壹、 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

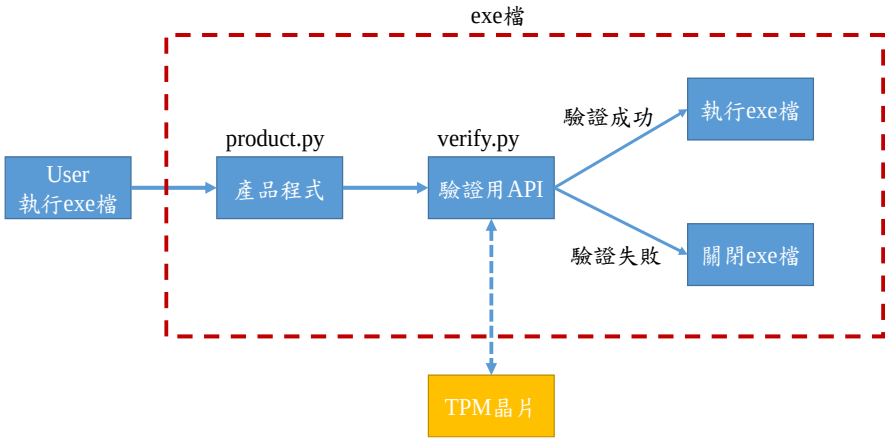
(圖 2) TPM 晶片反面圖

(圖 3) Advantech ARK-1123

(圖 4) Advantech ARK-3532

(圖 5) Advantech IPC-7132MB-50B

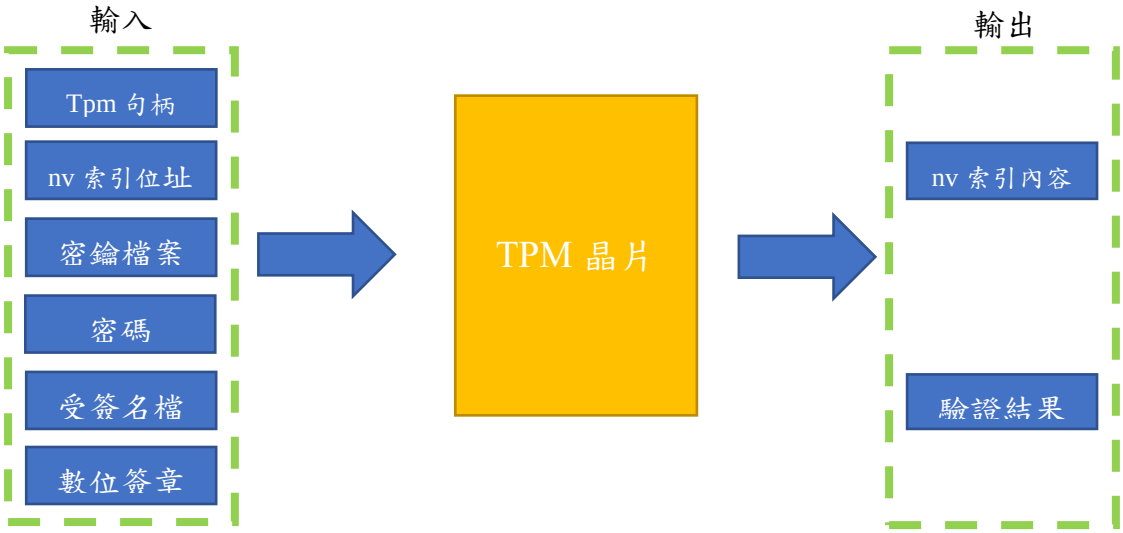
貳、 專案架構圖



(圖 6)為專案架構圖

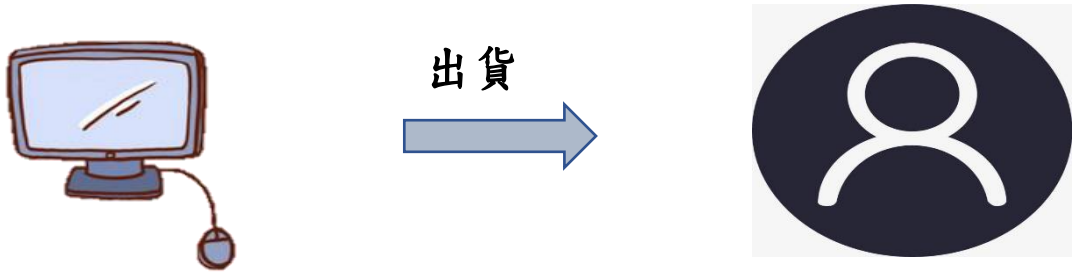
專案架構內所用到的數位簽章、金鑰以及執行動作都存在於TPM內部，於要保護用的產品程式(product.py)內呼叫模組化的程式(verify.py)，再透過模組化的程式(verify.py)對TPM晶片模組進行驗證動作，verify.py 對TPM晶片模組傳送指令，TPM晶片模組回傳結果給verify.py，verify.py使用回傳的結果判斷驗證成功與否，成功則繼續執行.exe檔，失敗則直接關閉程式，由於模組化的驗證程式，同一台設備上的程式都可使用以此達到保護、驗證效果。

參、 TPM 架構圖



(圖 7)為 TPM 輸入輸出架構圖

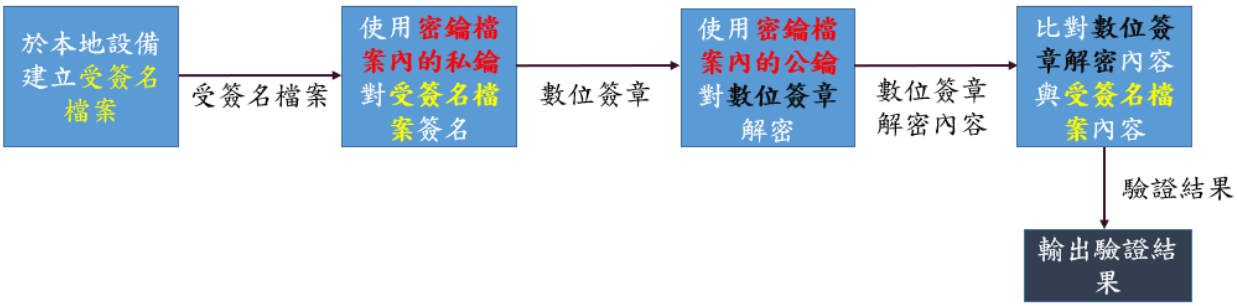
對 TPM 的操作，區分成輸入與輸出，輸入部分的金鑰以及數位簽章檔案平時儲存於 TPM 內部，透過權限以及密碼方式確保除硬體破壞外無法被其他方法提出或是破解，nv 索引位址以及密碼則是產品出貨前設定好的，於程式因為寫成了模組化，因此能自行設定及調整 nv 索引位址、密碼等輸入部分。



(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。

肆、 驗證架構圖



#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)。

伍、 優缺點分析

優點:

1.無網路環境執行

2.硬體層面保護

3.高彈性

4.不受斷電影響

5.唯一性

6.操作需要密碼

驗證流程全程都是在電腦內部執行，並且不使用任何電腦外部存取。這是因為初始設計時預設驗證流程可能是使用在無網路環境的工廠設備。專題所用到的檔案、金鑰以及執行動作都存在於TPM內部，此外，TPM內部儲存空間部份不受斷電以及電腦重開機的影響。透過程式與TPM進行操作，達到硬體設備的綁定。為了防止駭客可能嘗試跳過程式，自行對TPM進行任何破解或操作的風險，於出貨前對TPM設定綁定一組自行設定的密碼，此密碼並不會在任何情況下於外部被獲取。當駭客對TPM進行操作時，沒有正確密碼的情況下，多次失敗會導致TPM啟動DA保護，以防字典攻擊。程式部分寫成API，方便彈性使用於任何需要保護軟體的情境。

缺點:

1.驗證用檔案遺失就需要重新設定

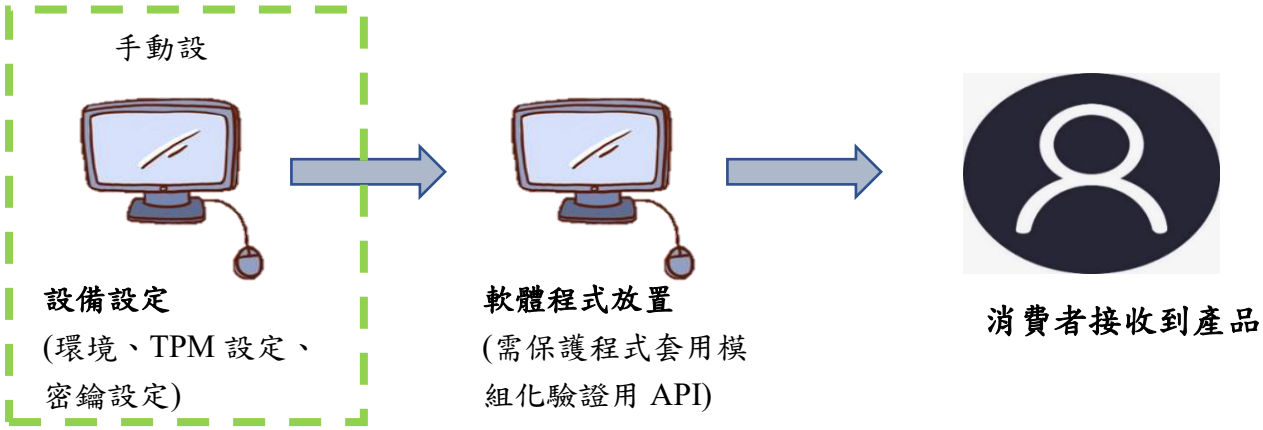
陸、 成果展示及未來展望

成果展示:

(圖 10)為成果對比圖

於兩台 IPC 上分別透過./TESTAPI 來執行該 TESTAPI.exe，各自執行兩次，兩次分別為，執行時設備內驗證用檔案為自身設備檔案(CO.dat)，以及交換各自的驗證用檔案(CO.dat)，第二次執行並無任何輸出，表示驗證失敗，即關閉程式。

未來展望:



(圖 11)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。