於NXP-MPC5748G上實現基於ECIES

混合加密系統的檔案傳輸

Implementation of hybrid encryption system on NXP-MPC5748G for file transmission

指導教授:張燕光

專題成員:吳仕群，雷智翔

開發工具: NXP Design Studio

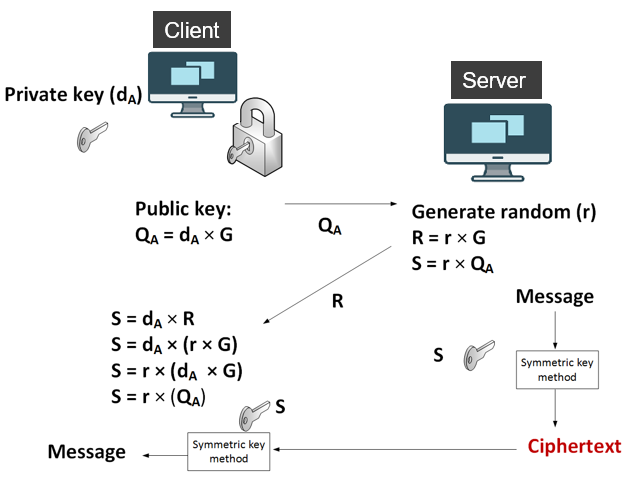
測試環境:Windows 10

**一、目的與簡介：**

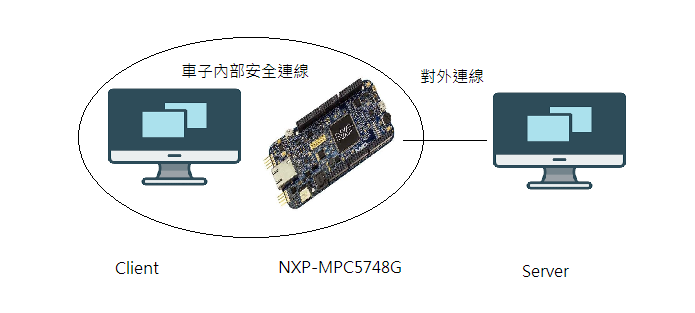
**(一) 目的:**

* 隨著汽車產業的發達，車聯網的應用和需求日漸增加，為了實現安全的資料傳輸，我們以NXP-MPC5748G開發版做為橋樑，使用ECIES作為加密系統，將傳送的封包資料作加解密。

**(二) ECIES演算法介紹:**

* ECIES的 是一種結合了ECC非對稱密碼學和對稱金鑰加密的混合加密系統，在許多的密碼學標準裡都有被提及。ECIES的優點是會使用非對稱式加密裡的公耀私鑰的概念，但又可以保持對稱式加密的運行效率。因次被廣泛應用，例如Google在Google Pay 上使用ECIES確保資料安全。
* ECIES總共包含了四個部分，橢圓曲線密碼學、金鑰衍生函式、對稱金鑰加密演算法、訊息驗證碼演算法。ECC密碼學的部分，我們使用secp256k1這條橢圓曲線，對稱加密使用AES演算法，MAC的演算法則是使用HMAC，並使用sha256做hash的雜湊。
* 金鑰產生流程:
* 
* 這個是ECIES產生金鑰的流程圖，兩端使用同樣的橢圓曲線secp256k1和同樣的基準點G。每次傳檔之前會進行一次這樣的流程更新雙方使用的金鑰，這樣可以保持加密的安全。一開始Client透過ECC產生公鑰Q和私鑰d，將公鑰Q傳給Server，Server接者亂數產生常數r 乘上基準點G計算出R後回傳給Client，然後用r 乘上Client傳來的公鑰Q得到共同密鑰S。接著Client使用自己的私鑰d 乘上Server傳來的R後也可以得到共同密鑰S。左下的運算流程可以證明Server和Client的手上的S會是相同的。S的大小為64bytes，最後我們使用SHA256演算法把S雜湊成32bytes，將前16bytes當成加密資料使用的key，用在AES加密演算法上，後16bytes當成HMAC演算法使用的key。

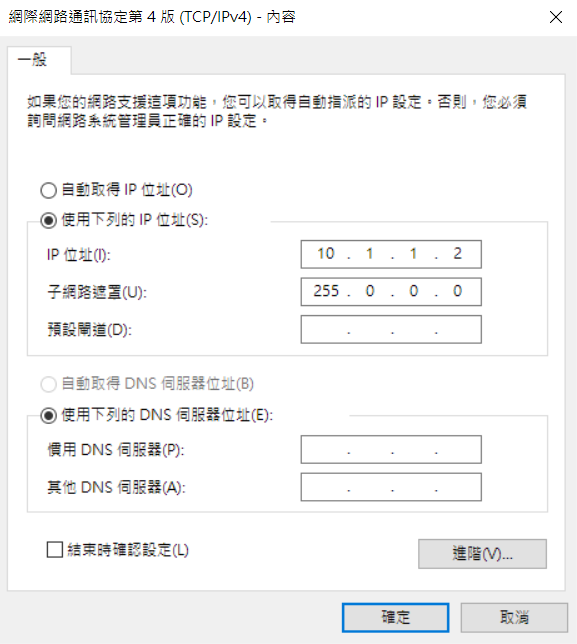
**(三) 專題架構:**

* 系統架構圖: ****
* 這是我們專題的模擬環境，左邊我們假設Client和NXP開發版是在同一部車內，兩者之間的連線是車子內部安全連線，不會被監聽或攔截，因此傳送的資料封包是不加密的。
* 車子的對外連線會使用NXP開發板作為Gateway進行連線，右邊的Server我們假設是一個在遠端的伺服器，車子和server之間為對外連線，可能會被監聽或攔截，因此傳送的資料封包會經過加密，加解密的部分會由NXP開發版進行。

**二、安裝環境:**

**1. 電腦環境**

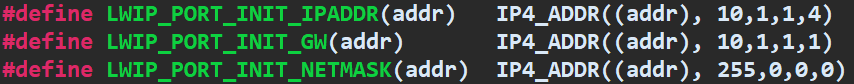
* 使用S32 Design Studio來撰寫MPC5748G開發板所需使用的程式
* 網路設定:

Server端將網路卡IP位址做設定

IP位址設成10.1.1.2，子網路遮罩設成255.0.0.0

Client端將網路卡IP位址做設定

IP位址設成10.1.1.3，子網路遮罩設成255.0.0.0

NXP內IP設定改成

**2.硬體環境**

* MPC5748G開發板
* 兩台電腦，一台當Server，一台當Client
* 網路線及相關網路設備

**三、程式碼解說**

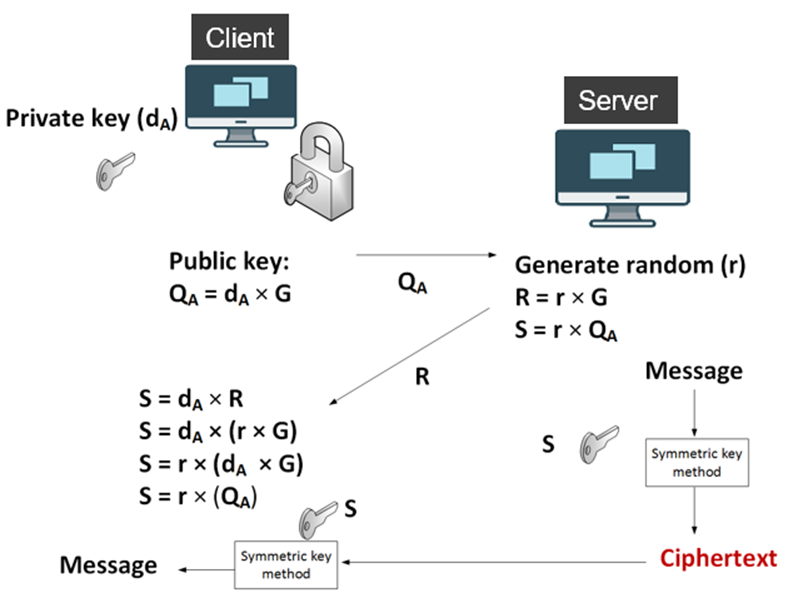
專題主要分為三個部分

* ECIES演算法
* Server、Client檔案系統
* NXP系統

**ECIES 演算法**

檔案加密是由Server端執行用Python編寫，檔案解密則是在NXP上執行用C編寫。

金鑰產生流程:



一開始Client端連上Server端後，雙方會先完成金鑰建立的部分。

**一張含有 文字 的圖片

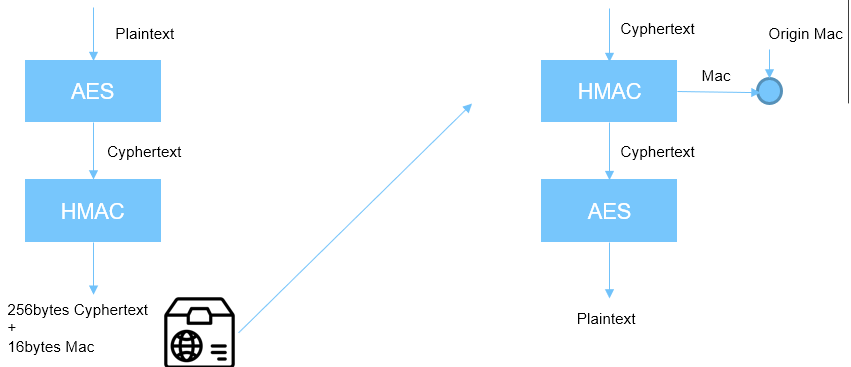
自動產生的描述**

wait\_public\_key( )這個函式在接收到公鑰後會產生亂數r，使用scalar\_mult()將r乘上基準點G產生R回傳給Client，然後再用scalar\_mult()將r乘上公鑰Q產生私鑰S，此時S的長度會是64bytes使用sha256雜湊成32bytes，然後S前16bytes做加密金鑰，後16bytes做hmac金鑰。

一張含有 文字 的圖片

自動產生的描述

Client端接收到回傳的R後，使用uECC\_shared\_secret()這個函式將R和自己的私鑰做相乘，產生和Server同樣的S，然後一樣砸湊成32bytes，前16bytes做加密金鑰，後16bytes做hmac金鑰。

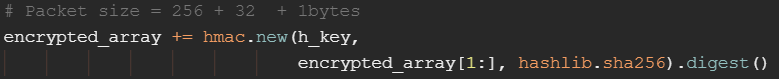


這個是檔案加解密的流程，為了符合AES演算法的規格，每次將128bits(16bytes)分成一組進行加密，然後將16組做hmac演算法，最後將256bytes Cyphertxt和16bytes Mac放在一個封包內傳送出去。解密的流程則是相反。

一張含有 文字 的圖片

自動產生的描述

AES\_encrypt()這個函式做加密，origin\_byte是要做加密的plaintxt長度為16bytes，如果小於就補零。按照AES演算法規定的格式做11輪的加密。



然後使用python的hmac函式庫中hmac()產生mac和Cyphertxt放在封包。

一張含有 文字 的圖片

自動產生的描述

Client接收到檔案封包後，會在NXP上做解密，一開始會先檢查封包的mac再做解密。Check\_hmac()這個函式會將封包內的Cyphertxt用hmac\_sha256()產生mac值hmac\_array，然後和封包內的mac值sig做比對，如果有相異就回傳1丟掉封包，相同則回傳0繼續做解密。

一張含有 文字 的圖片

自動產生的描述

Decrypt()這個函式做解密的部分，data為要進行解密的Cypthertxt長度為16bytes，依照AES演算法的規定同樣進行11輪跟加密相反的流程。

**Server、Client檔案系統**

一張含有 文字 的圖片

自動產生的描述

這裡的Sender為Server端，再連上socket後wait\_public\_key()用來產生金鑰。find\_file()會搜尋目前路徑裡的所有檔案當作檔案系統中的檔案，然後將這些檔案資訊傳給client，file\_transmission()會進入檔案傳送模式。

一張含有 文字 的圖片

自動產生的描述

這裡的receiver為Client端，連上socket後find\_file\_on\_server()會接收server傳來的檔案資訊並print出來，file\_transmission()進入檔案傳送模式後使用者可以選擇要傳送哪個檔案。

**NXP系統**

為了實現NXP作為gateway連接client、server，作法是client和server兩端都透過tcp socket連接到nxp板子，修改tcpecho.c，讓NXP可以將封包轉傳，實現client和server之間的網路通訊。使用multithread的功能讓NXP能同時控制兩個socket的傳送。

一張含有 文字 的圖片

自動產生的描述

Tcpecho\_thread\_client()這個函式會開一個tcp socket bind port 54321，然後讓client端連接上，用來控制client和NXP之間的網路連線。

一張含有 文字 的圖片

自動產生的描述

Tcpecho\_thread\_server()這個函式會開一個tcp socket bind port 12345，然後讓server端連接上，用來控制server和NXP之間的網路連線。

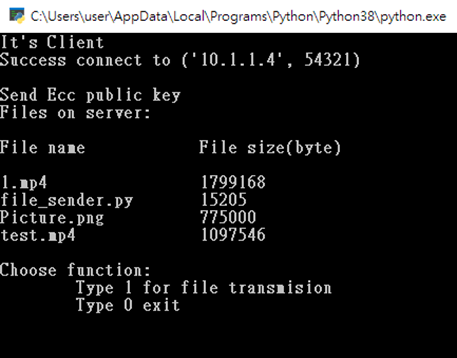
**四、測試****結果與執行畫面：**

* 執行流程:

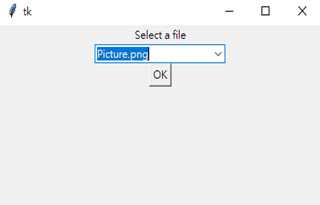
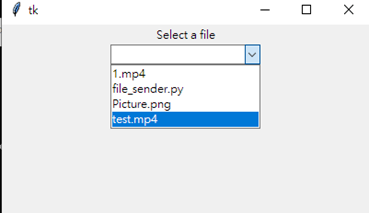
一張含有 文字 的圖片

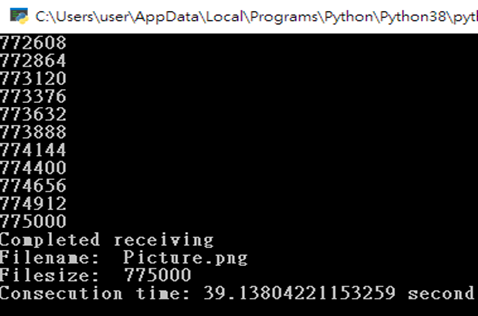
自動產生的描述

* 在兩端各自執行python檔
* Server會詢問要進行傳輸檔案或是結束，選擇檔案傳輸的選項後會等候Client端傳送Key，且回傳Server端存在的檔案資料
* Cilent端在傳送完Key之後，接收sever端的檔案訊息



* 輸入1選擇檔案傳輸，跳出選擇視窗，OK進行傳輸



* 經過一段時間的傳送，再傳送完成後會得到檔名、檔案大小、傳輸時間等資訊，並詢問是否要繼續。
* 

**五、結語：**

將ECIES加密檔案系統使用在MPC5748G上， 在未來車用電子更加普及的時候，可以保障資料傳輸的安全性。

**六、參考資料：**

<https://www.nxp.com/design/development-boards/automotive-development-platforms/mpc57xx-mcu-platforms/mpc5748g-development-board-for-secure-gateway:DEVKIT-MPC5748G>

<https://cryptobook.nakov.com/asymmetric-key-ciphers/ecies-public-key-encryption>

<https://asecuritysite.com/encryption/ecc3>

<https://github.com/hlilje/aes-python/blob/master/aes.py>

<https://gist.github.com/Yunyung/2e38fd71c3fb5967c802049514909569>