務：
  - 負責讓製造商伺服器 MF、閘道器 GW 取得訊息中繼站 BK 之 URL 位址與埠號清單，以發布或訂閱特定主題的訊息。
- 可信任的第三方金鑰產生中心：
  - TTPKGC 選擇一組橢圓曲線方程式相關參數與基點，並分配給製造商伺服器 MF 與訊息中繼站 BK。另外，選擇一組橢圓曲線方程式相關參數與基點，並分配給訊息中繼站 BK 與閘道器 GW。
  - TTPKGC 產生製造商伺服器 MF、訊息中繼站 BK、閘道器 GW 的公、私金鑰對，並分別將長期私鑰b_{MF}、b_{BK}、b_{GW}透過安全通道分別回傳至製造商伺服器 MF、訊息中繼站 BK、閘道器 GW。
  - TTPKGC 將自身產生的長期公鑰K_{MF}、K_{BK}、K_{GW}作註冊以產生憑證$Cert_{MF}$、$Cert_{BK}$、$Cert_{GW}$。由 TTPKGC 所簽發的憑證$Cert_{MF}$、$Cert_{BK}$、$Cert_{GW}$皆已被預載至製造商伺服器 MF、訊息中繼站 BK 以及閘道器 GW 的儲存空間。
  - TTPKGC 的自身公鑰K_{TTP}被預載至製造商伺服器 MF、訊息中繼站 BK 以及閘道器 GW 的儲存空間。

5.6.2 訂閱階段

閘道器 GW 須向訊息中繼站 BK 成功建立連線後，隨後才能向訊息中繼站 BK 發送訂閱請求，訂閱階段之流程如圖 7 所示，流程為：

(1) 閘道器 GW 向訊息中繼站 BK 發送 CONNECT 控制封包，請求建立連線。

(2) 訊息中繼站 BK 向閘道器 GW 回傳 CONNACK 控制封包，確認建立連線。

(3) 閘道器 GW 向訊息中繼站 BK 發送 SUBSCRIBE 控制封包，以訂閱一個或多個主題。

(4) 訊息中繼站 BK 向閘道器 GW 回傳 SUBACK 控制封包，以告知閘道器 GW 其所訂閱的一個或多個主題已經建立。

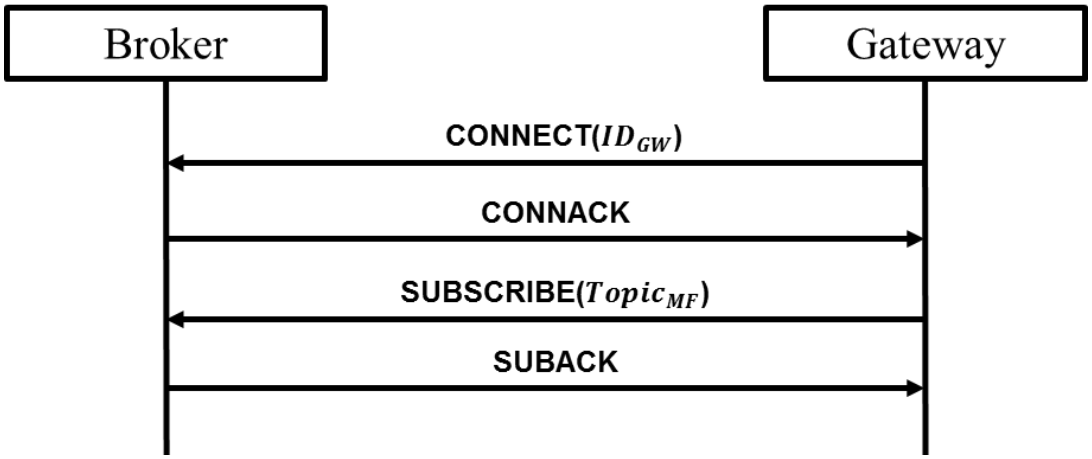

圖 7、 訂閱階段之流程圖

5.6.3 第一階層：製造商伺服器與訊息中繼站之通訊

每當製造商欲針對其生產之特定型號感測裝置 SN 發布新的韌體版本時，將韌體由製造商伺服器 MF 推送給訊息中繼站 BK，便進行第一階層的通訊。第一階層由金鑰協商階段、韌體推送階段組成。

5.6.3.1 金鑰協商階段

在製造商伺服器 MF 與訊息中繼站 BK 間，雙方採用 ECDH 以協商出相同的會談金鑰。然而，只單純採用 ECDH 金鑰協商並無法達成驗證對方身分的目的，故通訊雙方必須互相檢驗對方的數位簽章以及憑證，以驗證對方的身分。第一階層之金鑰協商流程，如圖 8 所示。

(1) 製造商伺服器 MF→訊息中繼站 BK：PUSH($Cert_{MF}$, M_1, ID_{MF}, Q_{MF})

製造商伺服器 MF 將利用 MQTT 協定之 CONNECT 控制封包向訊息中繼站 BK 請求建立連線，隨後訊息中繼站 BK 會回傳控制封包為 CONNACK 的訊息至製造商伺服器 MF，至此確立了製造商伺服器 MF 與訊息中繼站 BK 間成功建立連線。

隨後，製造商伺服器 MF 端會利用橢圓曲線密碼學以產生短期私鑰d_{MF}以及計算出短期公鑰$Q_{MF} = d_{MF}G_{MF-BK}$，再利用自身的長期私鑰b_{MF}對製造商伺服器 MF 識別碼ID_{MF}、短期公鑰Q_{MF}進行簽章，以計算出簽章$M_1 = Sign_{b_{MF}}(ID_{MF}, Q_{MF})$。隨後，製造商伺服器 MF 傳送 PUSH($Cert_{MF}$, M_1, ID_{MF}, Q_{MF})控制封包至訊息中繼站 BK。

圖 8、　第一階層之金鑰協商階段流程

(2) 訊息中繼站 BK→製造商伺服器 MF： RESPONSE($Cert_{BK}$, M_2, ID_{BK}, Q_{BK})

訊息中繼站 BK 端接收到來自製造商伺服器 MF 傳送的 PUSH($Cert_{MF}$, M_1, ID_{MF}, Q_{MF})控制封包後，將利用內部儲存的可信任第三方金鑰產生中心(TTPKGC)之公鑰K_{TTP}以對憑證$Cert_{MF}$進行驗證，確認$Cert_{MF}$是否由該 TTPKGC 所簽發。若該K_{TTP}可成功驗證$Cert_{MF}$，代表$Cert_{MF}$確實由 TTPKGC 所簽發。則訊息中繼站 BK 可以從$Cert_{MF}$取得製造商伺服器 MF 的長期公鑰K_{MF}，以驗證簽章M_1。若驗證M_1成功，則代表ID_{MF}、Q_{MF}由製造商伺服器 MF 傳送至訊息中繼站 BK 的過程中未被竄改，且確實由合法的製造商伺服器 MF 所傳送；若上述過程中發生無法成功驗證憑證$Cert_{MF}$或簽章M_1的情況，則立刻中止本協定。

承上，若訊息中繼站 BK 端成功驗證了M_1，則利用橢圓曲線密碼學以產生短期私鑰d_{BK}，並計算出短期公鑰$Q_{BK} = d_{BK}G_{MF-BK}$，至此訊息中繼站 BK 可以計算出會談金鑰$SK_{MF-BK} = d_{BK}Q_{MF} = d_{BK}(d_{MF}G_{MF-BK})$。接著訊息中繼站 BK 利用自身的長期私鑰$b_{BK}$對訊息中繼站 BK 之識別碼$ID_{BK}$以及短期公鑰$Q_{BK}$、$Q_{MF}$進行簽章，以計算出簽章$M_2 = Sign_{b_{BK}}(ID_{BK}, Q_{BK}, Q_{MF})$。隨後，訊息中繼站 BK 傳送 RESPONSE($Cert_{BK}$, M_2, ID_{BK}, Q_{BK})控制封包至製造商伺服器 MF。

在製造商伺服器 MF 接收到 RESPONSE($Cert_{BK}$, M_2, ID_{BK}, Q_{BK})控制封包之後，將利用內部儲存的K_{TTP}以對憑證$Cert_{BK}$進行驗證，確認$Cert_{BK}$是否由 TTPKGC 所簽發。若該K_{TTP}可成功驗證$Cert_{BK}$，代表$Cert_{BK}$確實由 TTPKGC 所簽發。接著製造商伺服器 MF 可由$Cert_{BK}$取得訊息中繼站 BK 的長期公鑰K_{BK}，以驗證簽章M_2，若驗證M_2成功，則代表ID_{BK}、Q_{BK}由訊息中繼站 BK 傳送至製造商伺服器 MF 的過程中未被竄改，且確實由合法的訊息中繼站 BK 所傳送；若上述過程中發生無法成功驗證憑證$Cert_{BK}$或簽章M_2的情況，則立刻中止本協定。

承上，若製造商伺服器 MF 端成功驗證了M_2，則計算出會談金鑰$SK_{MF-BK} = d_{MF}Q_{BK} = d_{MF}(d_{BK}G_{MF-BK})$。至此完成了以 ECDH 在製造商伺服器 MF 與訊息中繼站 BK 間協商出一把相同的會談金鑰SK_{MF-BK}，並且同時完成了製造商伺服器 MF 與訊息中繼站 BK 彼此的身分鑑別，因此確保此會談金鑰SK_{MF-BK}只會由合法的製造商伺服器 MF 與訊息中繼站 BK 共同知悉，而第三方無法得知SK_{MF-BK}。

5.6.3.2 韌體推送階段

進行韌體推送階段時，為了檢驗推送的韌體是否確實由原感測裝置之製造商伺服器 MF 所提供，而運用在初始化階段所協商出來的會談金鑰以進行 HMAC 運算，以進行韌體的完整性以及韌體來源合法性的驗證，流程如圖 9 所示。

(1) 製造商伺服器 MF→訊息中繼站 BK：PUBLISH(S_1, $Topic_{MF}$, ID_{MF}, FW, MD_{SN}, V_{FW}, $Token$)

首先，在製造商伺服器 MF 端會計算出一個 Token 值，$Token = SC \oplus HMAC_{PK_{SN}}(ID_{MF}, FW, V_{FW})$，隨後使用金鑰協商階段所計算的會談金鑰 SK_{MF-BK} 以計算出 $S_1 = HMAC_{SK_{MF-BK}}(Topic_{MF}, ID_{MF}, FW, MD_{SN}, V_{FW}, Token)$。隨後，製造商伺服器 MF 將 PUBLISH($S_1$, $Topic_{MF}$, ID_{MF}, FW, MD_{SN}, V_{FW}, $Token$)控制封包傳送至訊息中繼站 BK。

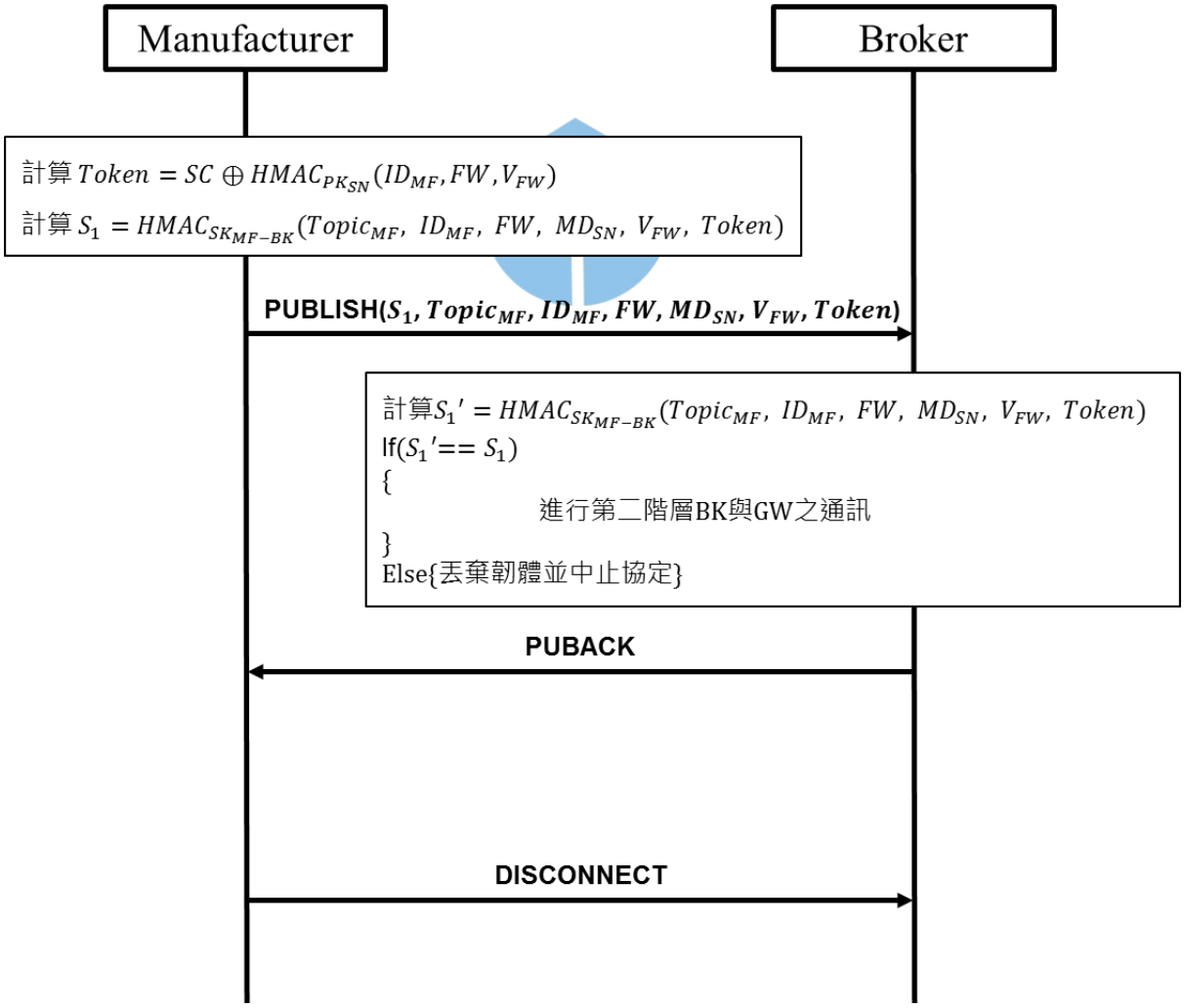

圖 9、 第一階層之韌體推送階段流程

(2) 訊息中繼站 BK→製造商伺服器 MF：PUBACK

在訊息中繼站 BK 端接收到 PUBLISH(S_1, $Topic_{MF}$, ID_{MF}, FW, MD_{SN}, V_{FW}, $Token$)控制封包之後，於訊息中繼站 BK 上計算 $S_1' = HMAC_{SK_{MF-BK}}(Topic_{MF}, ID_{MF}, FW, MD_{SN}, V_{FW}, Token)$，並驗證$S_1'$與$S_1$是否相等。若兩者相等，則可驗證來自製造商伺服器 MF 的 PUBLISH(S_1, $Topic_{MF}$, ID_{MF}, FW, MD_{SN}, V_{FW}, $Token$)控制封包沒有被竄改，且確實是由原感測裝置之製造商伺服器 MF 所傳送；若S_1'與S_1兩者不相等，則訊息中繼站 BK 立刻丟棄接收的FW並中止本協定。

承上，若S_1'與S_1兩者相等，則訊息中繼站 BK 將回傳 PUBACK 控制封包至製造商伺服器 MF，代表訊息中繼站 BK 確實至少已接收到一次來自製造商伺服器 MF 所傳送的 PUBLISH(S_1, $Topic_{MF}$, ID_{MF}, FW, MD_{SN}, V_{FW}, $Token$)控制封包。隨後，製造商伺服器 MF 將發送 DISCONNECT 控制封包給訊息中繼站 BK，關閉兩者間的連線。

5.6.4 第二階層：訊息中繼站與閘道器之通訊

若欲觸發第二階層訊息中繼站 BK 與閘道器 GW 之通訊的金鑰協商階段，必須先滿足兩個條件，第一，必須已經有閘道器 GW 向訊息中繼站 BK 完成訂閱階段之流程，詳情請參考 5.6.2 小節；第二，訊息中繼站 BK 已完成第一階層的流程，而成功取得了由製造商伺服器 MF 所推送的韌體。第二階層由金鑰協商階段、韌體推送階段組成。

5.6.4.1 金鑰協商階段

在訊息中繼站 BK 與閘道器 GW 之間，雙方採用 ECDH 以協商出相同的會談金鑰，並互相檢驗對方的數位簽章以及憑證，以驗證對方的身分。第二階層之金鑰協商流程，如圖 10 所示。

(1) 訊息中繼站 BK→閘道器 GW：PUSH($Cert_{BK}$, M_3, ID_{BK}, Q_{BK})

訊息中繼站 BK 端會利用橢圓曲線密碼學以產生短期私鑰d_{BK}以及計算出短期公鑰$Q_{BK} = d_{BK}G_{BK-GW}$，再利用自身的長期私鑰b_{BK}對訊息中繼站 BK 識別碼 ID_{BK}、短期公鑰 Q_{BK} 進行簽章，以計算出簽章 $M_3 = Sign_{b_{BK}}(ID_{BK}, Q_{BK})$。隨後，訊息中繼站 BK 傳送 PUSH($Cert_{BK}$, M_3, ID_{BK}, Q_{BK})控制封包至閘道器 GW。

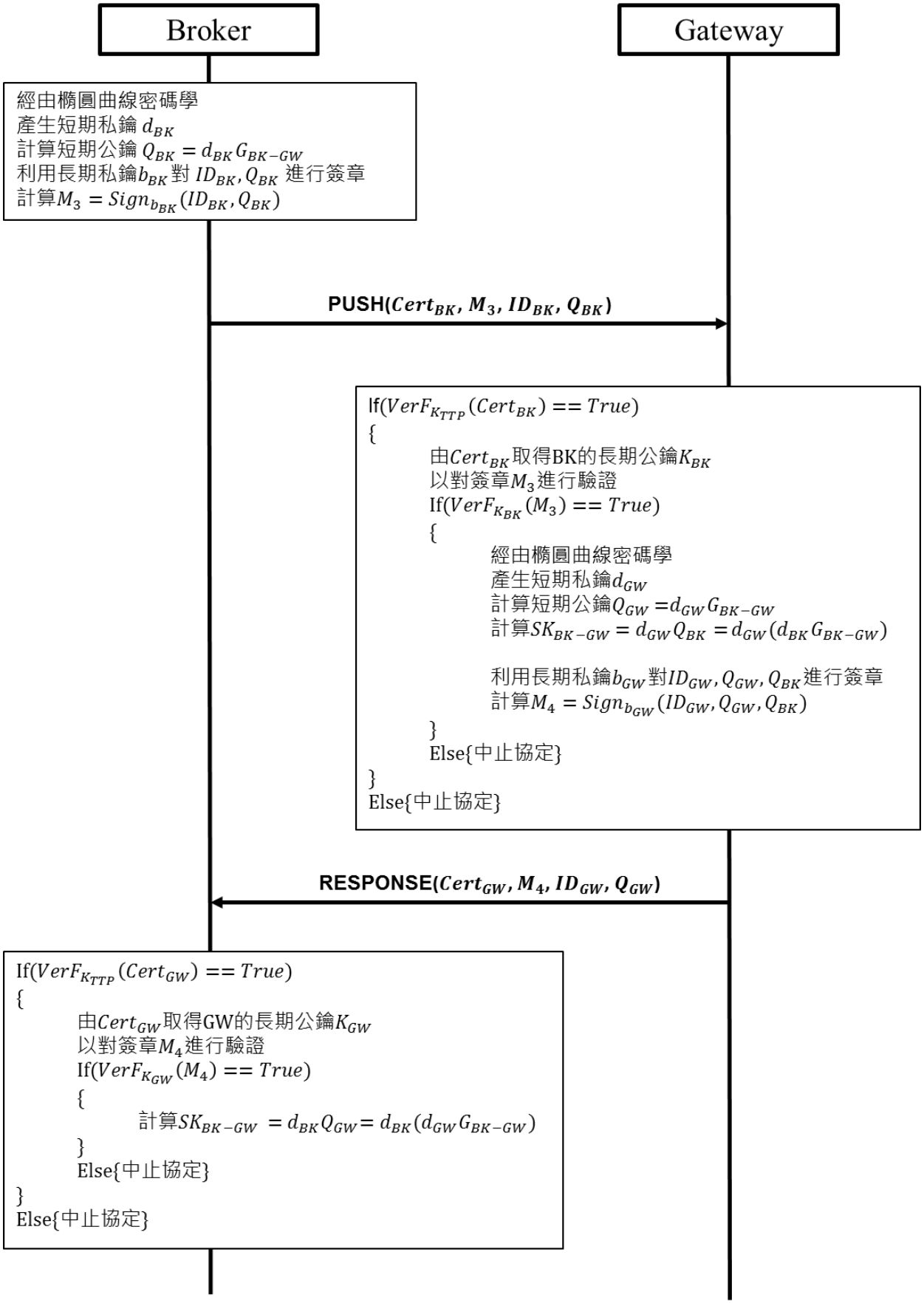

圖 10、 第二階層之金鑰協商階段流程

(2) 閘道器 GW→訊息中繼站 BK：RESPONSE($Cert_{GW}$, M_4, ID_{GW}, Q_{GW})

閘道器 GW 端接收到來自訊息中繼站 BK 傳送的 PUSH($Cert_{BK}$, M_3, ID_{BK}, Q_{BK})控制封包後，將利用先前已預載的公鑰K_{TTP}以對憑證$Cert_{BK}$進行驗證，確認$Cert_{BK}$是否由該 TTPKGC 所簽發。若K_{TTP}可成功驗證$Cert_{BK}$，代表$Cert_{BK}$確實由 TTPKGC 所簽發。若驗證$Cert_{BK}$確實由 TTPKGC 所簽發，則閘道器 GW 可以從$Cert_{BK}$取得訊息中繼站 BK 的長期公鑰K_{BK}，以驗證簽章M_3。若成功驗證M_3，則代表ID_{BK}、Q_{BK}由訊息中繼站 BK 傳送至閘道器 GW 的過程中未被竄改，且訊息確實由合法的訊息中繼站 BK 所傳送。若上述過程中發生無法成功驗證憑證$Cert_{BK}$或簽章M_3的情況，則立刻中止本協定。

承上，若閘道器 GW 端成功驗證了$Cert_{BK}$及M_3，於閘道器 GW 上利用橢圓曲線密碼學以產生短期私鑰d_{GW}，並計算出短期公鑰$Q_{GW} = d_{GW}G_{BK-GW}$，至此閘道器 GW 可以計算出會談金鑰$SK_{BK-GW} = d_{GW}Q_{BK} = d_{GW}(d_{BK}G_{BK-GW})$。隨後閘道器 GW 利用自身的長期私鑰$b_{GW}$對閘道器 GW 識別碼$ID_{GW}$以及短期公鑰$Q_{GW}$、$Q_{BK}$進行簽章，以計算出簽章$M_4 = Sign_{b_{GW}}(ID_{GW}, Q_{GW}, Q_{BK})$。隨後，閘道器 GW 傳送 RESPONSE($Cert_{GW}$, M_4, ID_{GW}, Q_{GW})控制封包至訊息中繼站 BK。

在訊息中繼站 BK 接收到 RESPONSE($Cert_{GW}$, M_4, ID_{GW}, Q_{GW})控制封包之後，將利用先前已預載的 TTPKGC 之公鑰K_{TTP}以對憑證$Cert_{GW}$進行驗證，確認$Cert_{GW}$是否由該 TTPKGC 所簽發。若該K_{TTP}可成功驗證$Cert_{GW}$，代表$Cert_{GW}$確實由 TTPKGC 所簽發。若驗證$Cert_{GW}$確實由 TTPKGC 所簽發，則訊息中繼站 BK 可以從$Cert_{GW}$取得閘道器 GW 的長期公鑰K_{GW}，以驗證簽章M_4。若成功驗證M_4，則代表ID_{GW}、Q_{GW}由閘道器 GW 傳送至訊息中繼站 BK 的過程中未被竄改，且訊息確實由合法的閘道器 GW 所傳送。若上述過程中發生無法成功驗證憑證$Cert_{GW}$或簽章M_4的情況，則立刻中止本協定。

承上，若訊息中繼站 BK 端成功驗證了$Cert_{GW}$及M_4，則計算出會談金鑰$SK_{BK-GW} = d_{BK}Q_{GW} = d_{BK}(d_{GW}G_{BK-GW})$。至此完成了以 ECDH 在訊息中繼站 BK 與閘道器 GW 間協商出一把相同的會談金鑰SK_{BK-GW}，並且同時完成了訊息中繼站 BK 與閘道器 GW 彼此的身分鑑別，因此確保此會談金鑰SK_{BK-GW}只會由合法的訊息中繼站 BK 與閘道器 GW 共同知悉，而第三方並無法得知SK_{BK-GW}。

5.6.4.2 韌體推送階段

韌體推送階段時，為了檢驗推送的韌體是否確實由合法的訊息中繼站 BK 所提供，而運用在初始化階段所協商出來的會談金鑰以進行 HMAC 運算，以進行韌體的完整性以及韌體來源合法性的驗證，流程如圖 11 所示。

(1) 訊息中繼站 BK→閘道器 GW：PUBLISH($Topic_{MF}$, FW, MD_{SN}, V_{FW}, $Token$, S_2, ID_{GW})

在訊息中繼站 BK 端使用金鑰協商階段所計算的會談金鑰SK_{BK-GW}以計算出$S_2 = HMAC_{SK_{BK-GW}}(Topic_{MF}, ID_{GW}, FW, MD_{SN}, V_{FW}, Token)$。然後，訊息中繼站 BK 傳送 PUBLISH($Topic_{MF}$, FW, MD_{SN}, V_{FW}, $Token$, S_2, ID_{GW})控制封包至閘道器 GW。

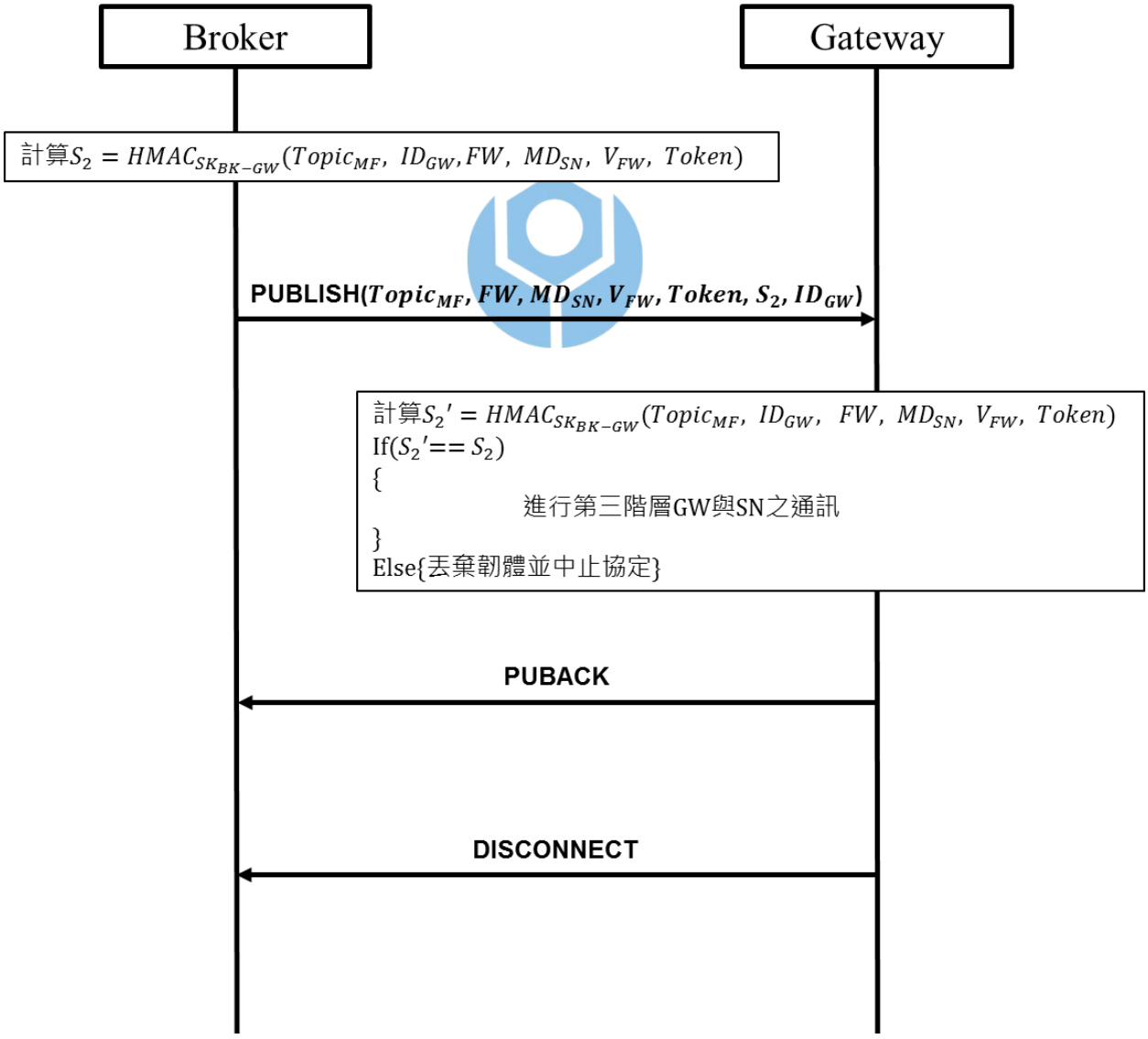

圖 11、 第二階層之韌體推送階段流程

(2) 閘道器 GW→訊息中繼站 BK：PUBACK

在閘道器 GW 端接收到 PUBLISH($Topic_{MF}$, FW, MD_{SN}, V_{FW}, $Token$, S_2, ID_{GW})控制封包之後，於閘道器 GW 上計算$S_2' = HMAC_{SK_{BK-GW}}(Topic_{MF}, ID_{GW}, FW, MD_{SN}, V_{FW}, Token)$，並驗證$S_2'$與$S_2$是否相等。若兩者相等，則來自訊息中繼站 BK 的 PUBLISH($Topic_{MF}$, FW, MD_{SN}, V_{FW}, $Token$, S_2, ID_{GW})控制封包沒有被竄改，且確實是由合法的訊息中繼站 BK 所傳送；若S_2'與S_2兩者不相等，則閘道器 GW 立刻丟棄接收的FW並中止本協定。

承上，若S_2'與S_2兩者相等，則閘道器 GW 將傳送 PUBACK 控制封包給訊息中繼站 BK，代表閘道器 GW 確實至少接收到一次來自訊息中繼站 BK 所傳送的 PUBLISH($Topic_{MF}$, FW, MD_{SN}, V_{FW}, $Token$, S_2, ID_{GW})控制封包。隨後，閘道器 GW 將發送 DISCONNECT 控制封包給訊息中繼站 BK，關閉兩者間的連線。

5.6.5 第三階層：閘道器與感測裝置之通訊

第三階層僅由韌體推送階段構成，如圖 12 所示。相較於第一、第二階層缺少了金鑰協商階段，原因在於第三階層是閘道器與感測裝置彼此進行通訊，若採用金鑰協商的方式，閘道器必須逐一與連接的感測裝置產生會談金鑰，然而，感測裝置在環境中的部署數量可能非常龐大，若採金鑰協商的方式使得閘道器分別與多個感測裝置協商出會談金鑰，對閘道器而言將是不小的運算負擔；而對感測裝置來說，其硬體計算能力有限，未必能負擔金鑰協商之運算。

感測裝置並不在意與其通訊之閘道器之身分是否合法，但是必須確認其所接收的檔案必須擁有韌體完整性、韌體來源合法性，以及感測裝置韌體適用性，隨後感測裝置才可進行安裝新版韌體的動作。以下為本論文對於三個特性的定義：

- 韌體完整性：即確保韌體在推送的過程中，沒有遭受攻擊者的竄改。
- 韌體來源合法性：即確保韌體的來源是原感測裝置之製造商所提供。
- 感測裝置韌體適用性：即確保韌體適用於特定型號的感測裝置。例如：A 製造商生產了兩種型號的裝置 MD-01、SA-02。而現在 A 製造商欲針對其生產的型號 MD-01 感測裝置進行韌體更新，即便攻擊者採用惡意手法將韌體調換成僅適用於型號 SA-02 感測裝置之韌體，透過本論文的設計，型號 MD-01 之感測裝置並不會安裝了僅適用於型號 SA-02 的韌體，以免造成型號 MD-01 之感測裝置其功能停擺。

因此，我們在韌體推送階段利用感測裝置在出廠前，由其製造商寫入的祕密值SC、預載金鑰PK_{SN}，感測裝置即可檢驗其所接收的檔案是否為惡意韌體。

5.6.5.1 韌體推送階段

閘道器 GW 將此次欲進行韌體更新的目標感測裝置型號MD_{SN}以廣播的方式傳送給感測裝置 SN，感測裝置 SN 將比對自身的裝置型號MD_{SN}^{L}是否與MD_{SN}相符，若相符，即代表此感測裝置 SN 為此次韌體更新之目標型號裝置。隨後，感測裝置 SN 會回傳感測裝置 SN 的識別碼ID_{SN}以及其目前正在執行的韌體版本號碼V_{FW}^{C}至閘道器 GW。

閘道器 GW 接收到感測裝置 SN 的識別碼ID_{SN}與韌體版本號碼V_{FW}^{C}後，將比對V_{FW}是否大於V_{FW}^{C}，若V_{FW}大於 V_{FW}^{C}，則代表目前在感測裝置 SN 上的韌體是舊版的，則閘道器 GW 會根據ID_{SN}將新版本韌體推送至對應的正在執行舊版韌體之感測裝置 SN，閘道器 GW 會先進行雜湊運算得到$F = H(ID_{GW}, Token, FW, V_{FW})$，隨後將參數$F$、$ID_{GW}$、$Token$、$FW$、$V_{FW}$傳送給感測裝置 SN。

感測裝置 SN 端將會再次比對V_{FW}是否大於V_{FW}^{C}，因為感測裝置 SN 並不信任閘道器 GW 傳來的訊息。若V_{FW}大於 V_{FW}^{C}，則執行雜湊運算得到$F' = H(ID_{GW}, Token, FW, V_{FW})$。隨後，比對$F'$與$F$是否相等，若$F'$與$F$相等，則由感測裝置 SN 的內部儲存空間取得原製造商在出廠前寫入的預載金鑰PK_{SN}，以計算$SC' = Token \oplus HMAC_{PK_{SN}}(ID_{MF}, FW, V_{FW})$，若$SC'$與$SC$相等，代表推送至感測裝置 SN 的韌體確實是由原製造商伺服器 MF 所提供、韌體未曾被竄改，以及韌體適用此感測裝置 SN 型號，因此感測裝置 SN 將安裝此韌體；反之，若SC'與SC不相等，則代表感測裝置 SN 接收的韌體可能並非由其製造商伺服器 MF 所提供、韌體可能已遭受攻擊者竄改，或是韌體不適用此感測裝置 SN 的型號，如此，感測裝置 SN 將會丟棄其接收的韌體。

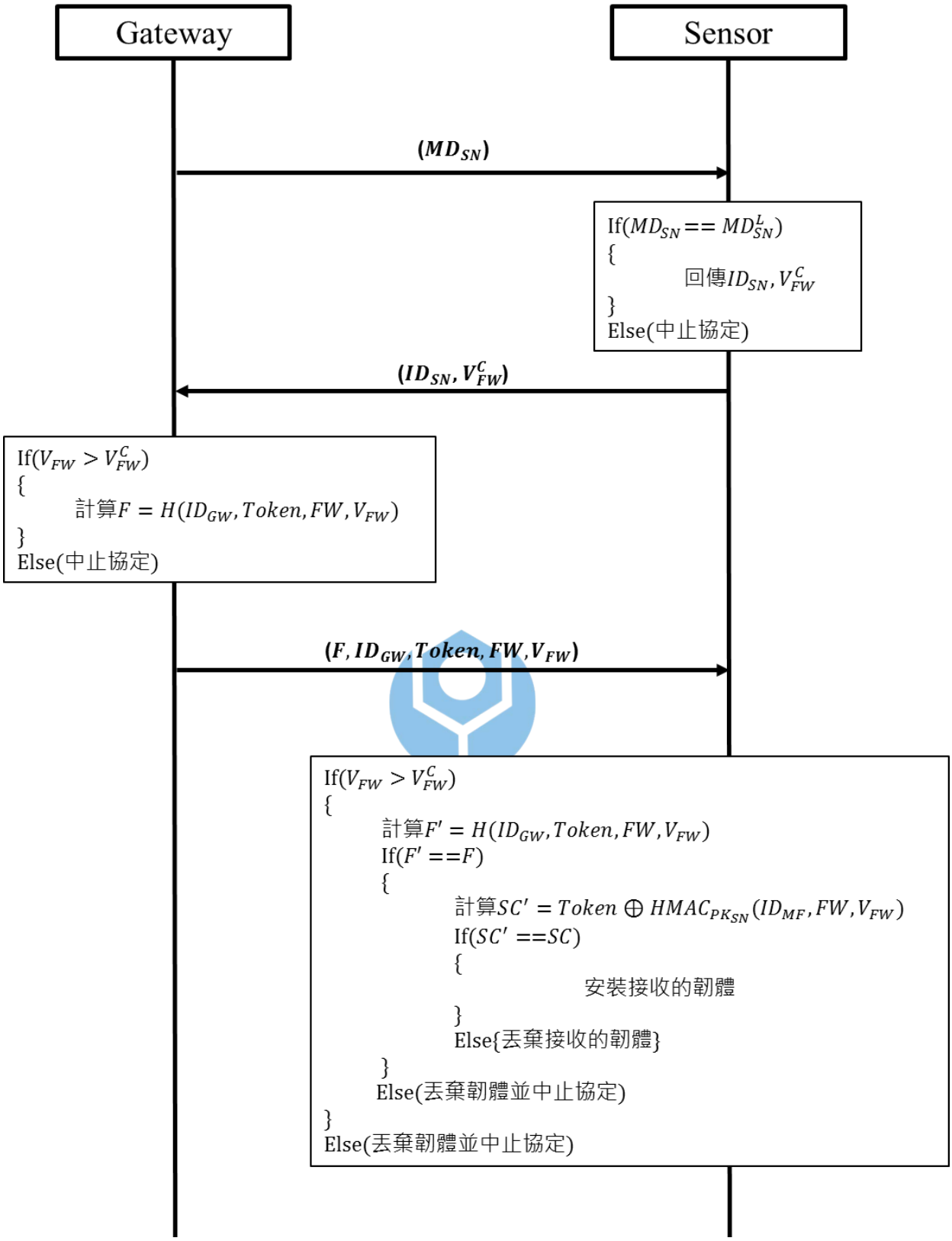

圖 12、 第三階層之韌體推送階段流程

第六章 安全性分析

在本章節將針對本論文設計的協定進行安全性的探討，探討的網路攻擊手法包含竊聽攻擊(Eavesdropping Attack)、金鑰安全性(Key Security)、中間人攻擊(Man-in-the-middle Attack, MITM)、重送攻擊(Replay Attack)、偽冒攻擊(Impersonation Attack)、前向安全性(Forward Secrecy)。

6.1 竊聽攻擊之防禦

竊聽攻擊為攻擊者針對傳輸的訊息進行攔截、監控，攻擊者的目標在於獲得重要訊息，然而在本論文中的情境，製造商所發布的韌體是要讓特定型號的裝置進行韌體更新，因此不需要對韌體進行加密，而是公開免費傳送至感測裝置。但是，即使攻擊者在通訊過程中發動竊聽攻擊，其並無法獲得第一階層通訊實體之間的會談金鑰SK_{MF-BK}及第二階層通訊實體之間的會談金鑰SK_{BK-GW}，因為SK_{MF-BK}是由製造商伺服器及訊息中繼站利用 ECDH 演算法計算出來的；而SK_{BK-GW}則是由訊息中繼站及閘道器利用 ECDH 演算法計算出來的，所以攻擊者無法藉由竊聽訊息的方式而取得SK_{MF-BK}及SK_{BK-GW}，一旦攻擊者推送惡意韌體將會被檢驗出來，因此竊聽攻擊將無法對本論文設計的協定造成危害。

6.2 金鑰安全性

在本論文所設計的協定中使用了多種金鑰，以達成通訊實體間的身分鑑別及韌體完整性的驗證，因此，須針對各金鑰的安全性作探討。若攻擊者的目標在於透過實體攻擊取得預載金鑰PK_{SN}，長期私鑰b_{MF}、b_{BK}、b_{GW}，短期私鑰d_{MF}、d_{BK}、d_{GW}，以完成協定的金鑰協商階段或是韌體推送的階段。由於這些金鑰並不會在通訊過程中進行傳送，且這些金鑰皆被存放於硬體安全模組內，所以攻擊者無法透過竊聽或實體攻擊取得這些金鑰。再者，由於在協定的金鑰協商階段皆採用 ECDH 演算法，攻擊者雖然可以取得橢圓曲線密碼學之短期公鑰Q_{MF}、Q_{BK}、Q_{GW}，但因為橢圓曲線離散對數問題，其無法得知橢圓曲線密碼學之短期私鑰d_{MF}、d_{BK}、d_{GW}，所以攻擊者無法在進行通訊的過程中計算或直接取得會談金鑰SK_{MF-BK}、SK_{BK-GW}。綜合上述，故可確保本論文設計的協定之金鑰安全性。

6.3 中間人攻擊之防禦

中間人攻擊為攻擊者介入欲進行通訊的雙方實體之通訊過程，使通訊雙方誤以為正在跟合法的對象進行通訊，然而事實上整個通訊過程都已由攻擊者所掌控，攻擊者可以竊聽訊息並竄改訊息，但訊息的接收者渾然不知其所接收的訊息已遭

中間人竄改，發生中間人攻擊的根本原因在於通訊雙方缺乏驗證對方身分的安全機制。

若攻擊者欲透過中間人攻擊，而成功達到在感測裝置上安裝惡意韌體的目的，則攻擊者必須要分別與第一階層的製造商伺服器及訊息中繼站分別進行 ECDH 金鑰協商，然而，由於在金鑰協商階段中，製造商伺服器與訊息中繼站皆會驗證對方的憑證與數位簽章，而攻擊者沒有由可信任的第三方金鑰產生中心所發放的憑證$Cert_{MF}$、$Cert_{BK}$，因此就會中止金鑰協商的流程；即使攻擊者透過竊聽方式取得了由可信任的第三方金鑰產生中心所簽發的合法憑證，但是攻擊者無法得知製造商伺服器的長期私鑰b_{MF}及訊息中繼站的長期私鑰b_{BK}，因此攻擊者無法分別與製造商伺服器及訊息中繼站完成金鑰協商流程。同理可證，在第二階層中，攻擊者亦無法分別與訊息中繼站、閘道器完成金鑰協商的流程。最後，在第三階層中，則是檢驗了由製造商伺服器在感測裝置出廠前寫入的祕密值SC，只要攻擊者推送惡意韌體至感測裝置，則感測裝置計算出來的祕密值SC'與預載的祕密值SC會不相等，感測裝置將會丟棄接收的韌體。綜合上述，故本論文所設計的協定，可以防止攻擊者利用中間人攻擊而造成感測裝置安裝惡意韌體的情況發生。

6.4 重送攻擊之防禦

重送攻擊指的是攻擊者利用竊聽的方式取得在不安全通道中所傳輸的訊息後，經過了一段時間後，將此訊息再度傳送給接收者以通過驗證，並存取伺服器上的資源。

若攻擊者欲透過重送先前竊聽的訊息而通過身分鑑別，而達成在感測裝置上安裝惡意韌體的目的。在第一階層中，若攻擊者將先前竊聽的訊息不作任何更動而進行重送，由於本協定使用挑戰回應法(Challenge-Response)，所以有進行亂數勾稽，因此簽章M_1、M_2會隨著不同的會談(Session)而變動，使得製造商伺服器與訊息中繼站將無法完成金鑰協商流程。同理可證，在第二階層中，攻擊者若進行重送攻擊，由於有進行亂數勾稽，簽章M_3、M_4會隨著不同的會談而變動，使得訊息中繼站與閘道器無法完成金鑰協商流程。

在第三階層中，則運用製造商在感測裝置出廠前所預載的祕密值SC以判別收到的韌體是否為惡意韌體，即使攻擊者竊聽了前次由製造商伺服器發布舊版韌體推送的訊息而重送至感測裝置，會使得感測裝置所計算的SC'與SC不相等，所以舊版韌體並不會被感測裝置進行安裝，以避免攻擊者利用重送攻擊，造成感測裝置的韌體被降版的情況發生。因為韌體更新的目的通常在於修補已發現的漏洞，若攻擊者可以讓感測裝置安裝了舊版韌體，也就代表攻擊者可以利用已發現之漏洞對感測裝置進行攻擊，所以本論文在設計協定時，已考量此種攻擊手法並進行

防禦。故本論文所設計的協定，可以防止攻擊者利用重送攻擊而造成感測裝置安裝惡意韌體的情況發生。

6.5 偽冒攻擊之防禦

偽冒攻擊代表的是有攻擊者欲偽裝成合法的通訊實體，向對方進行通訊而未被對方察覺。

若攻擊者欲透過偽冒攻擊將惡意韌體推送至感測裝置上，而達成感測裝置接收並安裝了惡意韌體的目的。在本論文之第一階層之金鑰協商階段中，若有攻擊者想假冒成合法通訊雙方之一(製造商伺服器或訊息中繼站)，與對方進行通訊，則因為攻擊者並沒有由可信任的第三方金鑰產生中心所簽發的憑證$Cert_{MF}$、$Cert_{BK}$，所以協定將中止；即使攻擊者竊取了合法通訊者的憑證，但因為攻擊者並不知道合法通訊者的長期私鑰b_{MF}、b_{BK}，故攻擊者自行產生的簽章將會無法被對方驗證成功，而無法與對方共同協商出會談金鑰SK_{MF-BK}，因此，攻擊者無法經由偽冒攻擊而與合法通訊者完成金鑰協商階段。即使攻擊者直接將惡意韌體傳送至訊息中繼站，但因為攻擊者並沒有經由合法協商過程產生的會談金鑰SK_{MF-BK}，所以訊息中繼站可以計算出S_1'與S_1不相等，而得知其所接收的檔案為惡意韌體，隨後丟棄該韌體並中止協定。同理可證，第二階層之金鑰協商階段中，若有攻擊者想假冒成合法通訊雙方之一(訊息中繼站或閘道器)，與對方進行通訊，攻擊者將無法與合法通訊者協商出會談金鑰SK_{BK-GW}，因此，攻擊者無法經由偽冒攻擊而與合法通訊者完成金鑰協商階段。即使攻擊者直接將惡意韌體傳送至合法閘道器，但因為攻擊者並沒有經由合法協商過程產生的會談金鑰SK_{BK-GW}，則閘道器可以計算出S_2'與S_2不相等，而檢驗其所接收的檔案為惡意韌體，隨後丟棄該韌體並中止協定。

在第三階層中，若攻擊者嘗試利用閘道器，將惡意韌體推送給感測裝置，則感測裝置計算出來的祕密值SC'與預載的祕密值SC會不相等，因此而檢驗出感測裝置所接收的韌體是惡意韌體，隨後丟棄此韌體並中止協定。故本論文所設計的協定，可以防止攻擊者利用偽冒攻擊而造成感測裝置安裝惡意韌體的情況發生。

6.6 前向安全性

前向安全性指的是即使攻擊者取得當下通訊雙方的會談金鑰，也無法推得先前通訊所使用的會談金鑰，因此先前進行的通訊仍是安全的。

若攻擊者欲取得當下會談金鑰，而達成取得先前的會談金鑰的目標，因為在本論文提出的第一、第二階層之金鑰協商階段，所採用的協商方法為 ECDH，由

於每次進行 ECDH 的會談時，通訊雙方必須藉由橢圓曲線密碼學產生短期私鑰，並計算出短期公鑰，並以自身的短期私鑰與對方所提供的短期公鑰以計算出會談金鑰。即使攻擊者取得當下的會談金鑰，但是每次的會談金鑰皆是獨立的隨機值，會隨著不同的會談而改變，所以攻擊者無法藉由目前得到的會談金鑰推得先前的會談金鑰，故過去的通訊仍是安全的。綜合上述，因此本論文所設計的協定可以達成前向安全性。

第七章 結論

由於物聯網的快速發展，目前在工廠或是家庭中許多裝置都具有連網能力。然而，具有連線功能的裝置，同時也代表著攻擊者可能透過網路對裝置進行攻擊，而目前裝置製造商一旦發現裝置存在著安全性漏洞，雖然會提供新版本的韌體進行修補，但是通常必須由裝置的管理者手動下載新版本的韌體並將其安裝於裝置上。但是，從裝置製造商發布新版韌體後，到新版韌體真正被安裝於裝置上，往往歷時長久。首先，裝置管理者可能不知曉主動進行韌體更新的重要性，或者是不知道有新版韌體的發布；再者，被部署於環境中的感測裝置數量十分龐大，且可能這些感測裝置是由不同的製造商所生產，或者是同家製造商不同型號而構成，所以該如何有效率地管理並監控這些裝置是必須被關注的議題。

因此，本論文針對上述議題提出一套解決方法，設計了基於 MQTT 協定架構之安全物聯網韌體更新機制。在製造商伺服器與訊息中繼站之間，以及訊息中繼站與閘道器之間，皆採用 MQTT 協定架構進行韌體推送，並利用橢圓曲線迪菲-赫爾曼金鑰交換、數位簽章、金鑰雜湊訊息鑑別碼，以檢驗韌體完整性以及韌體的提供者。而在閘道器與感測裝置之間，雖然不屬於 MQTT 協定架構，但是本論文有設計一套機制以金鑰雜湊訊息鑑別碼檢驗祕密值，若祕密值被成功驗證，感測裝置才會安裝此韌體；否則，則將檔案視為惡意韌體並丟棄。

在本論文的情境下，感測裝置製造商一旦發布新版本的韌體後，便會開始自動將新版韌體檔案推送至目標裝置，透過這樣的機制，可以避免攻擊者針對仍在運行舊版韌體的裝置之漏洞進行攻擊。然而，準備安裝於裝置上的韌體，其必須符合韌體完整性、韌體來源合法性，以及感測裝置韌體適用性的特性，新版韌體須符合前述三個特性，才會被感測裝置執行安裝。否則，感測裝置若安裝了由攻擊者發送的惡意韌體，可能會導致裝置無法運作、資料外洩等風險。

最後，本論文對自行設計的機制，進行了安全性分析，確保提出的協定可以抵擋竊聽攻擊、中間人攻擊、重送攻擊、偽冒攻擊等手法，且達到金鑰安全性及前向安全性，因此驗證了協定的安全性，感測裝置將不會安裝惡意韌體。在未來的研究發展目標是找尋適合的物聯網環境將此協定進行實作，分析協定是否能符合實際上裝置韌體更新的安全要求，以及效能上是否符合實務需求。

參考文獻

[1] B.-C. Choi, S.-H. Lee, J.-C. Na, and J.-H. Lee, "Secure firmware validation and update for consumer devices in home networking," *IEEE Transactions on Consumer Electronics,* vol. 62, no. 1, pp. 39-44, 2016.

[2] A. Mohan, "Cyber Security for Personal Medical Devices Internet of Things," *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on.* pp. 372-374, 2014.

[3] " IoT security threats are skyrocketing, AT&T reveals, " [Online]. Available:http://www.fiercewireless.com/wireless/iot-security-threats-are-skyrocketing-at-t-reveals (Accessed: March, 2017)

[4] "Gartner Identifies the Top 10 Internet of Things Technologies for 2017 and 2018, " [Online]. Available: http://www.gartner.com/newsroom/id/3221818 (Accessed: March, 2017)

[5] M. Steger, C. Boano, M. Karner, J. Hillebrand, W. Rom, and K. Romer, "SecUp: Secure and Efficient Wireless Software Updates for Vehicles," *Digital System Design (DSD), 2016 Euromicro Conference on.* pp. 628-636, 2016.

[6] 「為何駭客特別愛用 IoT 裝置當作攻擊跳板，裝置管理權責劃分不易是主因 」[Online]. Available: http://www.ithome.com.tw/news/112861 (Accessed: March, 2017)

[7] H. Krawczyk, M. Bellare, and R. Canetti, "RFC 2104: HMAC: Keyed-hashing for message authentication," 1997. [Online]. Available: https://tools.ietf.org/html/rfc2104 (Accessed: April, 2017)

[8] " SEC 1: Elliptic Curve Cryptography, " [Online]. Available: http://www.secg.org/sec1-v2.pdf (Accessed: May, 2017)

[9] "MQTT Version 3.1.1 Plus Errata 01, " [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (Accessed: April, 2017)

[10] " HiveMQ MQTT Essentials, " [Online]. Available: http://www.hivemq.com/blog/mqtt-essentials/ (Accessed: April, 2017)

[11] S. Lee, H. Kim, D.-K. Hong, and H. Ju,, "Correlation analysis of MQTT loss and delay according to QoS level," *Information Networking (ICOIN), 2013 International Conference on.* pp. 714-717, 2013.

[12] P. Thota, and Y. Kim, "Implementation and Comparison of M2M Protocols for Internet of Things," *Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD), 2016 4th Intl Conf on.* pp. 43-48, 2016.

[13] “ The Constrained Application Protocol, ” [Online]. Available: https://tools.ietf.org/html/rfc7252 (Accessed: April, 2017)

[14] A. J. Poulter, S. J. Johnston, and S. J. Cox, “SRUP: The secure remote update protocol,” *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on.* pp. 42-47, 2016.

[15] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, and A. Panya, “Authorization mechanism for MQTT-based Internet of Things,” *Communications Workshops (ICC), 2016 IEEE International Conference on.* pp. 290-295, 2016.

[16] M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar, “Secure MQTT for Internet of Things (IoT), ” *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on.* pp. 746-751, 2015

[17] A. K. Ranjan, and M. Hussain, “Terminal Authentication in M2M Communications in the Context of Internet of Things,” *Procedia Computer Science,* vol. 89, pp. 34-42, 2016.

[18] Lavanya, Natarajan, “Lightweight Authentication for COAP based IOT,” *Proceedings of the 6th International Conference on the Internet of Things(IoT),* pp. 167-168, 2016.

[19] Omaimah Omar Bamasag and Kamal Youcef-Toumi, “Towards Continuous Authentication in Internet of Things Based on Secret Sharing Scheme,” *Proceedings of the 2015 Workshop on Embedded Systems Security (WESS'15),* Amsterdam, The Netherlands, 2015.

[20] I. Butun, M. Erol-Kantarci, B. Kantarci, and H. Song, “Cloud-centric multi-level authentication as a service for secure public safety device networks,” *IEEE Communications Magazine,* vol. 54, no. 4, pp. 47-53, 2016.

[21] J. L. Hernandez-Ramos, M. P. Pawlowski, A. J. Jara, A. F. Skarmeta, and L. Ladid, “Toward a Lightweight Authentication and Authorization Framework for Smart Objects,” *IEEE Journal on Selected Areas in Communications,* vol. 33, no. 4, pp. 690-702, 2015.

[22] P. Kumar, A. Gurtov, J. Iinatti, M. Ylianttila, and M. Sain, “Lightweight and Secure Session-Key Establishment Scheme in Smart Home Environments,” *IEEE Sensors Journal,* vol. 16, no. 1, pp. 254-264, 2016.

[23] H. Chandra, E. Anggadjaja, P. S. Wijaya, and E. Gunawan, “Internet of Things: Over-the-Air (OTA) firmware update in Lightweight mesh network protocol for smart urban development,” *Communications (APCC), 2016 22nd Asia-Pacific Conference on.* pp. 115-118, 2016.

[24] R. Hassan, K. Markantonakis, and R. N. Akram, “Can You Call the Software in Your Device be Firmware?,” *e-Business Engineering (ICEBE), 2016 IEEE 13th International Conference on.* pp. 188-195, 2016.

[25] D. K. Nilsson, L. Sun, and T. Nakajima, “A Framework for Self-Verification of Firmware Updates over the Air in Vehicle ECUs,” *GLOBECOM Workshops*, pp. 1-5, 2008.