



國立臺灣科技大學
資訊工程系

碩士學位論文

確保事件日誌前向安全性的
物聯網裝置入侵偵測機制

An Intrusion Detection Scheme for IoT Devices

Supporting Forward-secure Event Logs

黃哲彬

M10015909

指導教授：鄧惟中 博士

中華民國 一百零八 年 七 月 十五 日





碩士學位考試委員審定書



M10015909

指導教授：鄧惟中

本校 資訊工程系 黃哲彬 君

所提之論文：

確保事件日誌前向安全性的物聯網裝置入侵偵測機制

經本委員會審定通過，特此證明。

學校考試委員會

委

員：

鄧惟中
黃哲彬
李漢南
趙文瑞

指導教授：

鄧惟中

學程主任：

馬輝文

系(學程)主任、所長：

中華民國 108 年 7 月 15 日

摘要

隨著近年來 IoT 物聯網的蓬勃成長，眾多不安全的裝置和惡意軟體的威脅也隨之而來，恐成為物聯網未來發展的最大隱憂之一。其中常見的就是入侵者在登入 IoT 設備取得控制權後，在某些特定目錄中植入惡意檔案/軟體，進而遂行其接下來的行動。通常嵌入式系統裝置在使用者登入以及發生系統異常事件時，會在日誌檔案中紀錄這些內容，包含入侵者的動作，都會被記錄下來。但是通常入侵者不但不會讓系統管理者發現他的入侵動作，也不希望在系統內留下相關的足跡和證據，所以就有可能會想辦法刪除或是修改事件發生當時的日誌檔案內容。

因此，本研究設計了一個 IoT 裝置的軟體架構，其功能為當入侵者在一個或是多個特定的可執行目錄下植入檔案時，IoT 裝置能夠即時發出警告通知系統管理者採取相對應的措施。此外也利用了具備前向安全性的加密演算法，即使在入侵者成功取得設備的控制權之後，裝置的系統仍然可以確保過去所有已產生的日誌檔案內容都是安全而無法被修改的。

本研究以樹莓派(不可信賴的裝置)、安全伺服器(可信賴的機器)和乙太網路交換器等環境設計，來模擬驗證本系統是否可以發揮預期的作用。根據實驗結果，當入侵者取得IoT裝置的控制權並新增或修改特定目錄下的檔案時，系統會將異常事件的內容以具有前向安全特性的加密演算法被加密，系統管理者也可以第一時間得知此異常事件，安全伺服器也會同步接收到此加密過異常事件紀錄。因為持有共同金鑰，所以可以解密出此異常紀錄的明文，並也可根據此紀錄對IoT裝置做進一步安全查核的動作。而且日誌檔案皆以具有前向安全特性的加密演算法被加密過，以至於入侵者無法去偽造過去的事件記錄，因此在異常事件發生之前的日誌檔案安全性得以確保。

Abstract

With the rapid growth of the IoT (Internet of Things), information security is considered one of the biggest concerns to the upcoming applications. It is common for intruders to implant malware in some directory after logging in to the IoT device to gain control, or after utilizing loopholes of network communication protocols. Usually, intruders try to remove footprints and evidences of intrusion action, like the records of login and abnormal events in the log files. Therefore, this research aims to realize a mechanism to secure intrusion detection ability of IoT devices.

The core function of the proposed mechanism is that when an intruder implants or alters a file in any one of the folders for executable binaries, the soon to be compromised IoT device immediately issue a warning to notify the system administrator. In addition, a cryptographic algorithm with forward security is utilized. Even after the intruder successfully gets control of the device, the operating system of the device ensures that all existing records in the log file cannot be faked.

In this research, an experimental implementation including a Raspberry Pi (untrusted device), a secure server (trusted machine), and an Ethernet switch are deployed to simulate whether the system can perform its intended function. According to the experiment results, when the intruder obtains control of the IoT device and adds or modifies the file in a specific directory, the system encrypts the content of the abnormal events with an encryption algorithm with forward security features. The system administrator can also know the abnormal event as soon as possible. The secure server also receives this encrypted abnormal event record synchronously. Because the common key is held, the records of abnormal events can be decrypted, and the IoT device is able to be further checked for security according to the record. Moreover, the log files are encrypted with a cryptographic algorithm with forward security features, so that the intruder cannot falsify past event records, so the log file security before the abnormal event occurs is ensured.

誌 謝

本論文得以完成，首先也是最感謝的就是我的指導教授鄧惟中博士，非常感謝老師在跟隨他做研究的這幾年裡，對我所不吝付出的教導與關心。老師除了將豐富的專業知識傳授與我，也分享了很多過去的經歷和經驗，讓我除了在專業領域上有更近一步的深入學習，也得到了許多人生的寶貴經驗，將來在職場上也更能走的順暢，非常感謝老師。

接著要感謝的是小妃，在我就讀研究所的期間，辛苦擔起照顧家庭的責任重擔，盡心盡力的照顧兩個兒子，並在我失意挫折的時候給予安慰和努力，讓我沒有後顧之憂的可以專心攻讀學業。

最後要感謝的是公司的主管和同仁們，除了幫我分擔工作上的負擔，還讓我能夠在工作之餘，有更多充裕的時間可以準備學業並順利畢業。



目 錄

教授推薦書.....	II
論文口試委員審定書.....	III
摘要.....	IV
Abstract.....	V
誌 謝.....	VI
目 錄.....	VII
圖目錄.....	IX
第 1 章 緒論.....	1
1.1 背景.....	1
1.2 研究目標.....	1
1.3 論文架構.....	2
第 2 章 相關研究.....	4
2.1 前向安全性.....	4
2.2 前向安全性在日誌文件上的加密研究.....	6
2.2.1 介紹.....	6
2.2.2 情境.....	7
2.2.3 解決方案.....	8
2.2.4 方法論.....	8
2.3 前向安全序列聚合.....	13
2.3.1 介紹與情境.....	13
2.3.2 解決方案.....	15
2.3.3 定義與屬性.....	15
2.3.4 FssAgg MAC 方案.....	17
2.3.5 FssAgg Signature 方案.....	18
2.4 關於系統日誌保全的其它研究.....	20
2.5 inotify 檔案異動偵測技術.....	21

第 3 章 研究方法.....	23
3.1 對前向安全日誌加密演算法的改良.....	23
3.2 IoT 設備特性與容易被攻擊的原因.....	25
3.3 入侵的定義和方式.....	26
3.4 IoT 設備的限制條件	26
3.5 系統架構與設定.....	27
3.6 系統功能.....	28
3.6.1 即時檔案異動偵測.....	29
3.6.2 產生具有前向安全性的加密日誌項目	30
3.5.3 IoT 裝置與安全伺服器的通訊	30
3.7 預期目標.....	31
第 4 章 實驗設計與結果.....	32
4.1 情境與架構環境.....	32
4.2 過程與結果.....	34
4.3 實驗結果分析.....	41
第 5 章 結論.....	42
參考文獻.....	43



圖目錄

圖 2.1 前向安全示意圖	5
圖 2.2 日誌安全演算法硬體架構圖	7
圖 2.3 新增項目至日誌檔案	10
圖 2.4 入侵時間與日誌項目安全	12
圖 3.1 日誌加密演算法改良	23
圖 3.2 研究架構設定圖	27
圖 3.3 系統功能流程圖	28
圖 3.4 inotify 架構示意圖	29
圖 3.5 heartbeat 示意圖	30
圖 4.1 實驗架構與環境	32
圖 4.2 情境 1 - Raspberry Pi 之實驗結果	36
圖 4.3 情境 1 - Linux server 之實驗結果.....	36
圖 4.4 情境 2 - Raspberry Pi 之實驗結果	37
圖 4.5 情境 2 - Linux server 之實驗結果	38
圖 4.6 情境 2 - 模擬入侵者之實驗結果	38
圖 4.7 情境 3 - Raspberry Pi 之實驗結果	39
圖 4.8 情境 3 - 模擬入侵者之實驗結果	40
圖 4.9 情境 3 - Linux server 之實驗結果	40

第 1 章 緒論

1.1 背景

隨著近年來嵌入式系統(embedded system)和網路設備的快速蓬勃發展，IoT 物聯網(Internet of Things)的誕生已經是人類下一個世代科技革命的新里程碑。IoT 設備的目標在於將短距離行動資料收發器，嵌入到日常生活中的小工具或事物中，為資訊通訊的技術領域，帶來新的發展面向。以 Cisco 先前的預估，到了 2020 年，全世界的 IoT 設備數量將會達到 500 億個[1]，而根據 IHS 預測，到 2030 年全世界的 IoT 設備數量更將會大幅倍增到 1250 億個[2]。

然而，不安全的裝置和惡意軟體，恐成為 IoT 物聯網的安全隱憂。例如在 2016 年 1 月，美國醫療用品公司 St. Jude Medical 旗下產品的心臟起搏器被 FDA 證實存在著漏洞，可能會被駭客入侵[3]。心臟起搏器的功能本來是用於監測和控制患者的心臟功能，防止心臟病發作。但是一旦被成功入侵，除了可能會有耗盡電池電力的風險，或是產生不可控制的起搏，甚至是對心臟的衝擊，嚴重危害患者的生命健康。



而另外一個有名的例子是，在 2016 年 10 月 21 日知名網路服務商 Dyn 遭受殭屍網路發動三波巨大規模 DDoS 攻擊，世界各大網站服務皆因為此攻擊而中斷，包括 Amazon、Twitter、Github、PayPal 等大型網站都因此受到影響[4]。根據相關研究發現，此次 DDoS 攻擊的發起者雖然不明，但多數攻擊流量來自殭屍網路「Mirai」，利用 IPCAM、CCTV、DVR、IoT 裝置等系統進行 DDoS 攻擊。目前全世界有數十億個網路設備正在運行，類似的駭客攻擊將有可能會入侵電力網基礎設施、聯網汽車(connected car)，交通監視器、核電廠等等，將成為國家級的網路安全危機。

1.2 研究目標

一般來說，IoT 裝置存在著 5 大安全弱點，因而容易遭駭客所利用，分別是 (1) IoT 裝置本身已存在潛在可利用的漏洞、(2) 使用不安全的網路協定、雲端及行動 App 服務，或是提供不安全的軟體、韌體更新、(3) 仍保留不安全的網

路連接埠、(4) 允許未授權的系統變更，以及 (5) 授權/認證強度不夠及缺乏足夠安全的加密機制。

其中常見的現象就是入侵者在登入設備取得控制權後，在某些特定目錄如 binary folder 中植入惡意檔案/軟體，進而遂行其接下來的行動。通常嵌入式系統裝置在使用者登入以及發生系統異常事件時，會在 Log File 中紀錄這些內容，包含入侵者的動作，都會被記錄下來。但是通常入侵者不但不會讓系統管理者發現他的入侵動作，也不希望在系統內留下相關的足跡和證據，所以就有可能會想辦法刪除或是修改事件發生當時的 Log File 內容。

因此，一個安全的 IoT 裝置所必須具備的功能，是除了在入侵者植入惡意軟體時能夠即時發出警告通知系統管理者，採取相對應的措施之外，我們也必須找出一個方法，即使在入侵者成功取得設備的控制權之後，裝置的系統仍然可以確保過去所有已產生的 Log File 內容都是安全而無法被修改的。

在此，我們使用了 Linux 檔案系統 driver level 的 inotify 程式碼，來達成對特定目錄即時監控的功能，以期能在入侵者在目錄下植入檔案時，能夠被立即偵測到並第一時間通知系統管理者。另外，我們也利用了 Bruce Schneier 和 John Kelsey 所提出的前向安全性 Forward Security 的演算法機制，來確保過去產生的所有 Log File 的安全性。其原理為透過一台不可信機器(Untrusted Machine)與一台可信機器(Trusted Machine)共享一個初始密鑰(Secret Key)，利用這個密鑰來加密已產生的 Log Entry，然後將此密鑰以單向運算方式演化出一個新的密鑰，再用新的密鑰去加密下一次新產生的 Log Entry。透過雙方的資料交流，可信機器也可用密鑰來驗證不可信機器上的 Log File 是否有異常。如此一來，入侵者因為無法以反向方式取得過去的密鑰，就無法解密過去的 Log Entry 內容並加以修改，也因此無法偽造出假的 Log Entry 而來掩飾或抹滅其入侵行為的跡證。

本篇論文研究目的並非是為了在事前預防未經授權的入侵行為，而是主要著重於入侵事件當時的即時警告通知以及事後的稽核和驗證。

1.3 論文架構

本篇論文一共有五個章節，第二章介紹 inotify 功能與原理、前向安全性

(Forward Security)相關研究內容，以及其它進階的前向安全性相關論文；第三章介紹研究方法，以及功能方塊；第四章介紹本論文的實驗環境、設計，以及呈現實驗的結果；最後第五章提出本論文的結論和貢獻。




第 2 章 相關研究

2.1 前向安全性

前向安全(forward security)該定義最早是由 Mihir Bellare 和 Sara K. Miner 在 CRYPTO' 99 上提出的關於數位簽章的性質[5], 而 perfect forward secrecy 則是由 Christoph G. Günther 在 EUROCRYPT ' 89 提出的, 其最初用於定義工作階段金鑰切換式通訊協定的一種安全性保證機制[6]。

1997 年, Anderson 提出了前向安全數位簽章的概念。前向安全的簽名把公開金鑰的存留期劃分為很多時段, 每個時段的私密金鑰各不相同, 在當前時段的簽名私密金鑰洩露後不影響在洩露以前時段的私密金鑰與簽名的安全[7]。

完美前向安全



完美前向安全(Perfect Forward secrecy)的主要含義是：就是用來產生密鑰(session key)的長期密鑰(long-term key)洩露出去，不會造成之前通訊時使用的會話密鑰(session key)的洩露，也就不會暴漏以前的通訊內容。簡單的說，當丟失了這個 long-term key 之後，以後的行為的安全性無法保證，但是之前的行為是保證安全的。

Perfect 的意義是包含了無條件安全的性質，大部分的 forward secrecy 方案是無法達到 Perfect 的。而 forward security 的保證的是：攻擊者獲取到了當前的金鑰，但是也無法成功偽造一個過去的簽名。

以非對稱加密系統而言，滿足 Forward secrecy 或者 forward security 的公開金鑰環境下的(簽名、金鑰交換或加密)方案，其公開金鑰是固定的，而金鑰則隨著時間進行更新。這個更新過程是單向的，因此也就保證了拿到當前的金鑰，是無法恢復出以前的金鑰，從而保證了「前向安全」。

弱完美前向安全

弱完美前向安全(Weak perfect forward secrecy)是較弱的屬性, 當代理人

的長期金鑰洩露，先前建立的工作階段金鑰的保密性會被保證，但這只對惡意攻擊方沒有活躍干擾的 session 而言。Hugo Krawczyk 在 2005 年提出了這個新概念，這之間的區別和轉發保密。這種較弱的定義隱式地要求完美前向安全保持先前建立工作階段金鑰的安全，即使在這個會話中惡意攻擊方做活躍的干擾攻擊，或試圖充當中間人。

簡單的說，這兩個概念是用在不同的環境中，但是其目的是一樣的：保證金鑰丟失之前的訊息安全性或簽名的不可偽造性。一般而言，滿足 forward security 的公開金鑰環境下的簽名、金鑰交換或加密方案，其公開金鑰是固定的，而金鑰則隨著時間進行更新。這個更新過程是單向的，因此也就保證了拿到當前的金鑰，是無法恢復出以前的金鑰，從而保證了前向安全，如圖 2.1 所示。與之相對應的還有「後向安全(backward secrecy 或 security)」的概念。

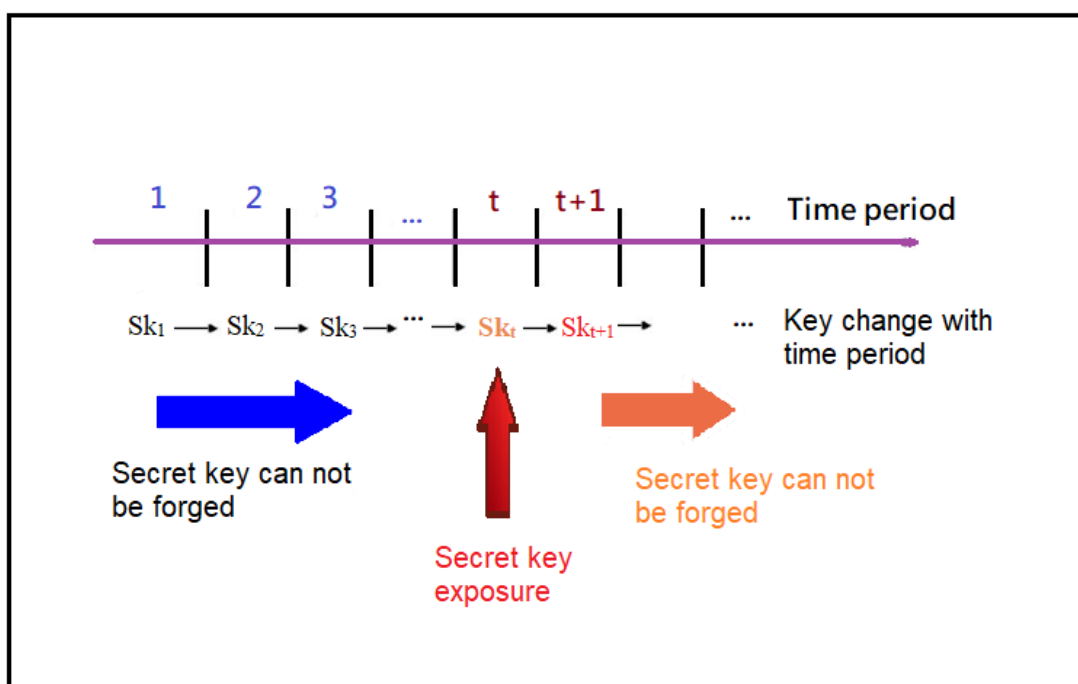


圖 2.1 前向安全示意圖

2.2 前向安全性在日誌文件上的加密研究

在世界上許多實際的應用中，敏感性/機密性的資訊都必須被保存在不可信任機器的日誌文件中。在一個攻擊者捕獲這台機器的事件中，我們希望能夠保證他從日誌文件中將會幾乎無法取得任何資訊，並限制他竄改日誌文件的能力。

在設備的擁有者與設備內的機密資訊擁有者不是同一個人的系統中，至關重要的是，要有稽核機制來確認是否存在某種企圖欺詐行為，而且這些稽核機制必須能夠在攻擊者無法控制的情況下倖免於難。用稽核機制來適當決定是否曾有一些企圖詐騙是有必要的，這些稽核機制必須在攻擊者無法預期的操控嘗試中還能存活著。這並非是一個系統為了能夠避免所有可能的日誌文件的操控，而是一個系統為了在事件發生之後能夠檢測並查證此類的操控。

2.2.1 介紹



在多數的情況下，最常發生的就是入侵者在登入設備取得控制權後，在某些特定目錄中植入惡意檔案/軟體，進而進行接下來的行為。而在入侵者真正成功取得設備控制權之前，也會有許多嘗試攻擊的行為和動作。通常在嵌入式系統裝置中，有發生使用者登入以及出現系統異常事件時，會將這些內容記錄在日誌文件中，包含入侵者的動作，都會被記錄下來。但是入侵者不但不會讓系統管理者發現他的入侵動作，也不希望在系統內留下所有相關的足跡和證據，所以會想辦法刪除或是修改事件發生當時的日誌文件內容。因此，若要達到能夠保證內容的真實性，事件發生之前的所有的日誌文件內容的保全方法就成了一個非常重要的工作。

因此，Bruce Schneier 和 John Kelsey 於在 ACM Transactions on Information and System Security (TISSEC) 中提出了有關於日誌文件的前向安全性加密研究，其目的就在於將過去所有的日誌項目內容都能夠以某種方式加以保全，不會因為攻擊者取得設備的控制權而遭到竄改使得以假亂真[8][9]。其理論為一個計算成本簡便的方法，來使該攻擊者無法來讀取在登錄裝置的入侵之前產生的所有日誌項目，並也不可能在未經系統管理者發覺的情況下修改或是摧毀日誌。

2.2.2 情境

其設定情境如圖 2.2 所示，我們有一個不可信任的機器，U，它沒有實體安全或是足夠的防竄改機制來保證它不會被一些攻擊者所接管。然而，這個機器需要能夠建立和維護一組檔案，這檔案包含了一些關於程序、測量、事件或是任務工作的稽核日誌項目。在攻擊者獲得對 U 的控制之後，沒有任何安全措施可以保護已寫入的稽核 Log 項目。並且在此之後，無論攻擊者希望如何寫入，U 都會如實寫入 Log 項目中，例如變造或消除相關入侵事件紀錄。

另外，我們還有一台具有高度的安全性，並預設不會被攻擊者入侵竄改或竊取資料的可信賴機器，T。設備 T 透過可信賴的網路傳輸與不可信任機器 U 作連結。

我們假設 U 既有短期儲存儲存器，也有長期儲存器。長期儲存器將儲存稽核日誌，而且我們假設它足夠大，填滿日誌不會有空間問題。另外，我們假設 U 能夠不可挽回地刪除短期儲存器中保存的訊息，並且每當新的密鑰被導出時就會這樣做。我們也假定 U 有一些產生隨機或密碼強偽隨機值的方法。最後，我們假設存在幾個密碼學原語，並且透過一個很好理解的方式來建立跨不安全介質的安全連接。所有這些方法在其他文獻都有詳細的描述：請參見[10][11][12]。

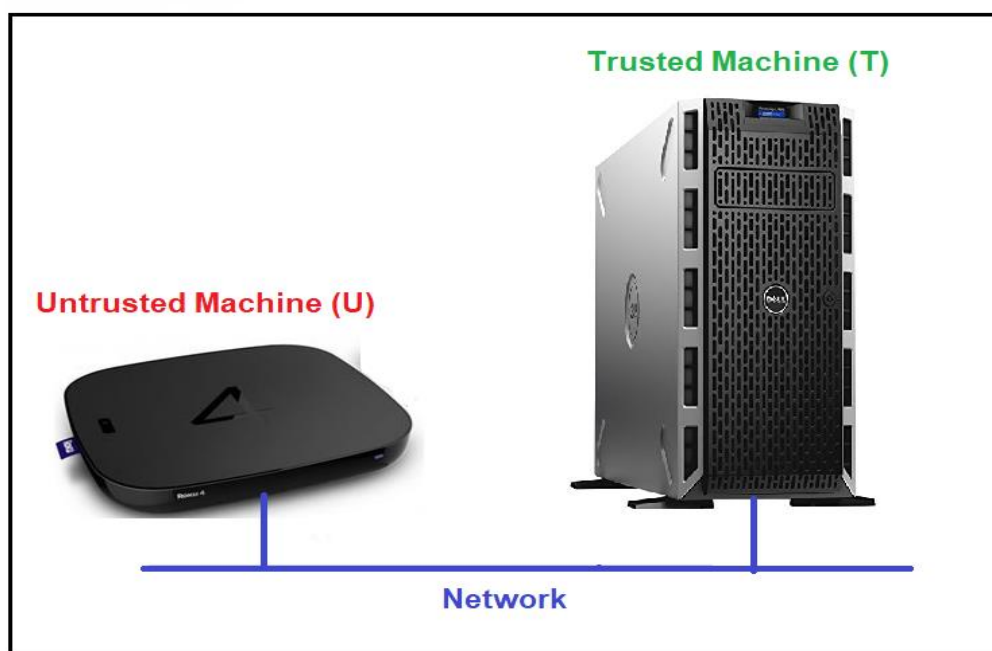


圖 2.2 日誌安全演算法硬體架構圖

2.2.3 解決方案

前面提到，在攻擊者獲得對 U 的控制之後，沒有任何安全措施可以保護稽核日誌項目。在這一點上，無論攻擊者想寫什麼， U 都會寫入日誌。其中一個解決方案是，如果 T 和 U 之間有一條可靠的網路通道，那麼這個問題就不會出現。 U 將簡單地加密每個日誌項目當它創建時，並通過這個通道發送給 T 。一旦日誌被 T 所儲存，儘管攻擊者有意在被入侵的裝置 U 上做某種程度上“重寫歷史記錄”，但在此稽核機制下，仍然可以檢測出相關的行為與操控。我們就能在某種程度上相信日誌的內容正確性。

另外，我們希望能夠達到對 U 上登錄的日誌項目真實性提供最強的安全保證。特別是，我們不希望在時間 t 獲得對 U 的控制權的攻擊者成為能夠讀取在時間 t 之前所創建的日誌項目，並且我們不希望他能夠在時間 t 之前更改或刪除所做的日誌項目，以至於使得當 U 與 T 互動交談時無法檢測到攻擊者對日誌項目的篡改操作。

因此，更可靠的做法就是拒絕攻擊者能夠讀取或修改在他入侵登入設備之前的日誌項目。即使 U 受到攻擊，我們也必須能夠對裝置 U 先前已生成的日誌項目的安全性做出強有力的保證，此種方式就是利用所謂的前向安全性 (Forward Security) 來達成。

經由具有前向安全性的日誌項目，加上 U 的透過可靠的網路連結與 T 作出最小互動，即可對不可信任機器上的所有日誌文件作出相當高程度的內容安全確認性，並且也可以達到事後稽核驗證的功能。

2.2.4 方法論

我們的系統利用了這樣一個事實，即創建日誌文件的不可信機器一開始與受信任的驗證機器共享一個密鑰，並且用這個密鑰，來創建日誌文件。

應用此方法的日誌文件安全性，來自以下四個基本事實：

- 在寫入日誌項目後，立即使用單向雜湊函數對日誌的認證密鑰進行雜湊處理。驗證密鑰的新值會覆蓋並不可挽回地刪除以前的值。
- 每個日誌條目的加密密鑰都是使用單向過程從該項目的身份驗證密鑰導出的。這樣就可以提供單個日誌的加密密鑰給部分信任的用戶或實體（以便他們可以解密和讀取項目），而不允許這些用戶或實體進行無法檢測的更改。
- 每個日誌項目在雜湊鏈中包含一個元素，用於驗證所有以前的日誌條目的值 [13][14]。這個值實際上是經過驗證的，這使得透過驗證單一雜湊值來遠程驗證所有歷史的日誌項目這件事成為可能。
- 每個日誌項目都包含它自己的權限遮罩。該權限遮罩定義了部分可信用戶可以訪問哪些日誌項目。不同的部分可信用戶可以被授予對不同種類的條目的訪問權限。由於每個日誌條目的加密密鑰部分來自日誌項目類型，因此說明給定日誌項目具有的權限可確保部分受信任的用戶根本不會獲得正確的密鑰。

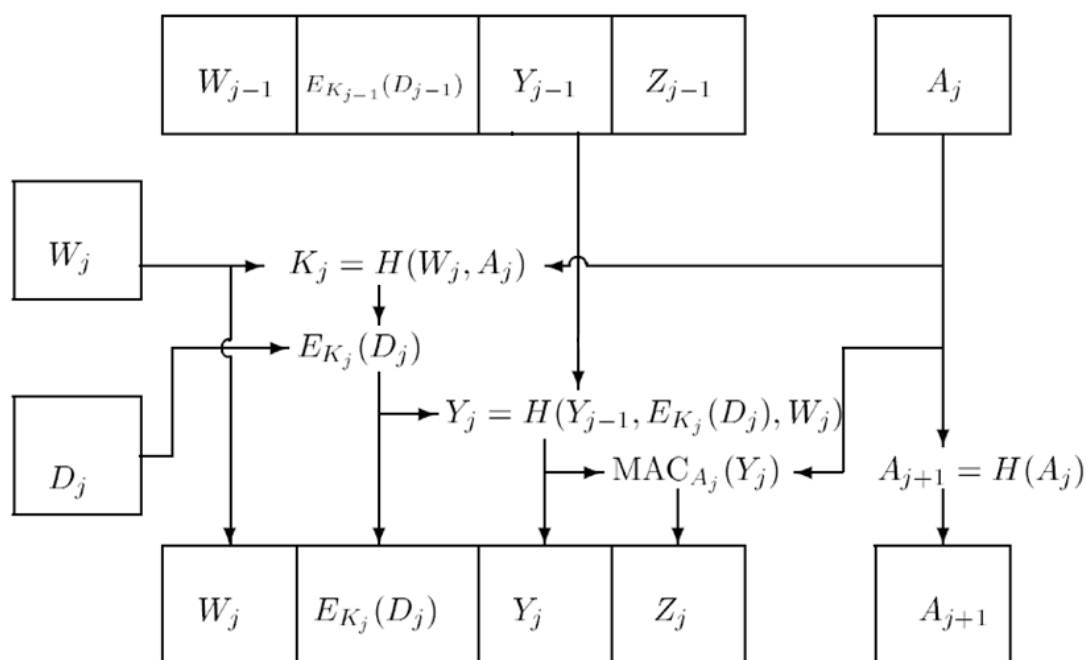


圖 2.3 新增項目至日誌檔案[9]



日誌文件中的所有項目都使用相同的格式，並按照以下過程構建，並在圖2.3中，完整顯示出了這個過程的例子。

- (1) D_j 是要在 ID_{log} 的第 j 個日誌項目中放入的資料。

它僅能是日誌條目的讀者將可以明確理解的某段文字，意即明文。 D_j 是要在 ID_{log} 的第 j 個日誌項目中輸入的數據。 D 的具體數據格式在這個方案中沒有規定：它僅能是日誌項目的讀取者將可以明確理解的某個東西，並且幾乎在所有情況下都可以與隨機亂序區分開來。

- (2) W_j 是第 j 個日誌項目的項目類型。

W_j 是第 j 個日誌項目的項目類型。此類型用作 V 的權限遮罩； T 將允許控制哪些日誌項目類型可以被任何特定的 V 訪問。

(3) A_j 是給日誌中第 j 個項目的身份驗證密鑰。

這是提供這個解決方案所有安全性的核心秘密。

請注意，在啟動日誌文件之前，U 必須生成新的 A_0 ； A_0 可以在啟動時由 T 傳送給 U，或者 U 可以隨機生成它，然後將它安全地傳送給 T。

(4) $K_j = \text{hash}(\text{"Encryption Key"}, W_j, A_j)$

這是用來加密日誌中第 j 個項目的密鑰。

請注意， W_j 用於密鑰派生，以防止部分可信任的驗證者 V 得到他被不允許存取的日誌項目類型的解密密鑰。

(5) $Y_j = \text{hash}(Y_{j-1}, E_{K_j}(D_j), W_j)$

這是我們維護的雜湊鏈，用來允許部分可信任用戶 Vs，能夠通過與可信機器 T 的低頻寬網路連接來驗證日誌的部分內容。 Y_j 是基於 $E_{K_j}(D_j)$ 而不是 D_j ，因此可以在不知道日誌內容的情況下來驗證雜湊鏈。在啟動時， Y_1 被定義為 20 個二進制零的字節塊。

(6) $Z_j = \text{MAC}_{A_j}(Y_j)$

以 A_j 為金鑰， Z_j 為 Y_j 的訊息認證碼。

(7) $L_j = W_j, E_{K_j}(D_j), Y_j, Z_j$

其中 L_j 是第 j 個日誌項目

(8) $A_{j+1} = \text{hash}(\text{"Increment Hash"}, A_j)$

將密鑰 A_j 進行單向函數作雜湊處理，取得下一個 session 密鑰 A_{j+1}

請注意，當計算完 A_{j+1} 和 K_j 時，先前的 A_j 和 K_{j-1} 值將被無法挽回地破壞；在正常操作下，U 上沒有這些值的複本。此外，在步驟 (4) 中使用後 K_j 會立即被銷毀。（當然，攻擊者在控制 U 後也有可能會儲存 A_j 值。）

上述過程定義了如何將第 j 個項目寫入日誌中，給定 A_{j+1} ， Y_{j-1} 和 D_j 。

圖2.4則是說明，如果攻擊者在時間 t 獲得了 U 的控制權，他將會得到一個有效的日誌項目列表, L_1, L_2, \dots, L_t 和 A_{t+1} 的值。他不能計算任何 $n \leq t$ 的 A_{t-n} ，所以他無法讀取或偽造任何過去的項目。 他可以刪除一個項目區塊（或整個日誌檔案），但他不能創建新的日誌項目來替換他已經刪除掉的項目，無論他是刪除了日誌中間的一個項目區塊或是刪除了日誌結尾的一個項目區塊。當下次 U 與 T 互動時， T 會意識到項目已經從日誌中被刪除了，並且（1） U 可能提交了一些未經適當審核的無效操作，以及（2） U 可能已經提交了一些已經被刪除的有效動作的審核記錄。

如果攻擊者在步驟（8）之前獲得對 U 的控制，他可以學習到 A_t 。 在這種情況下，第 t 條日誌項目就會有被刪除或操縱的危險。即使攻擊者沒有刪除項目，審核系統也應該包含指向被攻擊者成功入侵紀錄的日誌項目。同樣的，由於攻擊者無法讀取過去的項目，他將無法知道他的入侵動作是否被日誌系統記錄下來。

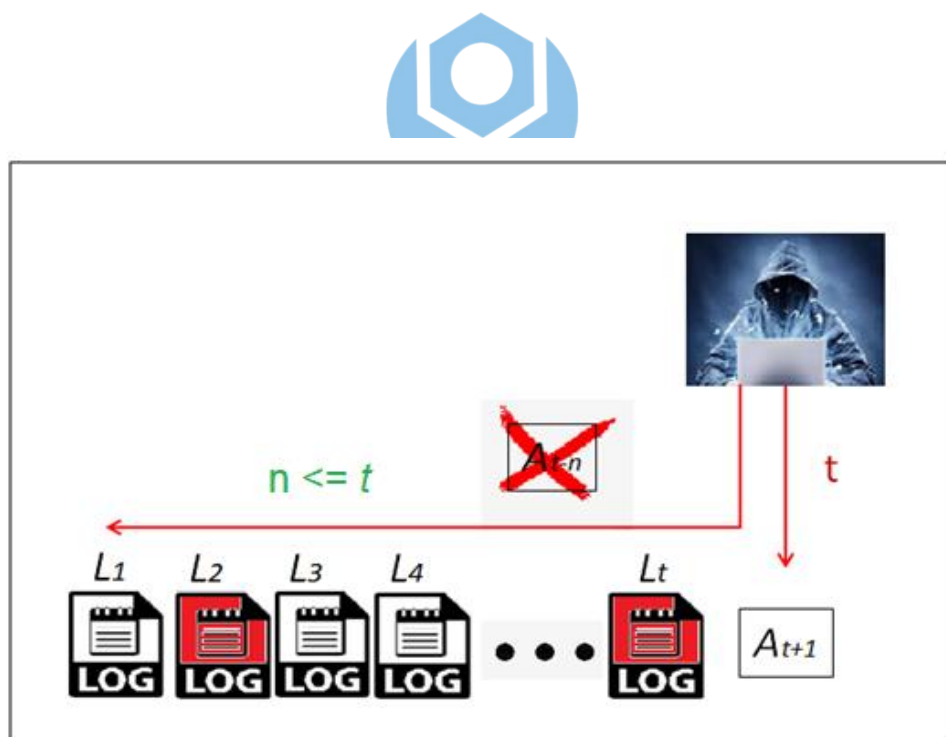


圖 2.4 入侵時間與日誌項目安全

2.3 前向安全序列聚合

2.3.1 介紹與情境

在不安全環境中，IoT 感測器裝置可能會被入侵，並且可能會對其入侵後的資料進行操控。一個重要的問題除了是上面所提到的前向安全性 – 如何確保可能被入侵的資料無法被操控之外，由於典型的感測器在儲存和通信設施方面有限制，因此另一個問題就是如何地減少由於累積資料所造成的資源消耗。

感測器可以在許多不同的設置、場景和應用中實現大規模的資料收集。在民用和軍事領域的各種追蹤和監測應用中都有許多例子。以無線感測器網路(WSN)來說，可能包含數百或數千個低成本感測器和一個或多個接收器或資料收集器。單個感測器從環境中獲取量測結果，並且（定期或根據請求）將累計的資料轉發到接收器。接收器可能是另一個網路的閘道器、強大的資料處理或儲存中心、或人機界面存取點的入口。（有些 WSN 經由接收器支援使用者驅動的資料查詢和命令。）



本節提供了兩種假設的感測器環境的情境，並使用術語“收集器(collector)”和“接收器(sink)”來區分在這兩種情況下收集資料的實體裝置：

A. 感測器之間不進行通信，即沒有感測器網路。反之，有一種稱之為收集器的行動設備，它的功能只不過是感測器和離線（可信）接收器之間的中介，此收集器可能不能完全被信任。

B. 感測器可以通信，但實際上並沒有“網路化”，意即通信限於僅將資訊從其他感測器轉發到接收器或多個接收器。在這種情況下，接收器是完全可信的實體。

無論哪種情況，感測器都可能無法隨意與接收器通信。相反的，它收集資料並等待（可能很長一段時間）或者對於訊號 – 或者某個預定時間 – 將累積資料上傳到收集器或接收器。換句話說，沒有感測器與收集器或接收器之間的感測訊息的即時報告。

資料完整性和(感測器)認證是大多數感測器應用所需的基本安全服務[15]，

因為感測器通常用於無人值守和危險環境。它們與物理環境和操作人員密切相互作用，因此面臨著廣泛的安全風險。攻擊者可以注入自己的資料以及修改和刪除感測器所產生的資料。因此，感測器的資料在處理之前必須經過認證，才能用於任何目的。特別是在關鍵環境中（例如輻射，地震或入侵監測），需要強大的資料完整性和真實性保證。諸如 MAC（訊息認證碼）或數位簽名之類的標準教科書技術可以用於需要資料完整性/真實性的應用中。但是，有幾個障礙阻礙了這些標準技術的直接使用。

其中一個重要的問題是感測器被入侵的威脅，以及被入侵後隨之而來的，用於 MAC 或簽名的密鑰將會面臨曝露的危險。密鑰曝光會使得攻擊者可以輕鬆地在入侵後產生出感測器的欺詐資料，甚至是發生在尚未匯報給接收器或收集者的時候，這件事顯然是不能夠被允許發生的。幸運的是，現今存在著所謂的前向安全加密技術，其允許簽名者（在這種案例下指的就是感測器）週期性地演變其秘密金鑰，從而入侵了當前秘密金鑰也不會影響在過去時段中使用的秘密金鑰。因此可以經由使用簽名方法來減輕感測器被入侵的影響。換句話說，感測器不會等待直到必須發送所有產生的感應資料時，才會製作簽名（或 MAC），因為這樣做會被入侵者打開所有的被收集資料進行攻擊。相反的，它只要感測到資料就簽署數據並演變簽名密鑰。

另一個重要問題是儲存和通訊的成本開銷。顯然，在大多數感測器設置中，內建的儲存媒體是一種有限資源的元件，設計者很自然地會最小化其尺寸和消耗。在上面概述的情景 A 和情景 B 中，感測器逐漸積累資料（讀數，量測資料），將其儲存在本機中，並在稍後時間將其發送到接收器。我們在這裡不關心如何最小化實際資料所消耗的儲存空間，這本身是另一個研究主題。相反的，我們感興趣的是基於身份驗證標籤（即 MACs 或簽名）而最小化的儲存空間，因為它們代表了單純成本(pure overhead)。前向安全性技術迫使感測器計算每個資料感測單元的認證標籤，但是當感測器在等待一段時間或等待某一個訊號後才要卸載資料時，可能會累積了與感測訊息一樣多的認證標籤信號，這就會造成儲存上的問題。因 MAC（或是簽名）的大小可能容易超過實際(訊息)數據的大小。根據相關文獻研究，每個 MAC 有 128 位元或每個簽名至少有 160 個位元需要分配。

此外，資料通訊的開銷成本雖然可能不是關鍵的問題，但卻是一個與感測器資料傳輸相關的問題。在場景A中，感測器直接將收集的訊息上傳到收集器，因此，由於發送多個認證標籤而導致的通訊開銷比情況B中的問題少，其中，相同

的開銷影響了從其他感測器向接收器轉發訊息的所有感測器。(我們參考文獻[16]中經常引用的民間文章，聲稱單一位元的無線傳輸可消耗單一32位元計算能量的1,000倍以上。)

2.3.2 解決方案

為了能夠使儲存(和通訊)開銷最小化的需求與減少潛在的關鍵入侵(即獲得前向安全性)的需求能夠同時一致達成，Di Ma 和 Gene Tsudik 提出了前向安全序列聚合(*FSSAgg, Forward-Secure Sequential Aggregate*)認證方案的概念[17]。在公鑰密碼學的背景中考慮了 *FssAgg* 認證方案，並構建 *FssAgg* MAC 方案和 *FssAgg* signature 方案，每種方案適用於不同的假設情境。

此前向安全循序聚合(*FSSAgg*)身份驗證方案，除了可同時減輕密鑰洩露的威脅，並可實現最佳儲存和通訊效率。*FssAgg* 身份驗證方案允許簽名者將在不同密鑰/時間段中生成的多個身份驗證標籤組合為一個單一固定長度的標籤。就算當前密鑰的洩漏，也不會允許攻擊者偽造任何包含預先約定洩漏的元素的聚合身份驗證標記。任何插入新訊息，修改和刪除(包括截斷)現有訊息都會使聚合標記明顯的失效。

一般的聚合簽名方案是將由 n 個簽名者生成的 k 個簽名 ($k > n$) 組合成單一且緊密的聚合簽名，如果經過驗證，則同時驗證了每個元件的簽名。不同的是，此研究的目標是經由相同的簽名者(例如同一個感測器)來聚合簽名，但是，這些簽名是在不同的時期和以不同的密鑰來計算出來的。因此，我們的目標對聚合簽名的現有定義不施加額外的限制。此外，設想的方案不需要多個簽名的同時聚合，如[18]；相反的，我們需要如[19]或[20]中的順序(增量)聚合。

2.3.3 定義與屬性

在本節中，將介紹一些非正式的定義和屬性。*FssAgg* 簽名方案由接下來的演算法組成。它們與連續聚合簽名方案中的那些非常相似，特別是最近的 Lu 等人的方案[20]。

密鑰生成演算法 *FssAgg.Kg* 用於生成公鑰/私鑰配對。與[20]中使用的不


同，它也將 T 作為輸入 - 時間週期的最大數量（密鑰演變）。

符號與聚合演算法 $FssAgg.Asig$ 將私鑰、要被簽名的訊息和迄今為止的簽名（到此為止所計算的聚合簽名）作為輸入。它在輸入的訊息上計算一個新簽名，並將其與迄今為止的簽名合併，來產生一個新的聚合簽名。作為 $FssAgg.Asig$ 的最後一步，它會運行一個密鑰更新子程序 $FssAgg.Upd$ ，它將當前週期的簽名密鑰作為輸入，並回傳下一個週期（不超過 T ）的新簽名密鑰。讓密鑰更新部分的簽名和聚合演算法，以獲得更強大的安全保證（見下文）。

驗證演算法 $FssAgg.Aver$ ，在輸入假定的聚合簽名、一組可能已簽名的不同訊息和一個公鑰時，驗證輸出聚合是否有效。（與非前向安全方案的區別在於，這裡只使用單個公鑰，因為只有一個簽名者。）

密鑰更新算法 $FssAgg.Upd$ 將當前期間的簽名密鑰作為輸入，並回傳下一個期間的新簽名密鑰（假如當前期限不超過 $T-1$ ）。

一個安全的 $FssAgg$ 方案必須滿足以下屬性：

- 
1. **正確性**：使用 $FssAgg.Asig$ 生成的任何聚合簽名都必須被 $FssAgg.Aver$ 所接受。
 2. **不可偽造性**：在不知道任何簽名密鑰（對於任何時期）的情況下，則任何對手都不能計算任何訊息或訊息集合上的聚合簽名。
 3. **前向安全性**：對於任何 $j < i$ 期間，沒有任何一個入侵了簽名者的第 i 個簽名密鑰的攻擊者可以生成包含簽名訊息的有效聚合簽名，除了簽名者在入侵之前生成的迄今聚合的簽名之外，即對手在入侵後發現的聚合簽名。

請注意，最後一個屬性包含了針對截斷或刪除攻擊的安全性在內。對一個入侵簽名者的攻擊者來說，只有兩種選擇：不是在未來聚合的簽名中包含完整的迄今聚合的簽名，不然就是完全忽略迄今聚合的簽名，並開始一個全新的聚合簽名，它無法做的是選擇性地刪除已經生成的聚合簽名的元件。

2.3.4 FssAgg MAC 方案

在此節提出了一個簡單的FssAgg MAC方案。當不需要公共(可轉移)驗證時，它可用於驗證多個訊息。因此，它非常適合第1部分中介紹的場景B，其中的感測器與接收器進行通訊(透過其它感測器)。首先介紹該方案，然後展示如何將其應用於設想的感測器環境。

該方案使用以下密碼學原語：

H ：具有限制為 k 位字符串的域的抗碰撞單向雜湊函數 $H: \{0, 1\}^k \rightarrow \{0, 1\}^k$

H_a ：具有任意長度輸入的抗碰撞單向雜湊函數： $H_a: \{0, 1\}^* \rightarrow \{0, 1\}^k$

h ：一個安全MAC方案 $h: \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^t$ ，在輸入 k -bit 的密鑰 x 和任意訊息 m 時，輸出一個 t -bit 的MAC $h_x(m)$ 。



FssAgg.Kg. 可以使用任何對稱密鑰生成演算法來生成初始 k -bit 的密鑰 s 。並設定 $sk_0 = vk = s$

FssAgg.Asig. 在時間段 i ，簽名者被給予要簽名的訊息 M_i 和在訊息 M_1, \dots, M_{i-1} 上的聚合到目前為止的MAC $\sigma_{1,i-1}$ 。簽名者首先使用 sk_i 生成一個帶有 h 的MAC σ_i ： $\sigma_i = h_{sk_i}(M_i)$ 。然後通過將 σ_i 折疊到 $\sigma_{1,i-1}$ 到 H_a ： $\sigma_{1,i} = H_a(\sigma_{1,i-1} || \sigma_i)$ 來計算 $\sigma_{1,i}$ 。 H_a 扮演的角色為聚合函數。或者我們可以如下計算 $\sigma_{1,i}$

$$\sigma_{1,i} = H_a(H_a(\dots H_a(H_a(\sigma_1 || \sigma_2) || \sigma_3) || \dots) || \sigma_i) \text{ where } \sigma_j = h_{sk_j}(M_j) \forall j = 1, \dots, i \quad (1)$$

最後，簽名者執行定義如下的密鑰更新子程序：

FssAgg.Upd. 我們將第 i 個簽名密鑰 sk_i 定義為前一個密鑰 sk_{i-1} 的 H 下面的圖像：

$$sk_i = H(sk_{i-1}), i > 0 \quad (\text{這部分與[19]中的前向安全MAC方案相同。})$$

為了驗證透過訊息 M_1, \dots, M_i 所產生的候選 $\sigma_{1,i}$ ，驗證者(擁有與初始簽名

密鑰 sk_0 相同的驗證密鑰 vk) 通過公鑰更新功能來計算密鑰 sk_1, \dots, sk_i 。然後它模仿簽名過程並重新計算 $\sigma_{1,i}$ 並將其與 $\sigma_{1,i}$ 進行比較。如果兩個值匹配，則輸出有效，否則輸出無效。

2.3.5 FssAgg Signature 方案

如果需要公共（可轉移）驗證，我們需要一個FssAgg簽名方案來檢查資料記錄的真實性。簡單地說，如果我們將簽名者 i 的密鑰視為在時間段 i 中使用的密鑰（由相同的簽名者），則所有聚合簽名方案[18][19][20]可以用作FssAgg簽名方案。然而，由於簽名者（例如感測器）將會需要 $O(T)$ 的儲存空間來儲存其密鑰，因此簡單的構造對於此方案目的是無用的。

FssAgg簽名方案的總體效率取決於以下度量：1) 聚合簽名的尺寸；2) 簽名密鑰的尺寸；3) 密鑰更新的複雜性；4) 聚合簽名的複雜性；5) 驗證密鑰的尺寸；6) 聚合驗證的複雜性。前四個表示簽名者效率，後兩個表示驗證者效率；大小參數（聚合簽名、簽名密鑰和驗證密鑰）表示空間效率，複雜性參數（簽名、驗證和密鑰更新）表示時間效率。在我們假設的感測器場景中，簽名者效率比驗證者效率更重要，而空間效率比時間效率更重要。

著眼於簽名者和空間效率，我們提出了一種基於BLS簽名方案的FssAgg簽名方案[18]。BLS簽名可以透過任何的EC乘法來被聚合[21]。我們首先介紹BLS方案，然後展示如何將其修改為FssAgg簽名方案。

BLS方案以與雙線性映射成組實現。雙線性映射為一個映射 $e: G_1 \times G_2 \rightarrow G_T$ ，其中：(a) G_1 和 G_2 是質數階 q 的兩個（乘法）循環群；(b) $|G_1| = |G_2| = |G_T|$ ；(c) g_1 是 G_1 的生成子， g_2 是 G_2 的生成子。雙線性映射 $e: G_1 \times G_2 \rightarrow G_T$ 滿足以下屬性：

1. Bilinear: for all $x \in G_1, y \in G_2$ and $a, b \in \mathbb{Z}$, $e(x^a, y^b) = e(x, y)^{ab}$;
2. Non-degenerate: $e(g_1, g_2) \neq 1$

BLS方案使用全域雜湊函數 $H_I(\cdot): \{0, 1\}^* \rightarrow G_1$ 。密鑰的產生程序包含了為每個簽名者隨機選擇一個 $x \in \mathbb{Z}_q$ ，並計算 $v = g_2^x$ 。簽名者的公鑰是 $v \in$

G_2 ，她的密鑰是 x 。對訊息 M 進行的簽名程序則涉及了計算訊息雜湊 $h = H_1(M)$ ，然後簽名 $\sigma = h^x$ 。為了驗證簽名，計算 $\sigma = h^x$ 並檢查 $e(\sigma, g_2) = e(h, v)$ 。驗證成本為2個雙線性映射。

為了聚合 n 個BLS簽名，可以按如下公式計算各個簽名的乘積：

$$\sigma_{1,n} = \prod_{i=1}^n \sigma_i \quad (2)$$

其中 σ_i 對應於訊息 M_i 上的簽名。聚合簽名 $\sigma_{1,n}$ 與單獨的BLS簽名具有相同的大小，而且聚合的動作可以由任何人遞增地執行。

聚合BLS簽名 $\sigma_{1,n}$ 的驗證，包括了計算所有訊息雜湊的乘積並驗證以下匹配：

$$e(\sigma_{1,n}) \stackrel{?}{=} \prod_{i=1}^n e(h_i, v_i) \quad (3)$$

其中 v_i 是在訊息 M_i 上生成 σ_i 的簽名者的公鑰。

FssAgg.Kg 簽名者隨機選擇一個 $x \in \mathbf{Z}_p$ 並計算一對 (x_i, v_i) ($i = 1, \dots, T$) 為：

$$x_i = H(x_{i-1}), v_i = g^{x_i}_2$$

初始簽名密鑰是 x_0 ，公鑰是： $(v_1, \dots, v_T) = (g^{x_1}_2, \dots, g^{x_T}_2)$ 。

請注意，在這裡的感測器情境中，感測器（簽名者）不會生成自己的密鑰。相反的，接收器（或其他一些可信方）將為所有感測器生成所有公鑰和密鑰。但是，收集器只能獲得公鑰。

FssAgg.Asig 對於要被簽名的訊息 M_i 的輸入而言，在訊息 M_1, \dots, M_{i-1} 上的迄今聚合簽名 $\sigma_{1,i-1}$ 和當前簽名密鑰 x_i ，簽名者首先使用 x_i 來計算在 M_i 上的BLS簽名： $\sigma_i = H^{x_i}(index \parallel M_i)$ 其中 $index$ 表示 M_i 在儲存器中的位置。此索引的目的是提供訊息排序，因為原始BGLS聚合函數不會對聚合元件施加任何順序。

接下來，簽名者透過乘法將 σ_i 聚合到 $\sigma_{1,i}$ ： $\sigma_{1,i} = \sigma_{1,i-1} \cdot \sigma_i$ 。最後，簽名者更新密鑰。

FssAgg.Upd 簽名者通過散列函數 H 演變其秘密簽名密鑰： $x_i = H(x_{i-1})$ 。

FssAgg.Aver 驗證者使用公式3和公鑰 pk 來驗證聚合簽名 $\sigma_{1,i}$

這個FssAgg簽名方案的安全性基於底層的BLS方案，不需要其他假設。以下定理總結了此FssAgg簽名方案的安全性，並且是前瞻性的證明。

Theorem 1. *If BLS is a $(t', q_H, q'_S, \epsilon)$ -secure signature scheme, our construction above is a $(t, q_H, q_S, T, \epsilon)$ -secure FssAgg signature scheme where $t' = t + O(q_H + q_S)$, $\epsilon' = \epsilon/T$, and $q'_S = q_S/T$.*

[17]

2.4 關於系統日誌保全的其它研究

由於系統日誌的保全是裝置安全機制中非常重要的一環，因此近年來都有出現許多相關研究。例如在2017年由 Sepideh Avizheh 等人提出的關於智慧家庭 (Smart Home) 的系統日誌安全研究[22]，除了使用前向安全性的機制來將系統日誌加密產生三種簽名(單項簽名，錨簽名 和 FssAgg簽名)之外，並以一個用於管理智慧家庭（或類似環境）的資料通信，儲存，處理和管理的通用基於主機 (HOST-BASED) 的計算框架，來控管整個家庭中的所有IoT網路裝置，並會將簽名加密過的系統日誌，上傳至雲端的區塊鏈(block chain)作為安全保存及驗證。

另外還有Vishal Karande 等人所提出的 SGX-Log 機制[23]，它是一種全新的日誌記錄系統，可確保日誌數據的完整性和機密性。該研究是透過利用稱為 Intel SGX的最新硬體擴展功能來重新設計日誌記錄系統，此安全機制架構類似於伺服器系統中的可信賴平台模組(Trusted Platform Module)的功能，該系統提供了一個安全區域，其中包含密封和解封基元，以保護儲存器和磁碟中的程式代碼和數據不被未授權的方式修改，即使是高特權碼也是無法越權存取。

2.5 inotify 檔案異動偵測技術

Linux 核心系統開發者之一的 Rober Love 在 Linux Journal 網站上曾經提到，某些時候，了解 Linux 操作系統的變化是很重要的。系統所使用的用途通常包括必須在看到後立即處理的高優先級數據。查找和處理新文件數據的傳統方法是通常使用 cron 進行 polling。這個效率是非常低的，如果呼叫 polling 的事件過於頻繁，就會無法合理地提升系統性能 0。

Linux 使用了一種有效的方法可以在 user space process 對被關注的檔案文件提供警示功能。這個稱為 inotify Linux 系統呼叫首次是在 Linux Journal 的 2005 年文章中被討論，Robert Love 主要從 C 的角度闡述了新的特性之行為。

inotify 是 Linux 核心子系統之一，做為檔案系統的附加功能，它可監控檔案系統並將異動通知應用程式。本系統的出現取代了舊有 Linux 核心裡，擁有類似功能之 dnotify 模組。原始開發者為 John McCutchan、Rober Love 與 Amy Griffiths。於 Linux 核心 2.6.13 發行時(2005 年六月十八日)，被正式納入 Linux 核心。儘管如此，它仍可透過修補程式的方式與 2.6.12 甚至更早期的 Linux 核心整合。

inotify 的主要應用於以針對有變動的檔案重新索引，而不必沒有效率地每隔幾分鐘就要掃描整個檔案系統。相較於主動輪詢(polling)檔案系統，透過作業系統主動告知檔案異動的方式，讓程式撰寫者所開發的軟體甚至可以在檔案更動後一秒內更新索引。此外，諸如：更新目錄檢視、重新載入設定檔、追蹤變更、備份、同步甚至上傳等許多自動化作業流程，都可因而受惠。這對嵌入式系統來說是十分重要的功能，因為它可以簡化所有監測的動作並且節省裝置的資源

相較於被 inotify 取代較舊的 dnotify 模組，inotify 有諸多益處。在舊的模組中，程式必須為每一個被監控的目錄建立 file descriptor，這種作法很容易讓行程擁有的 file descriptor 逼近系統允許的上限，進而形成瓶頸。dnotify 產生的 file decriptor 也會導致系統資源忙碌，使可移除裝置無法被移除，徒增使用上的困擾。

由於 dnotify 只能讓程式設計師監控目錄層級的變化，「精細度」亦是「dnotify」的劣勢之一。為此，程式設計師必須付出額外的心力，自行撰寫程式碼以期追蹤更細微的檔案系統事件。

相較之下，inotify 使用較少的 file descriptor，亦允許 select()與 poll() 介面，優於 dnotify 使用的訊號系統。這也使得 inotify 與既有以 select()或 poll()為基礎之函式庫(如：Glib)整合更加便利。

inotify 可供應用程式追蹤的事件有：

- IN_ACCESS - 讀取檔案
- IN_MODIFY - 檔案被修改
- IN_ATTRIB - 檔案屬性變更
- IN_OPEN - 檔案被開啟
- IN_CLOSE_WRITE - 被開啟為「可寫入」狀態的檔案遭關閉
- IN_CLOSE_NOWRITE - 被開啟為「非寫入」狀態的檔案遭關閉
- IN_MOVED_FROM and IN_MOVED_TO - 檔案被搬動或更名
- IN_DELETE - 檔案或目錄被刪除
- IN_CREATE - 監控中的目錄下有新檔案產生
- IN_DELETE_SELF - 監控中的檔案遭刪除



第 3 章 研究方法

本研究針對了前述 IoT 裝置的安全弱點，設計了一套機制，除了對 IoT 裝置疑似被植入軟體/檔案的行為偵測提供即時示警，並可同時保證入侵之前所產生日誌檔案的安全。在章節 3.1 中，我們也針對了 Bruce Schneier 和 John Kelsey 的前向安全加密法，提出了一個改良的概念，避免因為在日誌加密的程序未完成前，被入侵者取得密鑰而造成明文洩漏的可能。同時在本章中也描述了此研究所適用的 IoT 裝置設備與條件，以及整體系統功能描述。

3.1 對前向安全日誌加密演算法的改良

在上一章所描述的 Bruce Schneier 和 John Kelsey 加密演算法中，假如共同密鑰 A_j 會在整個加密演算程序完成之前才會被演變成 A_{j+1} 並將 A_j 徹底刪除。倘若入侵者在加密程序完成前(步驟(8))就取得密鑰 A_j ，就算在步驟(4)把 K_j 徹底刪除，甚至是此筆內容已經加密完成，也會因為密鑰洩漏而被入侵者計算出 K_j 來反向解密出 D_j 明文，進而竄改 event message 的內容。

因此本研究提出了雙密鑰(Double-key)的概念，如圖 3.1，步驟詳述如下：

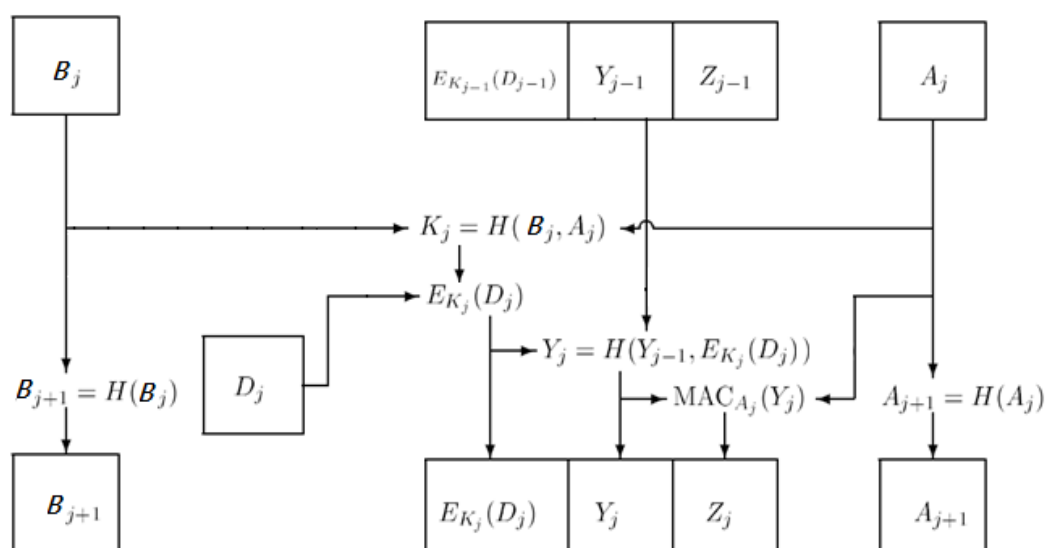


圖 3.1 日誌加密演算法改良

- (1) D_j 是要在 ID_{log} 的第 j 個日誌項目中放入的資料。

D_j 是要在 ID_{log} 的第 j 個日誌項目中輸入的數據，即 Event message 的明文。

- (2) A_j 是給日誌中第 j 個項目的身份驗證密鑰。

如同所有的前向安全加密法，這是與安全伺服器共享的共同密鑰。在啟動日誌文件之前，U 必須生成新的 A_0 ； A_0 可以在啟動時由 T 傳送給 U，或者 U 可以隨機生成它，然後將它安全地傳送給 T。

- (3) B_j 是此演算法中的第二個共同的演化密鑰。

這是本研究所提出的第二個與安全伺服器共享的共同密鑰。如同 A_j ，在啟動日誌文件之前，U 必須生成新的 B_0 ； B_0 可以在啟動時由 T 傳送給 U，或者 U 可以隨機生成它，然後將它安全地傳送給 T。

- (4) $K_j = \text{hash}(\text{"Encryption Key"}, B_j, A_j)$

K_j 是由 B_j 和 A_j 所聯合產生的雜湊值，是用來加密日誌中第 j 個 Event message 明文的密鑰。

- (5) $B_{j+1} = \text{hash}(\text{"Increment Hash"}, B_j)$

將密鑰 B_j 進行單向函數作雜湊處理，取得下一個 session 密鑰 B_{j+1} ，並將 B_j 的值立即刪除。

- (6) $Y_j = \text{hash}(Y_{j-1}, Ek_j(D_j))$

這個雜湊鏈用來允許部分可信任用戶 V，能夠通過與可信機器 T 的低頻寬網路連接來驗證日誌的部分內容。 Y_j 是基於 $Ek_j(D_j)$ 而不是 D_j ，因此可以在不知道日誌內容的情況下來驗證雜湊鏈。

- (7) $Z_j = \text{MAC}_{A_j}(Y_j)$

以 A_j 為金鑰， Z_j 為 Y_j 的訊息認證碼。

(8) $\underline{L_j = E_{K_j}(D_j), Y_j, Z_j}$

其中 L_j 是第 j 個日誌項目

(9) $\underline{A_{j+1} = \text{hash}(\text{"Increment Hash"}, A_j)}$

將密鑰 A_j 進行單向函數作雜湊處理，取得下一個 session 密鑰 A_{j+1}

當計算完 B_{j+1} 、 A_{j+1} 和 K_j 時，先前的 B_j 、 A_j 和 K_{j-1} 值將被無法挽回地破壞；在正常操作下，U 上沒有這些值的複本。

但 B_j 與 A_j 不同的是，在成功產生出 K_j 時， B_j 就會立刻演化成 B_{j+1} 並立即將 B_j 的值立刻刪除。因此，就算入侵者在加密程序未完成前就取得 A_j 的值，他也沒有辦法計算出 K_j 的值，也就沒有辦法反向解密出 D_j 的訊息明文，因此可以更加確保加密程序進行中的安全性。對於入侵者來說，要同時取得兩個共同密鑰的難度，也有可能高於單一共同密鑰的環境，也提升了取得密鑰來解開明文或是偽造下一筆加密訊息的難度。

同時此改良後的加密演算法，也保留了 Y_j 雜湊鏈，使得仍然可以讓部分可信任者 V 用加密過的 $E_{K_j}(D_j)$ 來驗證日誌的正確性，但不會因此洩漏 D_j 的明文訊息，並也可繼續得以透過驗證單一雜湊值來遠程驗證所有的歷史日誌項目，保留了原本演算法的優點。

3.2 IoT 設備特性與容易被攻擊的原因

與筆記型電腦、個人電腦或是手機不同，IoT 設備的特性通常是隨處不在，甚至是隱身在使用者的生活方式中，讓使用者很容易忽略了如資料洩露，拒絕服務等相關風險。

由於極大多數的 IoT 設備與使用者之間的互動很少，且使用者也缺乏相關的資安意識，以致其發生異狀時不易察覺；而製造商忽略了資安防護或是為了成本考量，以致產品本身缺乏安全的加密功能，這些全都是造成 IoT 設備容易被攻擊成功的重要原因。

3.3 入侵的定義和方式

IoT裝置通常廣布在各種環境，且大多數也是經由網路與伺服器做連結和通訊。所以為了達到大規模且有效的成果，攻擊者大多數都是透過網路實施入侵行為而非針對實體裝置攻擊。本研究所針對入侵者的攻擊方式，就是入侵者在登入設備取得控制權後，在某些特定目錄如binary folder中植入惡意檔案/軟體，進而遂行其接下來的行動。

通常嵌入式系統裝置在使用者登入以及發生系統異常事件時，會在日誌檔案項目中紀錄這些內容，包含入侵者的動作，都會被記錄下來。但是通常入侵者不但不會讓系統管理者發現他的入侵動作，也不希望在系統內留下相關的足跡和證據，所以就有可能會想辦法刪除或是修改事件發生當時的日誌項目內容。

因此，本研究的目的之一，就是讓入侵者無法刪除或是修改在入侵之前所產生的所有事件日誌項目，使系統管理者的事後安全稽核的功能更正確和完善。



3.4 IoT 設備的限制條件

市面上的 IoT 裝置的種類非常繁多，功能也各異。例如在軟硬體的架構上，功能較為強大的裝置，內建的處理器晶片通常為 CPU，並搭載一個作業系統(OS)運行，如嵌入式 Linux 甚至是 Android，並可能具備了檔案系統(File System)可供管理者進行相關操作，例如安裝軟體/進行客製化的 script 動作等；但若是較陽春的裝置，則可能為無 OS 的 MCU 嵌入式系統，只能透過固定的內建韌體執行相關功能。而在本研究中，針對適用的 IoT 裝置限制條件如下：

1. 裝置本身必須具備有作業系統和檔案系統
2. 主晶片的運算能力必須能夠執行資料加解密功能
3. 作業系統會將可執行檔集中於某些特定目錄下(如 binary folder)

3.5 系統架構與設定

在本研究的環境設定我們設計了一套如圖 3.2 所示的系統環境架構，其中包含了一個不安全的 IoT 裝置，這個裝置的工作除了執行日常的工作任務外，還必須要能夠建立和維護日誌檔案，這檔案包含了一組關於任務、事件、程序或是工作的稽核日誌項目。但是它本身沒有足夠的實體/軟體安全機制來保證它不會絕對被攻擊者所入侵，所以一旦在攻擊者入侵成功並獲得對 IoT 裝置的控制之後，也沒有任何安全措施可以保護已寫入的稽核日誌項目。並且在被成功入侵之後，無論攻擊者希望如何改寫過去的日誌紀錄，IoT 裝置都可能會全部寫入日誌項目中，例如變造或消除相關入侵事件紀錄。

另外，在此系統中還有一台具有高度的安全性，並預設不會被攻擊者入侵竄改或竊取資料的伺服器。而此伺服器透過可信賴的網路傳輸與 IoT 裝置做連結，並可互相傳送資料。

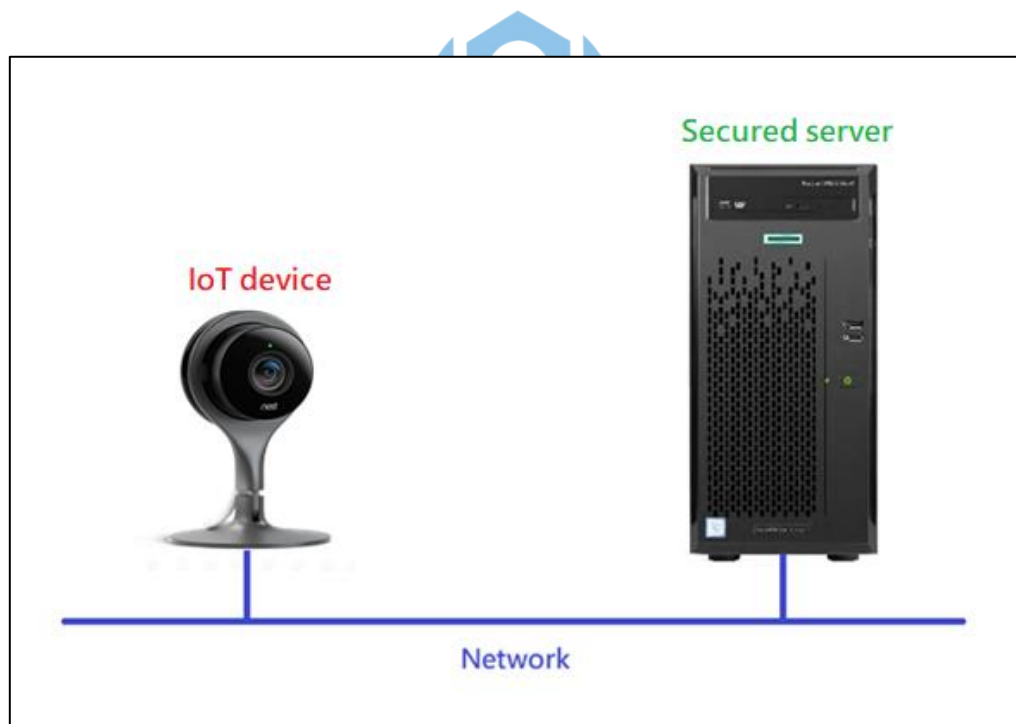


圖 3.2 研究架構設定圖

3.6 系統功能

本研究的系統功能流程如圖3.3所示。

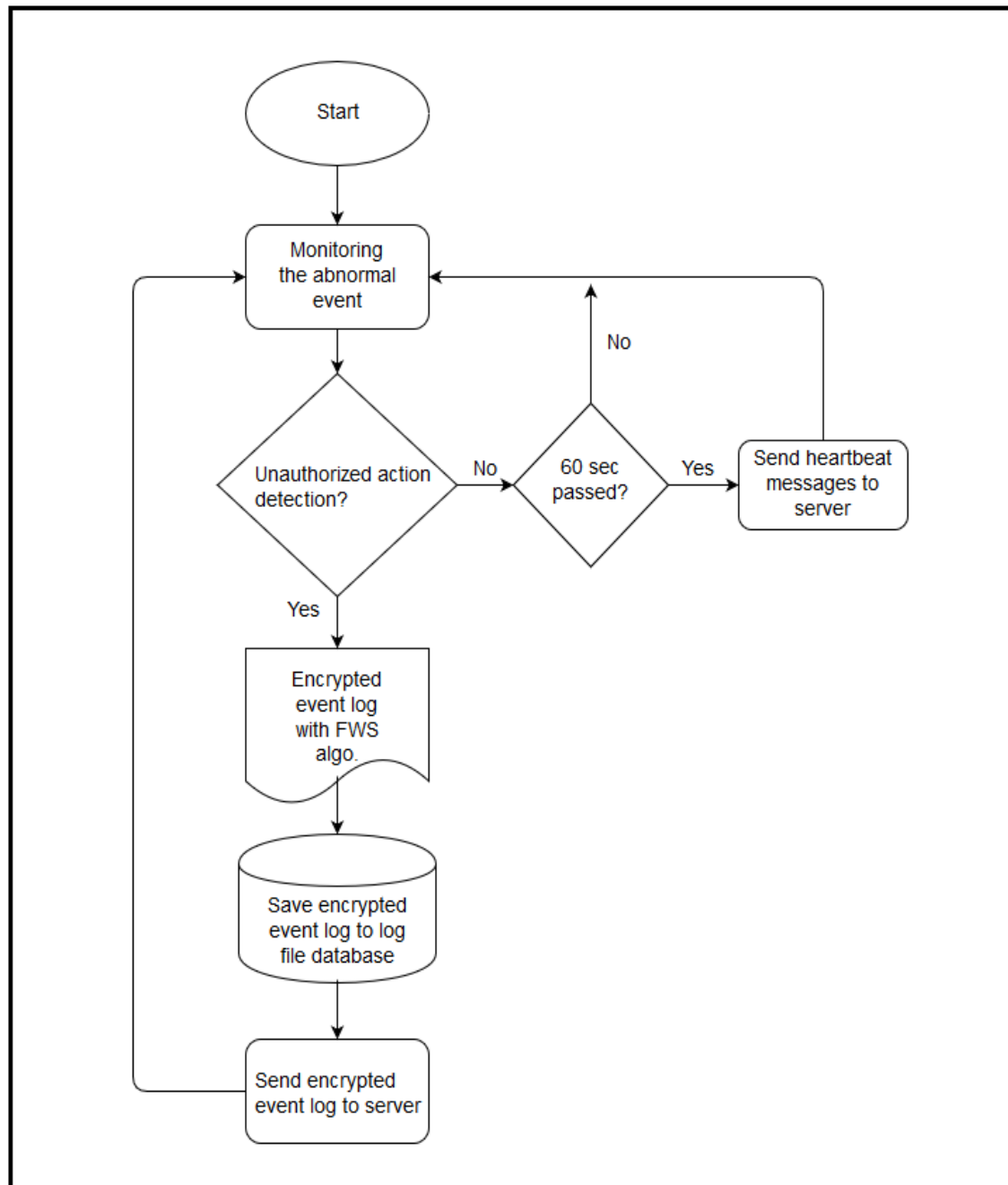


圖 3.3 系統功能流程圖

3.6.1 即時檔案異動偵測

本系統的第一個目標功能為能夠即時偵測IoT裝置中的特定目錄(如binary folder)檔案異動的情形，因此利用了Linux 核心子系統中的inotify function，改寫出一套對檔案系統的檔案變化情形做出通知的警示機制。

流程簡述如下，如圖3.4所示，我們的程式會對目標目錄產生出並watch list，此list包含被監測檔案的inode和mask資料。若IoT裝置被入侵，攻擊者在此監控目錄下執行了檔案新增、刪除和修改的事件，則會立即在kernel space依序產生事件佇列，此事件佇列內容可經由system call(如read())讀回 user space，則上層API則可以立即得知檔案異動內容。

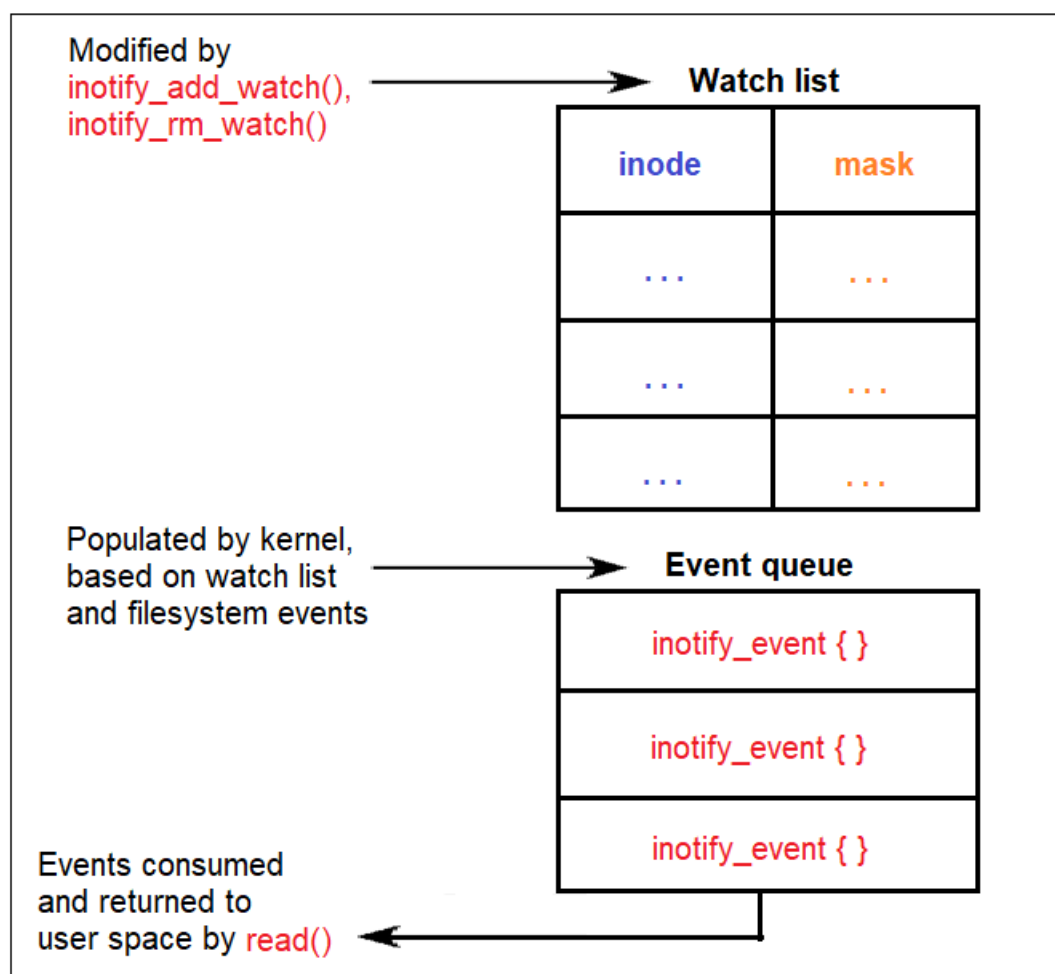


圖 3.4 inotify 架構示意圖

3.6.2 產生具有前向安全性的加密日誌項目

本系統的第二個目標功能為，當疑似遭到入侵的檔案異常事件發生，能夠將IoT裝置系統中產生的這些事件內容項目，以具有前向安全性的演算法做加密並存入日誌檔案，使得入侵裝置的攻擊者無法竄改過去的日誌內容，提供了日誌安全性的強力保證。

因此，我們採用了Bruce Schneier 和John Kelsey所提出的前向安全性加密演算法，即IoT裝置在事先會與安全伺服器共享一個初始共同密鑰 A_0 ，利用它來加密此次的檔案異常事件日誌內容然後儲存起來，並會將密鑰做單向不可逆的演化成 A_1 ，以實行下一次的內容加密，並立即刪除舊有的密鑰，每次新的日誌產生時都以此程序類推。因為攻擊者無法得知入侵時間以前所產生的密鑰，也就無法竄改過去的日誌內容。如此一來，系統就能夠產生出具有前向安全性的日誌項目，便也可以確保日誌項目內容的安全與正確性，

3.5.3 IoT裝置與安全伺服器的通訊

我們將IoT裝置透過可信賴的網路與安全伺服器做溝通，如圖3.5所示，一般狀態下，該裝置會以固定的時間頻率傳送固定的訊息給伺服器，作為IoT裝置系統正常存活的信號通知(heartbeat)。但是當檔案異常事件發生時，該裝置會將前述的前向安全性演算法加密過的日誌項目透過網路立即傳遞給安全伺服器作為同步紀錄，系統管理者也就能馬上得知異常事件發生，便可採取相對應的動作。

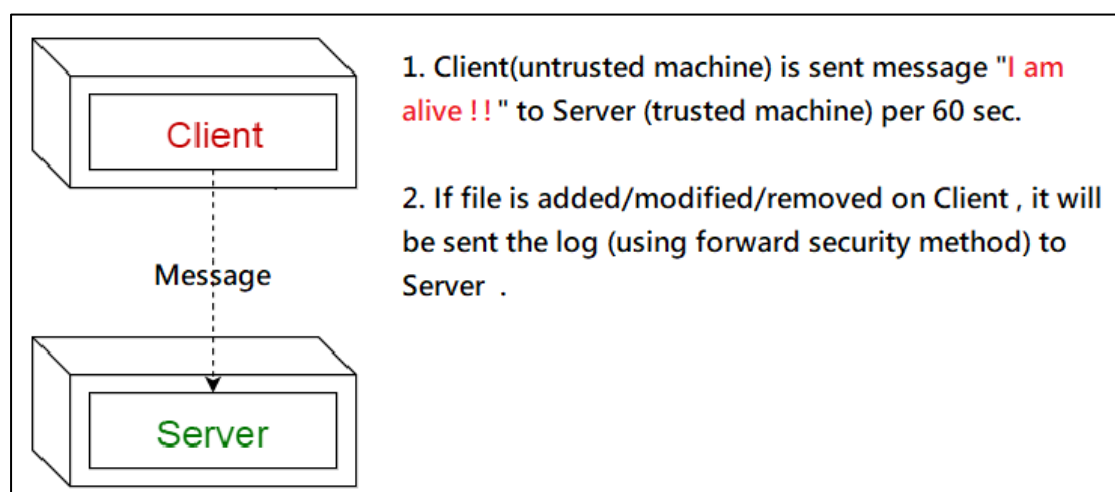


圖 3.5 heartbeat 示意圖

3.7 預期目標

此系統完成運作時，我們希望得到的結果是當入侵者取得IoT裝置的控制權並新增或修改特定目錄下的檔案時，系統會將異常事件的內容以具有前向加密性質的加密演算法被加密，系統管理者也可以第一時間得知此異常事件，

安全伺服器也會同步接收到此加密過異常事件紀錄，因為持有共同金鑰，所以可以解密出此異常紀錄的明文，並也可根據此紀錄對IoT裝置做進一步安全查核的動作。

因為日誌檔案皆以具有前向加密性質的加密演算法被加密過，以至於入侵者無法去偽造過去的虛假事件記錄，因此在異常事件發生之前的日誌檔案安全性得以確保。



第 4 章 實驗設計與結果

4.1 情境與架構環境

在本研究中所設定的實驗情境，是當有攻擊者入侵 IoT 設備並植入檔案，警示系統如何讓使用者/監控者立刻得知並做相關的處置和稽核？

另外在我們設定的 IoT 裝置，是安裝了作業系統及 file system，並且 CPU 必須具備加解密功能等運算能力，所以我們選擇以與市面上類似之 IoT 裝置架構的 Raspberry Pi 應用開發板作為 Untrusted machine；另外，我們以一台安裝了功能強大的 Linux 作業系統並具有高度實體與網路安全防護設定的伺服器作為 Trusted server。

此外，我們使用了 AES 對稱加密法來執行加密工作，並利用了 sha256 來實作 key 的雜湊運算。實際的架構與環境，請見圖 4.1。

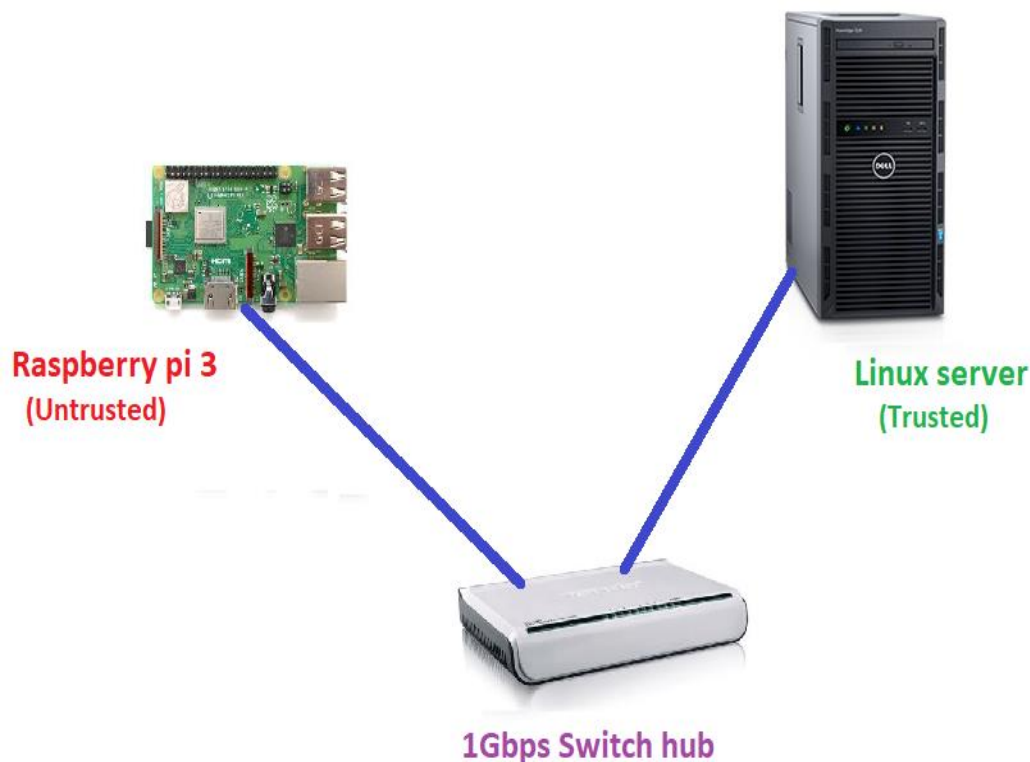


圖 4.1 實驗架構與環境

實驗設備的硬體詳細資訊如下：

■ Untrusted Machine

- Raspberry pi 3 -Model B
- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- 100 Base Ethernet
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- NOOBS (operation system)

■ Trusted Machine

- Linux server
- Quad-core Intel® Xeon® processor 5500
- 48GB DDR3 RAM
- 100/1000 Base Ethernet
- Ubuntu server 16.04 (operation system)



■ Reliable Network

- 1000Mbps Ethernet switch hub
- BCM5665 multilayer switch CPU
- 10/100/1000Mbps with Auto-MDI/MDIX
- High speed with non- blocking

4.2 過程與結果

依實驗結果，日誌項目以前向安全性方式加密的條件過程與產生。

假設系統偵測機制產生的第四筆日誌的明文(D_4)如下：

“ 2019/05/25 17:32 Event: Unauthorized action detected !!
Target file: file1.bin
Target action: create , write “

則依序產生條件與結果如下：

1. $A_4(A\text{-key}\#4)$:

8bc8cb9122d03b33d91f1216382666bce9d55f92ca84b399986fc3eb4bf9a4ea

2. $B_4(B\text{-key}\#4)$:

9ad494e2bcd32cbf716c136860c3c6bee9e5253c548c1eb4cefee88c02ald6f4

3. (B_4, A_4) :

9ad494e2bcd32cbf716c136860c3c6bee9e5253c548c1eb4cefee88c02ald6f4,
8bc8cb9122d03b33d91f1216382666bce9d55f92ca84b399986fc3eb4bf9a4ea

4. $K_4 = H(W_4, A_4)$:

bf23f4a0bd5b003d95f6e7681be0beec1c1ef45463091416ae438ad8399a6d77

5. $B_5 = H(B_4)$ (next A-Key):

8a684c032ba76ef83e0df48f219a7e5a12f2b6bc3000139644115acd8f253a36

6. $E_{K_4}(D_4)$:

33f485186f7d04b3658d1ff57f2b6162948a716c6de2e8e205852e36f552f12b9
ab5b4e54ffe6ebae539192cf2ea0e96218288f7a78e573f4e444c41a97af8af7a
27d3265f0

7. Y_3 (the last Y_j):

19ed96c20741bbaf8960966ce08a0d15ab46736898c1f9d412940904e60cf13b

8. $Y_4 = H(Y_3, E_{K_4}(D_4))$:

c363611106839708184bb44f72a2bf31cf1aff50e1cc1818102123b2e186a8e7

9. $Z_4 = MAC_{A_4}(Y_4)$:

53b931889b0a7be2485c2269d62a38

10. $L_4 = (E_{k_4}(D_4), Y_4, Z_4)$:

33f485186f7d04b3658d1ff57f2b6162948a716c6de2e8e205852e36f552f12b9
ab5b4e54ffe6ebae539192cf2ea0e96218288f7a78e573f4e444c41a97af8af7a
27d3265f0, c363611106839708184bb44f72a2bf31cf1aff50e1cc1818102123b
2e186a8e7, 53b931889b0a7be2485c2269d62a38

11. $A_5 = H(A_4)$ (next A-Key):

9b2b1e04a3a0f5d6c5e275a9fbb1043988d846077b38fe45a795c6d2073065a2

最終我們會得到加密過的第四筆以前向安全性加密所產生的 Log entry(L_4)，與提供下一次加密使用的共同金鑰(A_5)和(B_5)。

此外，我們設計了以下三種情境並在後面詳述實驗結果：

1. 正常事件 log，以前向安全性加密演算法加密並傳送給 server
2. 偵測到監控目錄下的異常事件發生，以前向安全性的加密演算法來加密 log 並傳送給 server。
3. 偵測到監控目錄下的異常事件發生，未以具有前向安全性的加密演算法來加密 log 並傳送給 server。

情境 1.

- 平時一般的狀況下，Raspberry Pi 會以固定的時間頻率傳送固定的信號通知訊息給 Linux server，以確認該裝置的持續正常存活狀態 (heartbeat)。
- 當 Raspberry Pi 產生了一般性的 event 事件後(如非特權使用者登入)，此事件會在 Raspberry Pi 的系統紀錄中作登錄。同時 Raspberry Pi 會立即產生 Event Log，並以前向安全性加密演算法加密，會依序計算產生此演算法所需的各項條件和數值，過程如圖 4.2 所示，並立即將此加密過的 Event Log 透過高速網路傳送給 Linux server。

```
Monitoring...
Log message #3 :
    2019/05/25 17:08      User log in : Amy Liu
                          Not Root Administrator.
                          Permission granted.
Update system level log.
Creating and encryptig new event log #3...
Send event log to trusted server...
Save to log file...
Monitoring...
```

圖 4.2 情境 1 - Raspberry Pi 之實驗結果

- Linux server 收到此加密的 Event log 訊息資料後，因為和 Raspberry Pi 擁有共同的初始密鑰 A_0 ，也因此可以計算出演化後的密鑰 A_t ，所以可以順利解密訊息並得知內容，如圖 4.3 所示。

```
Message: I am alive.
Message: I am alive.
Message: I am alive.
New event log (#3) received from untrusted machine...
Generating Key A3...
Decrypting the enent log #3...
Decrypted succeed...
Origin message:
    2019/05/25 17:08      User log in : Amy Liu
                          Not Root Administrator.
                          Permission granted.
Processing...
Done.
Message: I am alive.
Message: I am alive.
```

圖 4.3 情境 1 - Linux server 之實驗結果

情境 2.

- 我們在 Raspberry Pi 上利用 inotify 功能設計了即時監控在特定目錄下的檔案(子目錄)的功能，平時一般無遭受入侵的狀況下，Raspberry Pi 會以固定的時間頻率傳送固定的信號通知訊息給 Linux server，以確認該裝置的持續正常存活狀態(heartbeat)。
- 當模擬入侵者向監控的目標實施攻擊動作時，如被監控的檔案(目錄)被新增、修改、刪除等，此事件會立即在 Raspberry Pi 的系統警告紀錄中作登錄。同時 Raspberry Pi 會立即產生 Event Log，並以前向安全性加密演算法加密，過程如圖 4.4 所示，並立即將此加密過的 Event Log 透過高速網路傳送給 Linux server。

```
Monitoring...
waring!!

2019/05/25 17:32      Event: Unauthorized action detected !!
                        Target file: file1.bin
                        Target action: create , write

Update system level log.
Creating and encryptig new event log #5 with FWS Algo. ...
Send event log to trusted server...
Save to log file...
Monitoring...
```

圖 4.4 情境 2 - Raspberry Pi 之實驗結果

- Linux server 收到此加密的 Event log 訊息資料後，因為和 Raspberry Pi 擁有共同的初始密鑰 A_0 ，也因此可以計算出演化後的密鑰 A_t ，所以可以順利解密訊息並得知內容，如圖 4.5 所示。

```

Message: I am alive.
Message: I am alive.

New event log (#5) received from untrusted machine...
Generating Key A5...
Decrypting the event log #5...
Decrypted succeed...

Origin message:

    2019/05/25 17:32      Event: Unauthorized action detected !!

                          Target file: file1.bin
                          Target action: create , write

Processing...
Done.

Message: I am alive.
Message: I am alive.

```

圖 4.5 情境 2 - Linux server 之實驗結果

- 因為所有的 Event log 都是使用前向安全性加密法，入侵者就算取得當前的密鑰，也無法順利將在他入侵之前的 Event log 解密成正確的明文，如圖 4.6 所示。

```

Trying to modifying the event log #5...
Searching...
Find Key A6...
Decrypting the event log #5...

ISM1BcNe9zbRMJNafF1cvXHgjj8njBrpm1CFcBNvcfyVvb2n0
Uoa1hBJn7Biy86Vl3XPi8sXlDgRT2CWEzaC7NGlot20hmyvq3n
3wD9mBlgSfVC9t1arLsZySQAJJawWPqMOevjv8dIOh9uAvGfyvZzQzW6aixdevFD
unrqps0y2ywadwWDoDKktmTzYI8PTU6hN8pJh4y3FtKf8A

Done.

```

圖 4.6 情境 2 - 模擬入侵者之實驗結果

情境 3.

- 我們在 Raspberry Pi 上利用 inotify 功能設計了即時監控在特定目錄下的檔案(子目錄)的功能，平時一般無遭受入侵的狀況下，Raspberry Pi 會以固定的時間頻率傳送固定的信號通知訊息給 Linux server，以確認該裝置的持續正常存活狀態(heartbeat)。
- 當模擬入侵者向監控的目標實施攻擊動作時，如被監控的檔案(目錄)被新增、修改、刪除等，此事件會立即在 Raspberry Pi 的系統警告紀錄中作登錄。同時 Raspberry Pi 會立即產生 Event Log，並以一般的(單一密鑰)加密演算法加密，過程如圖 4.7 所示，並打算立即將此加密過的 Event Log 透過高速網路傳送給 Linux server。

```
Monitoring...
waring!!

2019/05/27 10:45      Event: Unauthorized action detected !!
                        Target file: file2.bin
                        Target action: create , write

Update system level log.
Creating and encryptig new event log #8 with non-FWS Algo. ...
Send event log to trusted server...
Save to log file...
Monitoring...
```

圖 4.7 情境 3 - Raspberry Pi 之實驗結果

- 雖然圖 4.7 的流程看似正常，但是事實上此入侵者在入侵 Raspberry Pi 時就已經取得此加密法的唯一密鑰 A_n ，並在 Raspberry Pi 把加密完成後的 Event log 送出給 server 之前就將程序暫停，並透過獲取的密鑰來解密出 Event message 的明文內容，且將之竄改為非警示內容後，再以同樣的加密法和密鑰來加密，並透過同一程序送出給 server，如圖 4.8 所示。

```

Searching...
Find Key An...
Decrypting the existing event log #8...

2019/05/27 10:45      Event: Unauthorized action detected !!

                        Target file: file2.bin
                        Target action: create , write

Trying to modifying the event log #8 and system log...

2019/05/27 10:45      User log in : Bill Wang

                        Not Root Administrator.
                        Permission granted.

Update system level log.
Creating and encryptig new event log #8 ...

```

圖 4.8 情境 3 - 模擬入侵者之實驗結果

- Linux server 收到此加密的 Event log 訊息資料後，因為和 Raspberry Pi 擁有共同的單一密鑰 A_n ，所以可以順利解密訊息並得知內容，但是因為內容已被入侵者所竄改，所以 server 解密出的內容為入侵者竄改過後的內容，而非真正的 event log 內容，如圖 4.9 所示。

```

Message: I am alive.
Message: I am alive.

New event log (#8) received from untrusted machine...
Decrypting the enent log #8 with Key An...
Decrypted succeed...

Origin message:

2019/05/27 10:45      User log in : Bill Wang

                        Not Root Administrator.
                        Permission granted.

Processing...
Done.

Message: I am alive.
Message: I am alive.

```

圖 4.9 情境 3 - Linux server 之實驗結果

4.3 實驗結果分析

根據實驗的結果，我們得到以下的觀察：

- Raspberry Pi 由於使用了前向安全性加密演算法，入侵者無法對攻擊時間點 t 之前的 Event Log 作竄改或是偽造的動作，入侵者的動作將會如實的被記錄起來。
- Raspberry Pi 也會在同一時間，將加密過後的 Event Log 傳送給 Linux Server，使用者/監控者可以更有效地得知 Raspberry Pi 何時開始變得不安全，以利下一步的動作。因此可以對 Raspberry Pi 做進一步的安全稽核工作，以確認是否真的遭到入侵者攻擊。
- 且每一筆 Event Log 都是以按照編號依序傳送給 Linux Server，當入侵者刪除了單筆紀錄或是整個紀錄區塊時，後續啟動的稽核機制也將可以發現此種異常狀態，因此可以更加提高此系統的安全性。

反之，若是沒有使用具有前向安全性加密演算法來加密 event message 內容，則入侵者很輕易的即可將已加密的 event log 解密並竄改內容，並進一步的欺騙稽核伺服器。

更進一步來看，本研究所提出的方法是否能應用在目前較普及的 IoT 設備，下表整理了 Raspberry Pi 與這些設備的比較。

Device	CPU
Raspberry Pi 3	64bit Q-core ARM Cortex-A53
IPcam	16bit ARM RISC
Intelligent Instrument	16bit ARM Cortex-MX
Wearable device	16bit ARM Cortex-M0
Apple watch	64bit Apple S3 CPU

第 5 章 結論

本研究設計了一個適用於 IoT 裝置的可確保日誌前向安全性的入侵偵測機制，除了當入侵者在一個或是多個被監視的特定可執行目錄下植入檔案時，能夠即時發出警告通知系統管理者採取相對應的措施。此外也利用了具備前向安全性的加密演算法，即使在入侵者成功取得設備的控制權之後，裝置的系統仍然可以確保過去所有已產生的日誌檔案內容都是無法被入侵者修改的。

根據實驗結果，我們驗證了當入侵者取得IoT裝置的控制權並新增或修改特定目錄下的檔案時，此機制可以將異常事件的內容以具有前向安全特性的加密演算法被加密，系統管理者也可以在同一時間得知此異常事件，安全伺服器也會同步接收到此加密過異常事件紀錄。也因持有共同金鑰，伺服器可以解密出此異常紀錄的明文，並也可根據此紀錄對IoT裝置做進一步安全查核的動作。而且日誌檔案皆以具有前向安全特性的加密演算法被加密過，以至於入侵者無法去偽造過去的事件記錄，因此在異常事件發生之前的日誌檔案安全性得以確保。

雖然此機制有達到我們預期的功能需求，但是仍有其限制條件。例如無法如防毒軟體一般可以對整個檔案系統作監控，而是只能監視多個特定的執行檔目錄，但若是入侵者在其它目錄取得執行權限時，此機制很可能就會失效。另外，若是IoT裝置有作韌體更新的動作時，也有可能造成此機制的誤判。因此，在未來的研究上，可以針對這些缺點作更進一步的改善，讓此系統機制能更趨完善。

參考文獻

- [1] “The Internet of Things , How the Next Evolution of the Internet Is Changing Everything.”
https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
- [2] “Number of Connected IoT Devices Will Surge to 125 by Billions by 2030, HIS Markit Says.”
<https://technology.ihf.com/596542/number-of-connected-iot-devices-will-surge-to-125-billion-by-2030-ihf-markit-says>
- [3] “FDA confirms St. Jude heart monitors can be hacked .”
<https://www.rt.com/usa/373157-fda-st-jude-cybersecurity/>
- [4] “2016 Dyn cyberattack
”https://en.wikipedia.org/wiki/2016_Dyn_cyberattack
- [5] Mihir Bellare and Sara K. Miner, “A Forward-secure Digital Signature Scheme,” Advances in Cryptology — CRYPTO’ 99, Lecture Notes in Computer Science, vol. 1666, pp. 431-448, 1999, Springer.
- [6] Christoph G. Günther. An Identity-Based Key-Exchange Protocol. In Advances in Cryptology— EUROCRYPT ’ 89, pages 29 – 37, Houthalen , Belgium, 1989. Springer.
- [7] Ross Anderson, “Two Remarks on Public Key Cryptology,” In Manuscript and Invited Lecture at the 4th ACM Conference on Computer and Communications Security, April 1997.
- [8] Bruce Schneier and John Kelsey, “Cryptographic Support for Secure Logs on Untrusted Machines,” USENIX Security Symposium, January 26-29, 1998.
- [9] Bruce Schneier and John Kelsey, “Secure Audit Logs to Support Computer Forensics,” ACM Transactions on Information and System Security (TISSEC) Vol. 2, Is. 2, pp. 159-176, May 1999.
- [10] Douglas R. Stinson, Cryptography: Theory and Practice (2nd ed.). 2002, CRC Press.
- [11] Bruce Schneier, Applied Cryptography (2nd ed.): Protocols, Algorithms, and Source Code in C. 1996, John Wiley & Sons.
- [12] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography. 1997, CRC Press.
- [13] Stuart Haber, Scott Stornetta, “How to Time-Stamp a Digital Document,” Advances in Cryptology (CRYPTO ’ 90), Lecture Notes in Computer

- Science, vol. 537, pp. 437–455, 1990, Springer-Verlag.
- [14] Bruce Schneier and John Kelsey, Automatic event-stream notarization using digital signatures. Security Protocols, International Workshop April 1996 Proceedings, Springer-Verlag, pp. 155–169 ,1997.
 - [15] A. Perrig, J. Stankovic, and D. Wagner. “Security in wireless sensor networks,” Vol. 47, Is. 6 , pp. 53–57, June 2004.
 - [16] **Kenneth C. Barr**, and **K. Krste Asanović**, “Energy aware lossless data compression.” ACM Transactions on Computer Systems (TOCS) , Vol. 24, Is.3, pp. 250–291, August 2006.
 - [17] Di Ma ,Gene Tsudik, “Forward-secure sequential aggregate authentication.” Proceedings of the 2008 ACM symposium on Information, pp. 86–91 , May 2007.
 - [18] D. Boneh, B. Lynn, and H. Shacham , “Short signatures from the Weil pairing” , In proceedings of Advances in cryptology – CRYPTO’ 01, LNCS (2001), pp. 514–532 , 2004
 - [19] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham , “Sequential aggregate signatures from trapdoor permutations” , Advances in Cryptology – EUROCRYPT 2004 pp. 74–90 , May 2004.
 - [20] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham , and Brent Waters , “Sequential aggregate signatures and multi signatures without random oracles” , In Proc. of Eurocrypt 2006, May 2006.
 - [21] D. Boneh, B. Lynn, and H. Shacham , “Aggregate and verifiably encrypted signatures from bilinear maps” . In Proc. of Proceedings of Eurocrypt 2003, vol. 2656 of LNCS May 2003.
 - [22] Sepideh Avizheh , Tam Thanh Doan, Xi Liu and Reihaneh Safavi-Naini , “A Secure Event Logging System for Smart Homes” , Session 3: Defense against IoT Hacks, IoT S&P’ 17, November 3, 2017.
 - [23] Vishal Karande, Erick Bauman, Zhiqiang Lin, Latifur Khan, “SGX-Log: Securing System Logs With SGX” , ASIA CCS ’ 17 Proceedings of ACM on Asia Conference on Computer and Communications Security , 2017.
 - [24] “Linux Filesystem Events with inotify.”
<https://www.linuxjournal.com/content/linux-filesystem-events-inotify>