國立高雄科技大學(第一校區)

電子工程系

專題報告書

基於可信任平台模組晶片之軟體智財保護

專題生：

林子豪 ZI-HAO LIN

指導教授：陳朝烈博士 Dr.Chao-Lieh Chen

中華民國一一二年五月

摘要

資訊技術不斷進步，使得軟體程式應用面以及商品愈來愈多，軟體程式的開發需要耗費大量資源，然而，軟體盜版和非法使用卻也愈加嚴重，這些問題會對軟體開發商造成巨大的經濟損失，因此，軟體開發商需要建立保護措施以此面對層出不窮的破解以及適應不同的軟體執行環境以避免漏洞，本專題利用 TPM(Trusted Platform Module)晶片模組建立一個軟體商品程式保護方式，TPM 是一種硬體式安全晶片，其設計目的是提供一個安全的環境，以保護電腦系統免受攻擊、入侵、串改，每個 TPM 在生產時都各自綁定一組唯一的非對稱式密鑰，並儲存在 TPM 晶片的不可讀區域中，並且無法透過任何方式獲取根密鑰，利用 TPM 除了能達到軟體商品啟動防盜以外，也能對 AI 模型加密以達到保護其內各種資訊，API 的形式搭配 TPM 的特性使得該專題於各種本地端的加密及驗證都能產生作用。

目錄

圖目錄

第一章 緒論

1.1 研究動機與目的

人工智慧（AI）訓練出來的模型和應用涉及了許多重要的資訊，包括但不限於個人資訊、知識產權、商業機密或其他公司資產等。然而，軟體盜版、非法使用以及資訊外洩等問題的出現，會對軟體開發商造成巨大的經濟損失。為了解決無網路環境下軟體保護的問題，保護軟體程式在工廠無網路環境中的執行，並實現軟體程式與設備之間的綁定，以避免軟體被盜版使用或濫用（單個程式被分到多台設備使用）。為了實現這個目標，將使用可信平台模組（TPM）技術，以確保軟體程式的安全性和完整性。

1.2 專題相關工具

1. TPM (Trusted Platform Module)
   - PCA-TPM(B1)
2. IPC （Industrial PC)
   - Advantech IPC-7132MB-50B
   - Advantech ARK-3532
   - Advantech ARK-1123
3. Linux
   - Ubuntu 20.04
4. Package
   - tpm2-tools
   - tpm2-abrmd

（圖 1）TPM 晶片正面圖

（圖 2）TPM 晶片反面圖

（圖 3）Advantech ARK-1123

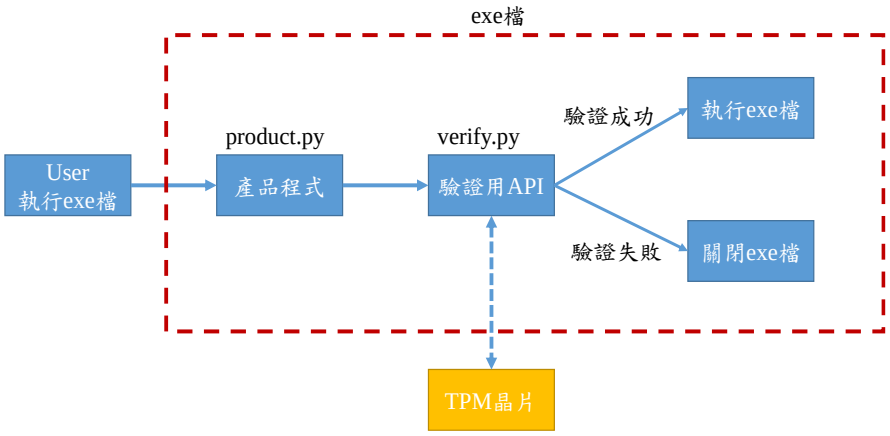
（圖 4）Advantech ARK-3532

（圖 5）Advantech IPC-7132MB-50B

第二章

研究架構與方法

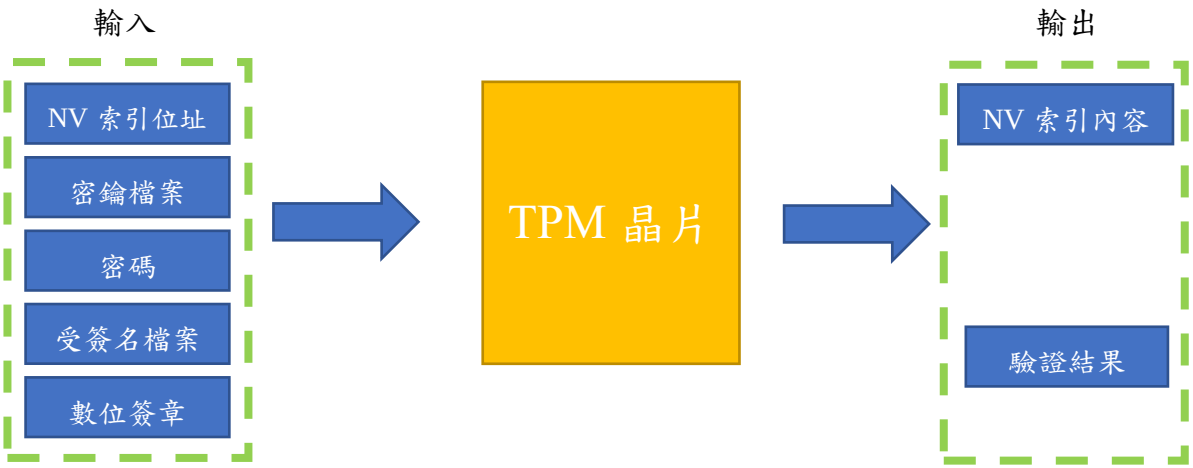
2.1 專案架構



(圖 6)為專案架構圖

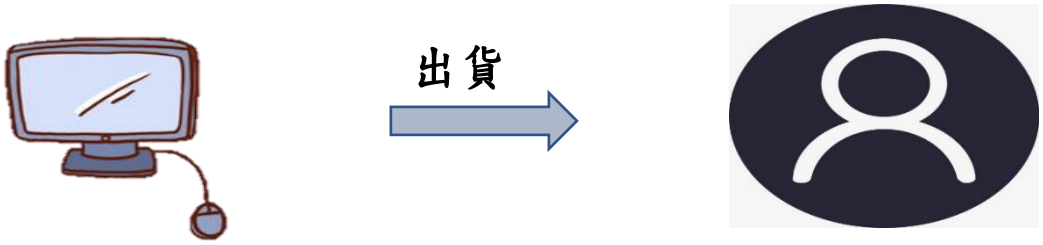
專案架構內所用到的數位簽章、金鑰以及執行動作都存在於 TPM 內部，於要保護用的產品程式(product.py)內呼叫模組化的程式(verify.py)，再透過模組化的程式(verify.py)對 TPM 晶片模組進行驗證動作，verify.py 對 TPM 晶片模組傳送指令，TPM 晶片模組回傳結果給 verify.py，verify.py 使用回傳的結果判斷驗證成功與否，成功則繼續執行.exe 檔，失敗則直接關閉程式，由於模組化的驗證程式，同一台設備上的程式都可使用以此達到保護、驗證效果。

2.2 TPM 架構



(圖 7)為 TPM 輸入輸出架構圖

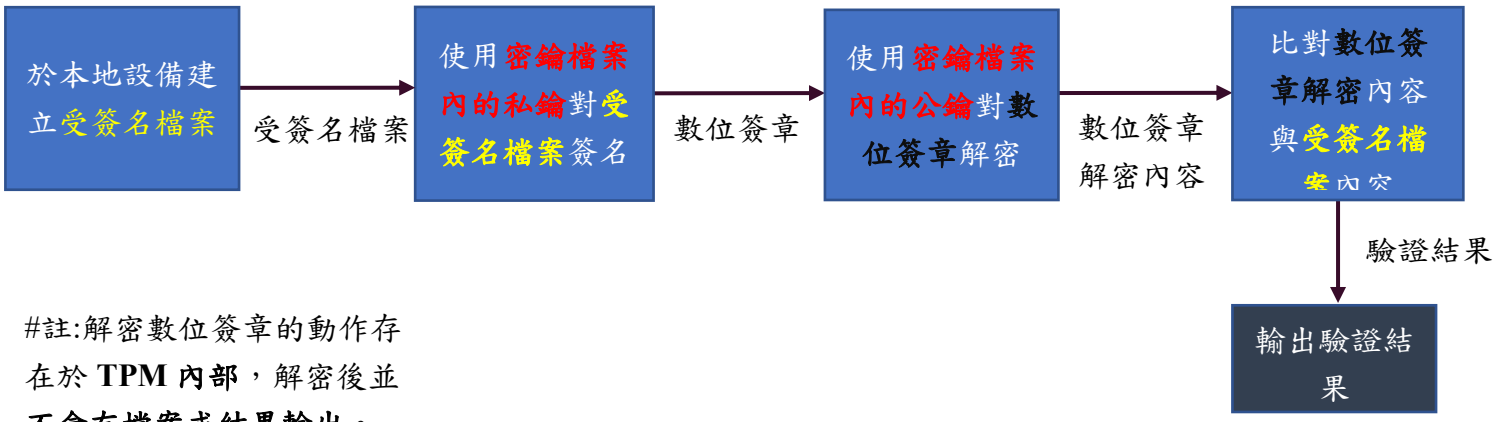
對 TPM 的操作，區分成輸入與輸出，輸入部分的金鑰以及數位簽章檔案平時儲存於 TPM 內部，透過權限以及密碼方式確保除硬體破壞外無法被其他方法提出或是破解，NV 索引位址以及密碼則是產品出貨前設定好的，於程式因為寫成了模組化，因此能自行設定及調整 NV 索引位址、密碼等輸入部分。



(圖 8)為出貨流程圖

產品的出貨包括設備，不僅限於程式，所以能做到出貨前的設定且方法不被消費者所知。

2.3 驗證架構



#註:解密數位簽章的動作存在於 **TPM 內部**，解密後並**不會有檔案或結果輸出**。

(圖 9)為驗證架構圖

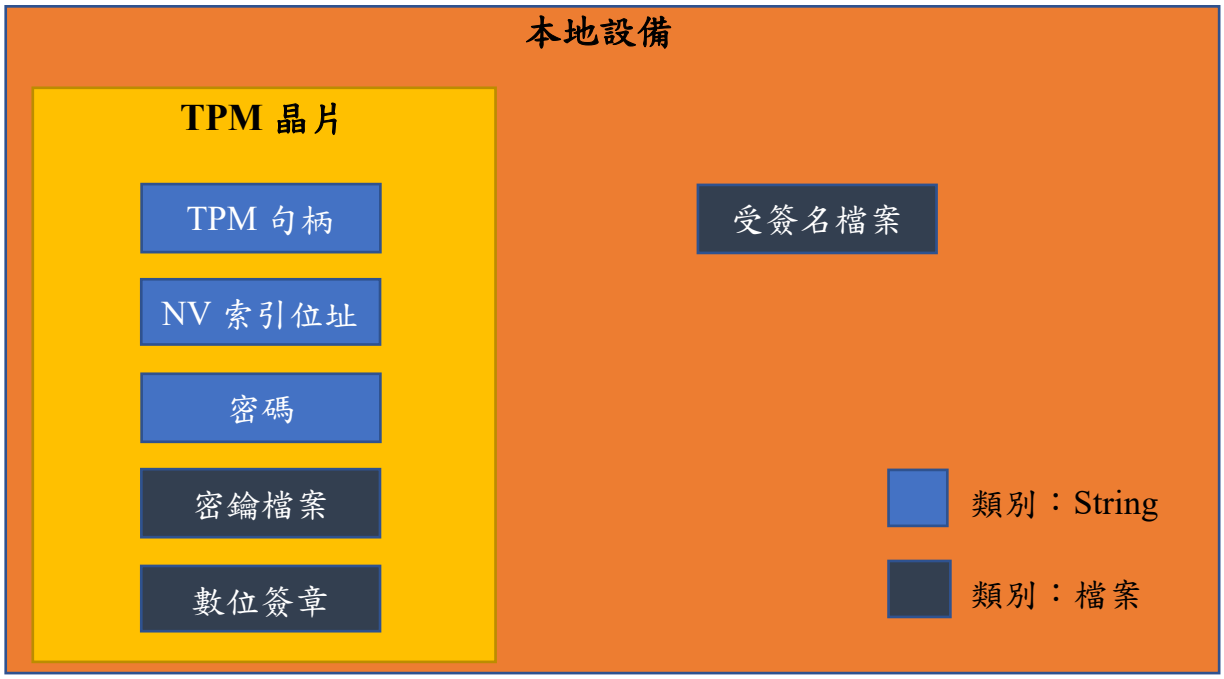
透過對開源的 tpm2-tools Source Code (https://github.com/tpm2-software/tpm2-tools)的修改，並且配置和安裝於設備上，使得驗證架構可以利用 TPM 的金鑰對檔案進行簽名以及驗證的功能，並且對 TPM 設定密碼，意即無法透過 Terminal 等等，只能透過 API 方式使用該 TPM 流程。

- 密鑰檔案　：包含了 TPM 公鑰以及私鑰的檔案，公鑰和私鑰於 TPM 內部透過指定算法後加密導出，並匯入密鑰檔案。每次重新導出的密鑰檔案都不一樣，且檔案不可讀。檔案類型(.ctx)。

- 受簽名檔案：用來產生以及驗證數位簽章的檔案，內容不限，可以是加密後檔案。檔案類型(.dat、.enc)。

- 數位簽章　：透過私鑰對檔案簽名(加密)產生出的不可讀檔案，可以透過使用公鑰對該數位簽章解密，並與受簽名檔案比較。檔案類型(.rssa)。

2.4 設定出貨用設備



(圖 10)為 TPM 設定架構圖

數位簽章的加解密以及驗證需要使用的有密鑰檔案、受簽名檔案、以及數位簽章，出貨前為了產生密鑰檔案以及數位簽章，會對設備進行設定，並於 TPM 外部產生密鑰檔案以及數位簽章，所以 TPM 外部會有密鑰檔案、受簽名檔案、以及數位簽章該三個檔案，為了提高安全性，將密鑰檔案以及數位簽章透過存放於 NV 索引位址以及 tpm 句柄內來達到不會受停電遺失以及被竄改，並且於 TPM 內設定一組使用者密碼，透過 TPM 進行的操作如果沒使用該密碼就會被阻攔並且該阻攔可抵抗字典攻擊，有了該方法就可避免駭客跳過程式直接對 TPM 進行檔案的提出或是權限設定的變更更。驗證用的受簽名檔案則依然存放於本地設備且 TPM 外部，不將該檔案也存放入 TPM 內部是因為檔案大小考量，而受簽名檔案可以是文字檔，也可以是壓縮檔等等。TPM 句柄並非一個直接的地址，他是一個取得另一對象的方法。

2.5 模組化驗證用 API

模組名稱	輸入	輸出	資料型態	方法
capture	rssaname filename rssapath filepath #註:rssa 為數位簽章，file 為被簽名檔案	sign	rssaname;str filename;str rssapath;str filepath;str sign;str	capture(rssaname,filename,rssapath,filepath)
模組名稱	**輸入**	**輸出**	**資料型態**	**方法**
verify	rssaname filename rssapath filepath capture()	True os._exit(1)	rssaname;str filename;str rssapath;str filepath;str	verify(rssaname,filename,rssapath,filepath)

(表1)為驗證用API模組列表

rssaneme,rssapath 及 filename,path 為數位簽章要釋出的名稱及路徑、被簽名檔案的檔案名稱以及路徑，使用到的參數型態皆為字串。

第三章

成果與貢獻

3.1 與其他保護程式方法比較

特性/方法	授權金鑰	USB 授權	雲端授權	HSM	FPGA	本專題
無網路需求(初次)		✓		✓	✓	✓
無網路需求(常態)	✓	✓		✓	✓	✓
無須連接外部硬體	✓		✓			✓
不受斷電影響	✓		✓	✓	✓	✓
不受更換硬體影響		✓	✓	✓		✓

(表2)為多方法特性比較表

註:HSM(Hardware Securitv Module)
FPGA(Field Programmable Gate Array)

優點:

1.**硬體層面保護**：和HSM、FPGA比起，並不需要額外物理連結和接線

2. **高彈性**：由於將專題模組化，所以於各種不同的程式內直接呼叫API就能達到驗證功能(該設備有TPM)

3. **唯一性**：由於將程式與設備綁定，就算將程式移植到其他設備上使用也無法正常執行，同一程式只有唯一一個使用者(設備)

4. **操作需要密碼**：密碼包括在API內，是為了防止駭客跳過程式自行對tpm進行指令操作，產品程式使用者並不需要額外操作

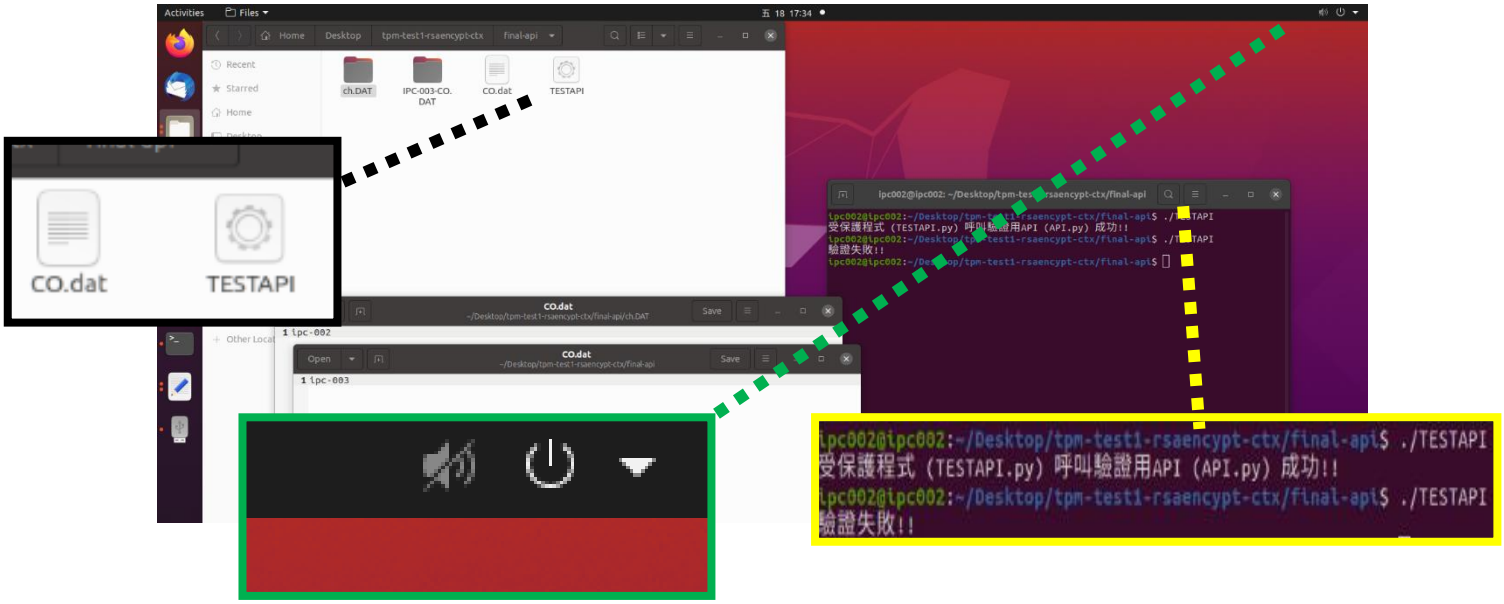
5. **無須網路環境執行**

6. **不受斷電影響**

缺點:

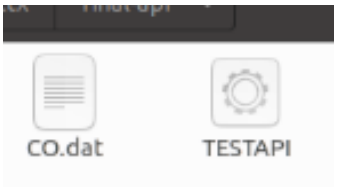
1.驗證用檔案遺失就需要重新手動設定

3.2 無網路環境下綁定設備正常使用軟體



(圖11)為IPC-002設備程式執行圖

於設備 IPC-002 上執行的程式為 TESTAPI(.exe)檔案，並且透過 ./TESTAPI 指令執行，該檔案包括了一個受保護程式(product.py)去呼叫驗證用模組化程式(API.py)，於設備右上角介面能看到當前並無網路環境。



(圖12) 為程式執行所需檔案圖

章節 2.3 提到驗證架構裡，會以密鑰檔案(.ctx)裡的私鑰檔案解密數位簽章(.rssa)，並將解密後內容與受簽名文件(.dat)即這裡的 CO.dat 檔案做比對，如果結果相同則驗證成功。

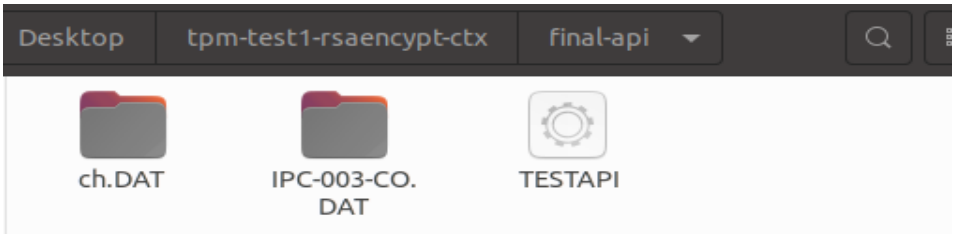
(圖13)為程式執行Terminal圖

於這次的測試裡，測試了兩個情境，第一個情境為，該程式執行於我們出貨的設備，所以受簽名文件是與該設備綁定的，即為第一個執行結果(驗證成功)，第二個情境為，該程式執行於我們出貨的設備，但是竄改、更動本身設備的受簽名文件或是移植了其他同批出貨設備的受簽名文件(CO.dat)，即為第二個執行結果(驗證失敗)。

（圖14）為各設備受簽名文件內容圖

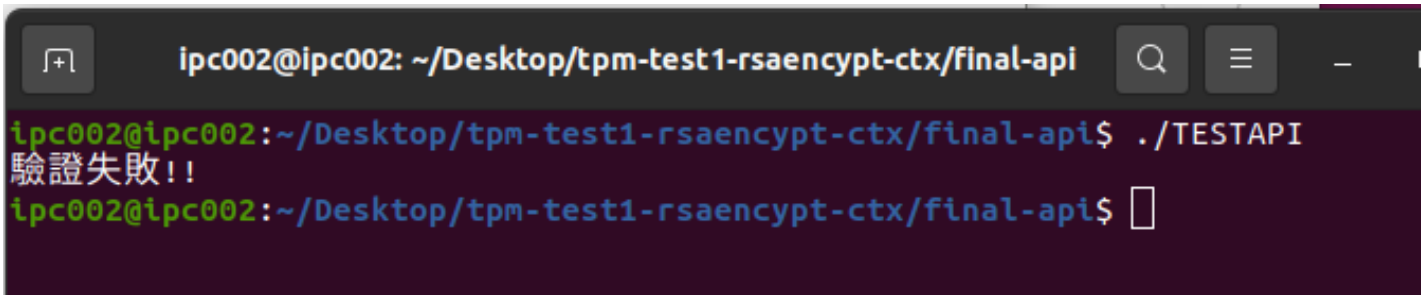
(章節 2.3)提到受簽名文件可以是多種檔案類型，這次測試使用.dat 檔案並且為了明顯區別各設備綁定的受簽名文件的差異，並以 IPC-002、IPC-003 做為內容，**(圖 11)**測試都於設備 IPC-002 上進行，第二次情境的情境模擬透過移植 IPC-003 設備上的受簽名文件(CO.dat)達成。

3.3 於其他設備上無法使用軟體



（圖15）為無受簽名文件(CO.dat)目錄圖

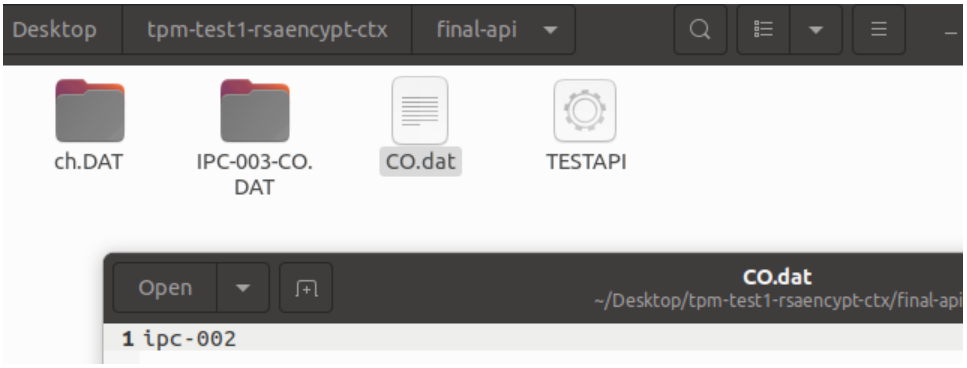
模擬一般設備(有TPM2.0晶片)獲取該軟體後(TESTAPI.exe)，因為無受簽名文件(CO.dat)，所以該目錄下只有軟體(TESTAPI.exe)。



（圖16）為無受簽名文件(CO.dat)執行圖

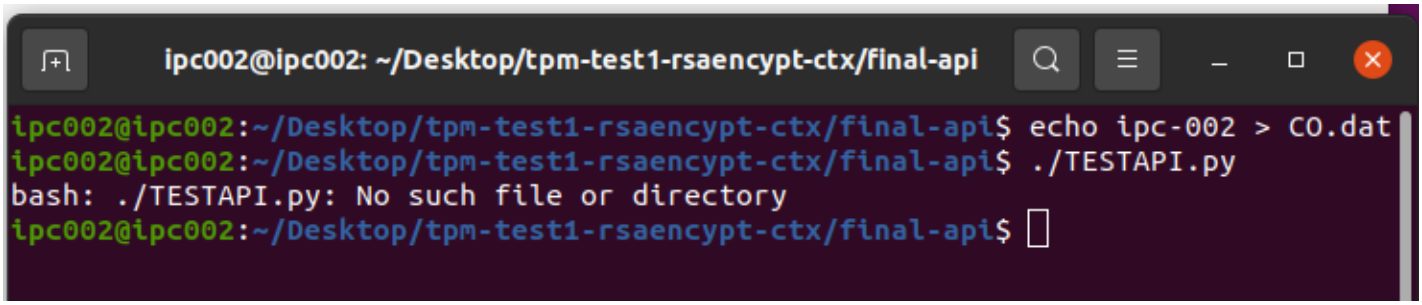
透過./TESTAPI 指令模擬一般設備(有 TPM2.0 晶片)獲取該軟體後(TESTAPI.exe)，無受簽名文件(CO.dat)的執行結果，可以顯然看見結果為**驗證失敗**。

3.4 偽造受簽名文件驗證測試



（圖17）為偽造受簽名文件內容圖

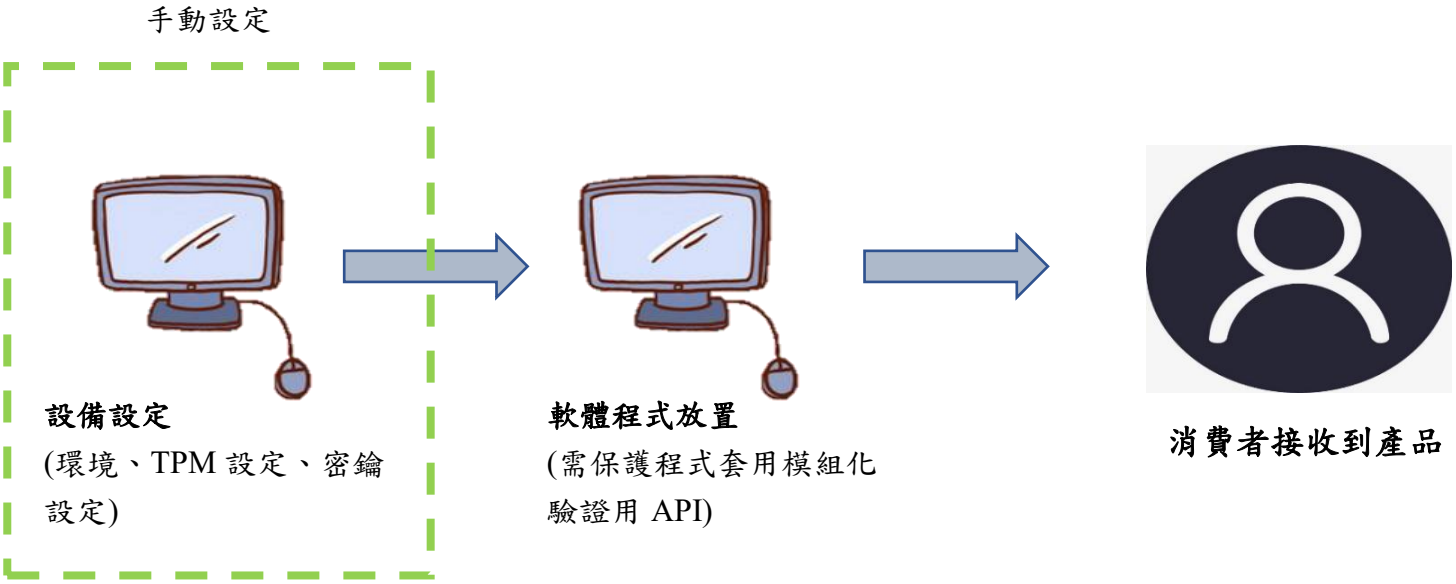
模擬一般設備(有TPM2.0晶片)獲取該軟體後(TESTAPI.exe)，以不知名方法得知受簽名文件(CO.dat)內容，並且創立一樣格式及內容的文件，該測試於IPC-002上執行，於**(圖14)**可看到綁定於IPC-002之受簽名文件內容為 ” ipc-002 ” ，所以這裡模擬建立一個內容為 ” ipc-002 ” 檔案名稱為CO.dat的受簽名文件。



（圖18）為偽造受簽名文件執行圖

透過 echo ipc-002 > CO.dat 指令，於當前目錄下建立內容為 ” ipc-002 ” 檔案名稱為 CO.dat 的受簽名文件，建立完成後再透過./TESTAPI 指令，測試結果，可以看到測試結果依然**無法成功驗證**。

第四章 未來展望



(圖 19)為出貨前設定流程圖

未來希望能夠將設備設定的部分腳本化，能夠快速設定，而非都透過手動慢慢設定。

第五章 參考文獻

1. Hardware: TPM module：
   https://paolozaino.wordpress.com/2018/06/15/tpm-module/
2. Linux: Configure and use your TPM 2.0 module on Linux：
   https://paolozaino.wordpress.com/2021/02/21/linux-configure-and-use-your-tpm-2-0-module-on-linux/
3. Trusted Platform Module Library (22.4節 Symmetric Encryption)：
   https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf
4. How to use TPM for encryptions：
   https://pagefault.blog/2016/12/23/guide-encryption-with-tpm/
5. 可信平臺（四）通訊的端點——TPM、保護存儲：
   https://www.twblogs.net/a/5e5054ccbd9eee21167d4acd
6. TPM Key相關概念：
   https://www.cnblogs.com/embedded-linux/p/6716740.html
7. TPM 2.0-NV 儲存體、原則、錯誤處理和證明：
   https://learn.microsoft.com/zh-tw/windows-hardware/test/hlk/testref/ec7a6d19-7083-47f0-877c-919c36189a8e
8. 關於TPM，加密解密，哈希等的理解：
   https://blog.csdn.net/fengxiaocheng/article/details/103763703