國立交通大學

資訊科學系

碩 士 論 文

系統晶片安全護衛

Security-Inside-SOC

研 究 生：薛仲佑

指導教授：何慎諾　教授

指導教授：莊仁輝　教授

中 華 民 國　九 十 四　年 六 月

系 統 晶 片 安 全 護 衛
Security-Inside-Soc

研 究 生：薛仲佑　　Student：Zhong-You Xue

指導教授：何慎諾 、莊仁輝　　Advisor：Luc Claesen、Jen-Hui Chuang

國 立 交 通 大 學

資 訊 科 學系

碩 士 論 文

A Thesis

Submitted to Institute of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

國立交通大學

研究所碩士班

論文口試委員會審定書

本校 **資訊科學** 系 **薛仲佑** 君

所提論文: 系統晶片安全護衛

Security-Inside-SOC

合於碩士資格水準、業經本委員會評審認可。

口試委員：

雷欽隆	郭斯彥
譚建民	

指導教授： 莊仁輝 何慎諾

系主任：莊仁輝

中華民國 94 年 6 月 22 日

Department of Computer and Information Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

As members of the Final Examination Committee, we certify that we have read the thesis prepared by **Zhong-You Xue**

Entitled **Security-Inside-SOC**

and recommend that it be accepted as fulfilling the thesis requirement for the Degree of Master of Science.

Chin-Laung Lei
Jiann-Liang Tan
Hsuchun Yen

Thesis Advisor: Jen-Hui Chang
Chairman: Jen-Hui Chang
Date: 2005.6.22

系統晶片安全護衛

學生：薛 仲 佑　　指導教授：何慎諾博士、莊仁輝博士

國立交通大學

資訊科學研究所

摘　要

隨著通訊技術的進步，生活中處處需要利用網際網路或其他通訊設備來交換資料。網路資料處理成為電腦系統所需解決的問題，這方面的問題在過去是由一般 CPU 來處理，而現在已發展成由專門的網路處理器(NPU)來提供高速的封包處理，以解決日亦嚴重的網路流量問題。而資訊加密的處理也跟著網路資料傳輸一樣在未來將會有越來越多的需求。因此在此篇論文中，設計出具前瞻性的安全模組晶片來加速一般 CPU 或網路處理器對加密資料的處理，並提出個人信賴裝置的概念。

PDA、手機等行動裝置加裝此安全模組晶片將成為可信賴之防竄改裝置，即個人信賴裝置。安全模組晶片中結合了對稱式密碼系統、非對稱式密碼系統、訊息摘要、簽章、亂數產生等演算法，能提供資料加解密以及認證、簽章等功能。更能進一步使個人信賴裝能在 PKI 架構下，進行認證、資料加解密及數位簽章等活動成為 PKI 架構下的具體裝置或載具。

Security-Inside-SOC

Student：Zhong-You Xue　　Advisor：Dr. Luc Claesen
Dr. Jen-Hui Chuang

Institute of Computer and Information Science

National Chiao Tung University

ABSTRACT

Along with the evolution of the communication technology, more and more data in our daily life are transferred and exchanged on internet or other communication equipments. Therefore, dealing with data on the internet becomes the problem solved in need of the computer operation system. Problems in this aspect were handled by general CPU, but nowadays the situation has changed. NPU is developed to offer high speed packet switching in order to solve the daily increasing internet flow problem. And there are more and more requirements both on information encrypted system and data transformation on internet. In this thesis, shortening the encryption time of general CPU or internet processor by foresighted secure module chip and personal trust device are proposed.

Mobile communication devices such as PDA and cellular phone will be inalterable and credible by installing the PTD (Personal Trusted Device). The secure module chip is the combination of mathematical calculations such as Symmetric Key Cryptosystem, Public Key Cryptosystem, Message Digest, signature and random number generator, and is capable to offer functions like encryption, decryption, certification and signature. Further more, the safety module chip enable the PTD to perform encryption, decryption, certification and digital signature, and become a concrete device or vehicle under PKI (Public Key Infrastructure).

致謝

首先要感謝已經遠在比利時的何老師。這篇論文因為何慎諾老師申請到國科會計劃，讓我有機會利用高功能的 FPGA 來實作本研究的密碼系統。還要感謝系主任莊仁輝博士，感謝老師在論文內容的更改與修正給了我很多的建議，也感謝口試委員顏嗣鈞教授、雷欽隆教授、譚建民教授給予我的建議與指教，讓我的論文能更加完整。最後要感謝我的家人在我多年的求學生涯所給我的支持，如今才能順利取得碩士學位，謝謝曾經給予我幫助的所有人。

目錄

圖目錄

表目錄

第一章 簡介

近來隨著半導體科技的進步，晶圓越做越大，晶片卻越做越小，越做越複雜，利用硬體方式實現加解密演算法已成為可行方案。如此一來，不僅能得到比軟體方式實現的加解密系統更安全的解決方案，速度更有大幅度的提升且能避免佔用處理器過多的資源，又能成為實現公開金鑰基礎建設（Public Key Infrastructure，PKI）、進行認證、資料加解密及數位簽章等活動的具體裝置或載具。本研究將提供系統單晶片在安全上的解決方案，這裡所提出的安全模組晶片可以以 Co-processor 方式加速密碼運算，並可以整合在同一個晶片中以節省硬體資源。

1.1 研究動機

近二十年來，資訊科技進步神速，微晶片和網路的發明，除了帶給人類無遠弗屆的想像與期盼，更使得現代生活上處處都看得到資訊科技對人類的影響；然而，資訊電子化帶給現代社會的便利，也引發了我們對資訊安全以及個人隱私權的考量。

在將來的智慧型網路環境將會越來越需要有線或無線通訊。假定利用一個公共建設和開放的通訊頻道來傳輸資料，因為懷有惡意的罪犯日益的增加而對我們造成資料被竊取的威脅，這些新型態的網路服務面臨了許多安全上的問題，如何確保交易或資訊交換的安全性，在網際網路上成為一個相當重要的課題。傳統的使用者認證機制，是透過使用者個別的識別碼

與密碼來代表使用者的合法性，並利用存放於個人電腦上的數位簽章進行資料的控管、考核與交易。隨著科技的發達、駭客技術的進步，以往藉由使用者輸入身分碼與密碼來登入系統、存取服務的方式已不再那麼安全可靠，也無法提供足夠的安全機制以確保個人資料的隱密。因此利用具有加解密功能的安全模組晶片加裝在設備中而得到可信賴的硬體設備，提供了建立安全架構的可能性，使得安全認證機制得以建立於可信賴的個人裝置之中。

本論文之安全模組晶片的概念源自於智慧卡（Smart Card）。在智慧卡的安全控管上，每張智慧卡的卡片作業系統（Card Operating System，COS）都會經過加解密演算法則規範出一套使用者身份認證機制以設定私密資料的存取權限。這套認證機制即為安全存取模組（Secure Acess Module，SAM）提供智慧卡各種安全上的控管需求，一般以軟體方式來實現，但以軟體的方式實現並不夠安全。因此本論文以硬體的方式來實現此認證機制，並加入數位信封（digital envelope）、數位簽章（digital signature）等安全機制整合在同一個安全模組晶片中以確保本文提出的個人信賴裝置（PTD）的最高安全性。

1.2 研究目的

取得即時的資訊不但能加速決策過程，還可以提昇客戶滿意度，為企業或個人帶來最大的利益價值。而在取得資訊之前，確認使用者身分與確保資訊的安全是相當重要的議題。這些驗證、授權等確保安全的程序也必

須能讓使用者隨身攜帶、方便即時驗證，滿足行動性的需求。由於微晶片技術進步，使得我們能將公認具有安全性的加解密演算法 RSA、AES、MD5，設計在單一整合晶片中，進而來提出個人信賴裝置的應用。本研究首先將目前最安全的加密方法作整理與瞭解，並實作安全加解密模組，並探討個人信賴裝置如何能真正達到行動式的使用者驗證與簽章，以確實滿足移動型使用者之需求實現行動辦公室的目的。本研究的研究目的為：

一、分析現有安全的資訊安全機制並選出適合本研究的

二、歸納出可行的安全模組晶片並實作之

三、透過這個研究來提出個人安全裝置的概念

1.3 研究架構

本論文研究架構主要內容概述如下：

第一章、　研究動機、研究目的、研究架構

第二章、　資訊安全技術文獻探討

第三章、　系統架構與系統安全模組晶片

第四章、　實作與評估

第五章、　結論與建議

第二章 資訊安全技術文獻探討

本章說明資訊安全相關的技術，包含密碼系統介紹、智慧卡（Smart Card）以及公開金鑰基礎建設（Public Key Infrastructure，PKI）。本研究的構想來源即來自於這些資訊安全相關的裝置和架構。

2.1 密碼系統的演進

隨著電腦網路快速發展，數位資訊及相關應用不斷地推陳出新，要確保數位資訊的安全性也變得相形重要，因此如何提供一個安全的秘密通訊，就是近代密碼學最主要精神所在。

一般的密碼系統有三位主角，即接收方、傳送方與破密者，密碼系統對資訊的保護，可提供下列功能：

1. 隱私性：防止破密者發現要傳達的資訊。
2. 認證性：確定資訊是由傳送方所送，而非別人偽造。
3. 完整性：確定資訊沒有被任意更改、取代或刪除。
4. 不可否認性：傳送方在事後，不可否認其傳送過的資訊。

傳送方將資訊以特殊編碼的方式來隱藏，使得破密者無法經由編碼後的訊息來得知原始資訊，只有接收方經由適當的解密方式才可還原回原始資訊，因而達到秘密通訊的目的。

一個典型的密碼系統，如圖 2.1 所示。通常將欲傳送的資訊明文

（Plaintext），以加密金鑰用一定的加密程序運算之後，所得到的密文（Cipher-text）經由通道傳送出去；而接收方將所收到的密文以解密金鑰搭配特定解密程序之後即可還原出原文。通常可將金鑰公開與否，分為秘密金鑰密碼系統（Secret Key Cryptosystem）與公開金鑰密碼系統（Public Key Cryptosystem）。

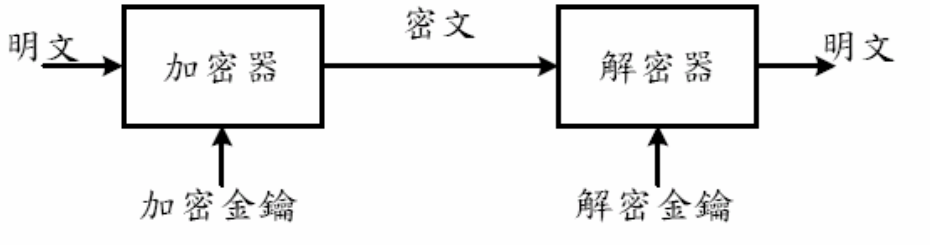

圖 2.1 典型密碼系統

2.1.1 秘密金鑰密碼系統（Secret Key Cryptosystem）

秘密金鑰密碼系統又稱為對稱式金鑰密碼系統。密碼系統中，若加密金鑰只有傳送方知道，則此系統稱為秘密金鑰密碼系統；在一般情況下，傳送方與接收方秘密的分享同一把金鑰，因此又稱為對稱式金鑰密碼系統（Symmetric Key Cryptosystem）。通常此種類型的密碼系統是利用複雜之換位與取代，最大的特點在於加密的速度快，常應用於需要即時（Real Time）或快速加解密的場合。然而，如何讓傳送方及接收方獲得金鑰，假設金鑰必須經由一個秘密通道來傳送，但在實際上並非容易做到，此即金鑰分配的問題。若通訊的對象不只一人，還需要管理其他人的金鑰，對每個使用

者而言，是相當大的負擔，即為管理金鑰的問題。同時，此系統無法達到不可否認性，由於雙方都知道彼此的金鑰，因此傳送方可以否認他先前傳送過的任何資訊，傳遞的資訊並沒有簽署的效力存在，且文件可以被任何擁有金鑰的人偽造。秘密金鑰密碼系統，如圖 2.2。

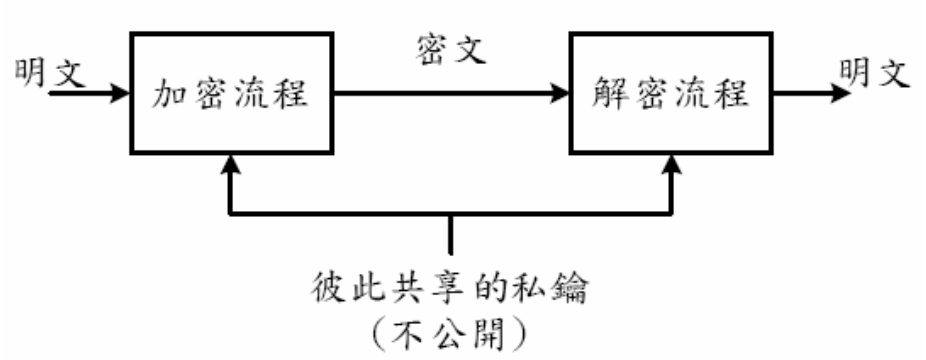

圖 2.2 秘密金鑰密碼系統

2.1.2 公開金鑰密碼系統（Public Key Cryptosystem）

公開金鑰密碼系統（Public Key Cryptosystem）有鑒於金鑰分配、金鑰管理等問題，1976 年 Diffie 和 Hellman 提出公開金鑰密碼系統的觀念。由於加密的過程中，不需要知道解密金鑰的資訊，因此公開金鑰密碼系統其加密金鑰與解密金鑰可以是不同的，故又稱為非對稱式金鑰密碼系統（Asymmetric Key Cryptosystem），其特點是加密金鑰可以被公開，故又稱為公開金鑰（Public Key），並且無法從公開金鑰得到解密金鑰的資訊，因此對於系統的安全性，又多了一份保障。傳送方利用接收方的公開金鑰對資訊加密並傳遞；接收方利用與加密金鑰不同的解密金鑰（Decryption

Key）對密文作解密，並且解密金鑰必須被接收方秘密地保存著，因此稱為私密金鑰（Private Key）。如此一來，金鑰分配問題、金鑰管理問題及不可否認性即可獲得解決：將自己的加密金鑰公佈在網路上，不但可以讓傳送方加密資訊，自己的解密金鑰亦沒有曝光之虞，解決了金鑰分配的問題；同時只要保管好自己的解密金鑰，無需管理其他人的金鑰，對使用者可說是相當方便；若將自己的解密金鑰對資料作解密的動作，即可對資訊作簽章，如此也可達到不可否認性。公開金鑰密碼系統，如圖 2.3 所示：

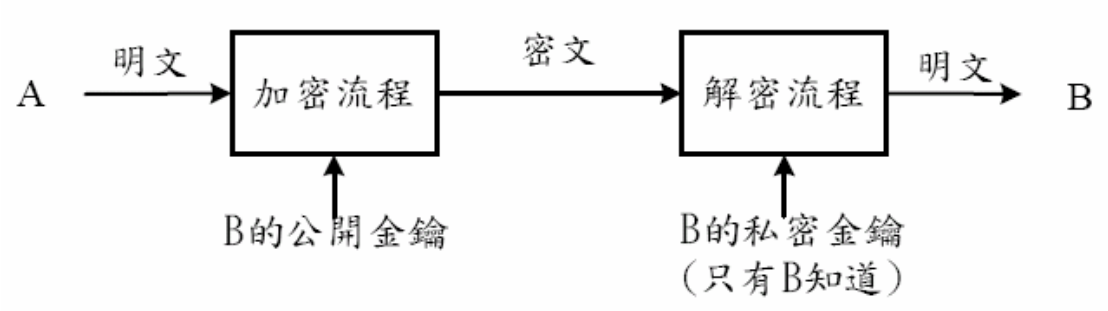

圖 2.3 公開金鑰密碼系統

2.1.3 單向雜湊函數

單向雜湊函數是 One-Way Hash Function 的縮寫。一般常見的有 MD2、MD3、MD5 以及美國國家標準技術研究院(National Institute of Standards and Technology，NIST）為配合數位簽章標準 DSA（Digital Signature Algorithm）於西元 1993 年 5 月公佈的安全雜湊函數 SHA（Secure Hash Algorithm)。

舉凡數位簽章或訊息驗證碼所使用之訊息摘要，皆由訊息透過單向雜

湊函數運算而得。單向雜湊函數的基本運作原理為輸入任意長度明文 M，經由雜湊函數運算後，皆可產生一個固定長度的輸出值-雜湊值（Hash value）H（M）。單向雜湊函數必須具有以下三項特性：

5. 就某一輸入值 M，可以快速運算產生輸出值 H（M）。
6. 無法自輸出值 H（M）回推得原輸值 M 滿足單向性。
7. 兩不同輸入值 M 與 M’ 經由相同的單向雜湊函數運算並不會產生相同的輸出值 H（M），以滿足碰撞（Collision）抵抗性。

雜湊函數應用十分廣泛，一般作為驗證通訊傳輸之訊息的完整性。接收方自傳送方取後得通訊資料，接收方只需將訊息進行單向雜湊運算，以產生的結果與傳送方給予之輸出值做比對，即可確認訊息於傳輸過程中是否遭受竄改，亦可防止網路傳輸品質不佳所引起之錯誤發生。

一般而言，密碼學上最普遍使用的單向雜湊函數包含 MD5 與 SHA。MD5 是美國麻省理工學院 Rivest 學者所設計，可將任意長度的訊息輸入值經運算後產生 128 Bytes 的訊息摘要，SHA 則為美國國家標準技術研究院為了配合數位簽章標準所設計，其可將 512 位元區塊的訊息經運算產生 160 位元的摘要。

2.1.4 數位簽章（Digital Signature）

在一般的書面文件我們可採用簽名蓋章的方式來證明此份文件的真實性。但在數位資訊中為了防止有心人士的蓄意偽造，可採用數位簽章的技

術。在公開金鑰密碼系統中，只有傳送者才擁有自己的解密金鑰，若傳送者先用解密金鑰對文件做解密運算得到簽章（Signature），之後以對方的公開金鑰作加密並傳送出去。接收者必須先對收到的資訊作解密，再到公開位置取得傳送者的公開加密金鑰之後與簽署文做加密運算，因此可得到原始資訊及傳送方的簽章。如圖 2.4 所示：

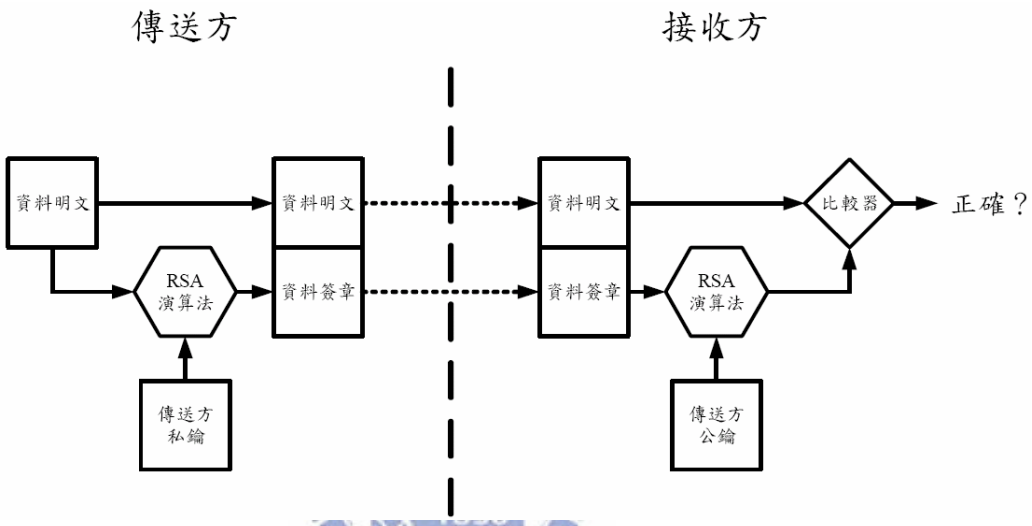

圖 2.4 數位簽章

傳送方不能夠否認先前傳送過的資訊，因為沒有人能用同樣的解密金鑰對文件作簽章；接收方可以由對方的公開金鑰來確認此文件是由傳送方所發出的。並且接收方也無法偽造傳送方所傳送的資訊，因為無法得到相同的簽章。而數位簽章通常具有以下的幾個特點：

1. 數位簽章必須是數位化的信號，且必須是與被簽章文件相關。
2. 數位簽章時必須使用簽章者獨一無二的秘密金鑰。
3. 對任何文件，簽章者應很容易產生其數位簽章。
4. 任何人應很容易驗證文件及其數位簽章之合法性。
5. 數位簽章應容易儲存，且不佔用大量記憶體空間。

6. 任何人均無法偽造數位簽章。

圖 2.4 的系統仍然存在著非對稱式密碼系統的缺點：速度太慢。為了改進這個缺點，我們可以使用訊息摘要（Message Digest）的技巧將需加密的資料長度減小，我們可以使用雜湊函數（hash function）的方式來達到減小資料長度的目的。現行著名的雜湊函數有 MD5 及 SHA-1 兩種。如圖 2.5 所示。被簽署之文件須先經過 SHA-1(或 MD5)的壓縮處理，產生 160 位元的訊息摘要(Message Digest)。簽章者再以其 RSA 私密金鑰對這份訊息摘要加以簽章，產生之數位簽章連同原先之明文傳送出去，以便讓接收方或是第三者來驗證。至於驗證的示意圖則如圖 2.6 所示，驗證端一方面以簽章者的公開金鑰將數位簽章解開，而在另一方面，他同時也把傳來之明文以單向雜湊函數處理。接著將這兩個值進行比對，比對無誤表示驗證成功。

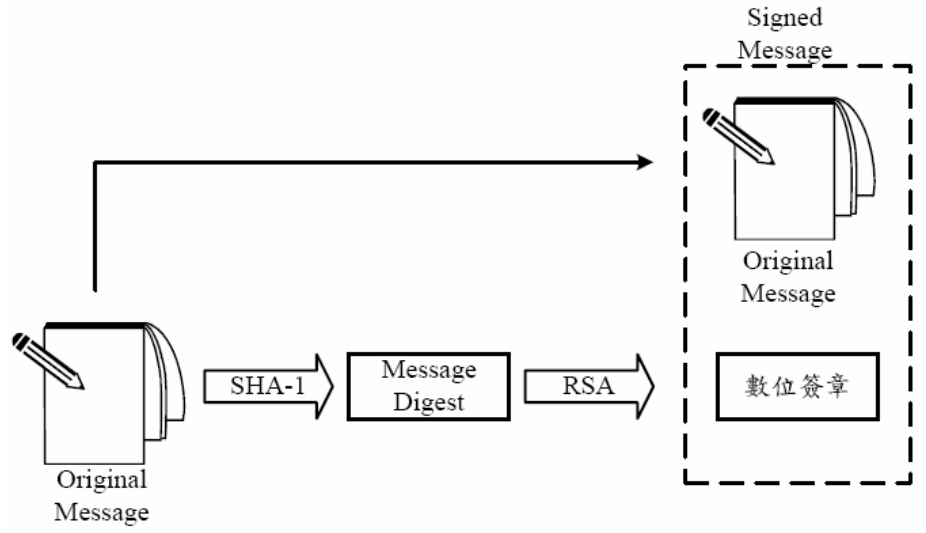

圖 2.5 數位簽章之產生

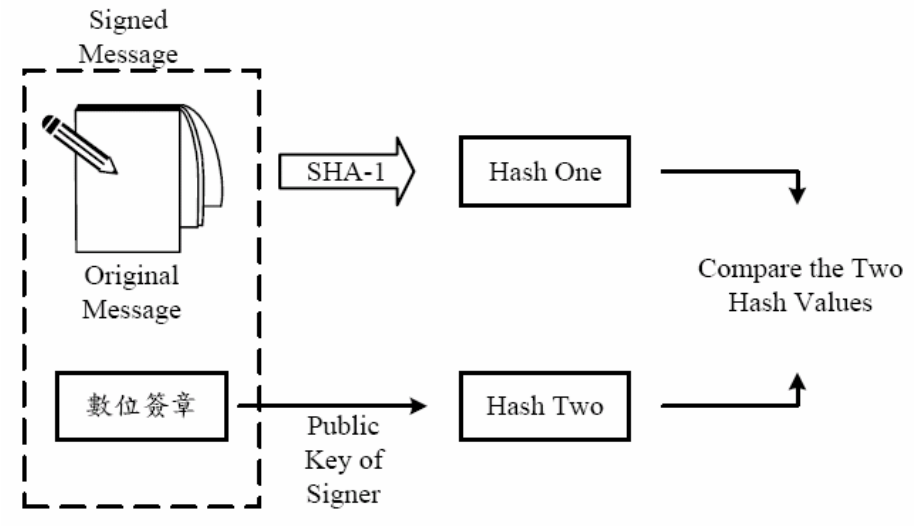

圖 2.6 數位簽章之驗證

2.2 智慧卡簡介

表 2.1 智慧卡 2000-2004 年成長率

地區	**整合計畫成長率**
歐洲、中東、非洲	29%
亞太地區	20%
日本	14%
北美	23%
美洲	14%

近年來智慧卡展業發展快速。由於智慧卡具有嵌入式的運算能力、安全性、可攜性與便性等優點，使得智慧卡廣泛應用於各種不同的領域中，

例如：健保卡、電子商務、通訊終端機、銀行電子交易、身份識別系統、收費系統、門禁管制系統等，皆是智慧卡應用成功的範例。智慧卡的發展已成一股不可抗拒的熱潮,且將成為下一世代最普及的個人隨身資訊產品。

2.2.1 智慧卡的優點

表 2.2 智慧卡內部單元及功能

	類別	功能
微處理器單元（MPU）	CPU	控制處理單元，用來解釋並執行COS所下的指令
	ROM	唯讀記憶體，用來控制程式、確保程式的安全
	RAM	隨機存取記憶體，用來存放資料運算處理過程中的臨時訊息，為運算、邏輯處理的工作區
	I/O	讀寫設備之輸出入介面處理
記憶體（MEMORY）	EPROM	可消除可程式唯讀記憶體，資料記錄區無法利用電流重寫資料
	EEPROM	電流可覆寫可程式唯讀記憶體，資料記錄區可利用電流重寫資料

智慧卡能夠取代磁條卡，發揮各領域的應用效應，乃是由於目前的磁條卡一般只能容納 72 Bytes，而智慧卡則可以儲存 8000 Bytes 資料，並且可以容易的寫入或消除資料，達到記載每筆交易資料的記錄功能。透過智

慧卡內之微處理機功能及邏輯處理電路的應用，智慧卡能對記憶體資料設定各種不同的存取控制，也就是針對不同的使用者對智慧卡內部不同的資料，給予不同的讀取或寫入權限：並能根據內部預設邏輯，來判斷允許外部系統介入的層次，例如連線交易。而安全性佳更是智慧卡優於其他媒介的主因之一。智慧卡的安全保護具計有卡片真偽鑑別（Authentication）、持卡人識別（Identification）、資料存取安全（Secure Read/Write）、資料傳輸安全（Encryption）以及交易認證（Certification）。透過多層的安全保護措施，使得智慧卡被偽造的機率極低。其資料安全措施共計四項：

A. 內藏 DES、RSA、雜湊函數等演算法
B. 智慧卡可驗證個人密碼（PIN：Personal Identification Nunber）達成持卡人身份認證之功能。
C. 配合終端設備，利用 DES、RSA 等安全系統由 Challenge/Reply Protocal（詢問/答覆協定）達成終端認證智慧卡，智慧卡認證終端之功能。
D. 智慧卡可利用 DES、RSA 等安全系統、對重要的交易產生電子簽章，作為交易授權的憑證。

2.2.2 智慧卡之應用

智慧卡常應用在在電子錢包、個人數位證書、個人秘密金鑰、或與網際網路線上交易結合等。電子錢包（Electronic Wallet），原本是指安裝在客戶端電腦上，並符合 SET（Secure Electronic Transaction）規格的

軟體。在 SET 文件中並沒有指定電子錢包發展的雛型。而 IC 卡型的電子錢包是讓消費者能夠藉由連結到個人電腦的讀卡機，使用卡片直接在網路上從事 SET 交易。這張電子錢包卡片必須能夠儲存符合 SET 規格的數位證書，並能夠對交易的重要資料做簽章加密的動作，是屬於先購物再付款的機制。

個人數位證書內含有公開的個人資料以及一把個人的公開金鑰，可利用智慧卡將個人數位證書儲存在內，然後利用這張卡片將數位證書傳送給他人，以證明自己的身分或是證實自己的公開金鑰是正確的，而他人亦可以利用這把公開金鑰將資料傳送給持卡者，以應用於 SET 或是 SSL（Secure Socket Layer）的協定。

個人秘密金鑰內含有非對稱式密碼器的秘密金鑰，或是對稱式密碼器的通訊金鑰，要取出金鑰前必須先輸入密碼，將個人秘密金鑰應用在智慧卡一般是透過個人電腦的讀卡機連上網路，在需要秘密金鑰時在鍵盤上鍵入密碼，為了安全起見，也可直接使用 Super Smart Card，直接在卡片上的 Keypad 鍵入密碼，以防止由個人電腦上不當的程式擷取密碼。

至於與網際網路線上交易之結合則是將智慧卡與付款機制相結合。如 1998 年由新加坡國立大學（The National University of Singapore）、渣打銀行（Standard Chartered Bank）及 VISA 國際組織共同推出的新標準 Smart Electronic Loyalty E-Commerce Transaction（SELECT），可以在卡片中儲存 SET 規格的數位證書，利用此一卡片可直接在網路上從事 SET 交易。

2.2.3 智慧卡之應用架構

智慧卡的應用實體架構包含 HOST、IFD（Interface Device）、智慧卡三個部分以及 SAM，如圖 2.7 所示。

A. 智慧卡：在 F.I.S.C.的規格裡，智慧卡可僅具磁條或可同時兼備 IC 晶片與磁條。卡片內含有卡片作業系統 COS（Card Operation System），處理卡片內部資料之管理與外界通訊的控管。

B. IFD：一般指讀卡機，包括卡片讀寫設備 R/W（Reader/Writer）、密碼輸入器（PIN Pad）以及資料安全模組 SM（Security Module）。

C. HOST：負責執行 IC 卡之應用程式，藉著各種通訊程式，HOST 透過 IFD 將指令送到卡片，反之 HOST 經由 IFD 得到 IC 卡回應的訊息。

D. SAM：安全存取模組在下一節加以介紹。

其中 IFD 可以為單機型讀卡機，本身具有螢幕鍵盤可直接在卡機上作業；若是從 HOST（PC）端直接下指令給 IFD，則為 PC/SC（Personal Computer/Smart Card）之基本架構。

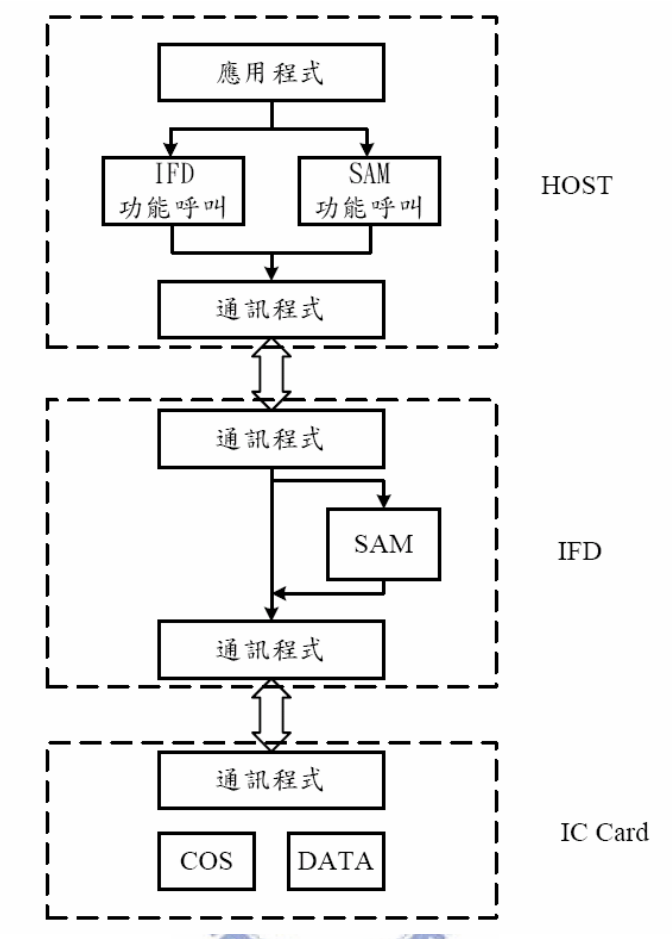

圖 2.7 智慧卡應用架構

2.2.4 安全存取模組（Secure Access Module，SAM）

SAM 是提供智慧卡各種安全上的控管需求,如智慧卡上有些資料是私密非公開的，所以每張卡片的作業系統都會通過加解密運算法則規範出一套使用者身份認證機制來設定私密資料的存取權限，而我們就必須以軟體或硬體把這認證機制實現出來，而這個軟體或硬體就稱之為 SAM。換句話說若想存取智慧卡上受保護的資料，但你沒有存取這份資料的權根，自然無法

通過 SAM 的檢查而被檔下，無法存取資料；或者根本沒依照智慧卡作業系統規範以軟體或硬體做出 SAM，那連認證動作都沒辦法進行，自然無法存取受保護的資料。一般來說，SAM 用下列三種方式實現：

A. 以軟體方式實現之：這是最常的方式，一般我們都會把 SAM 功能寫進讀卡機的應用程式函式庫中，以供發展智慧卡之程式設計師呼叫使用。這種方式的缺點就是不夠安全，有被人盜用之虞。

B. 以硬體方式實現之：將 SAM 認證機制做在讀卡機的晶片中，這種處理速度不但快而且是最安全的，因為只有這台讀卡機可以存取同一 SAM 認證的智慧卡資料。但此種方式無法讓智慧卡在市面上廣為流通，因每家智慧卡及讀卡機製造商的 SAM 認證機制無法統一。

C. 以智慧卡方式實現之：鑒於用軟體做 SAM 不夠安全，在讀卡機做 SAM 又不符經濟效益，於是把智慧卡加上 SAM 認證機制，稱之為 SAM 卡。也就是說智慧卡必須再配合上 SAM 卡才能存取智慧卡受保護的資料。

其中若以智慧卡方式實現 SAM，必須符合 SAM 卡之硬體架構為單一晶片之保護系統保護，內含運算法則、安全模組功能、主基碼、安全模組序號等。至少一組主基碼 8 Bytes 與安全模組序號 8 Bytes 之運算而在基碼的儲存空間規劃中，至少可以存放一個發卡單位之基碼，並將基碼資料檔存放於可重寫記憶體內。並且主基碼與安全模組序號在未供電的情況下，需維持至少三年資料不流失。資料運算必須符合 ANSI X3.92 的 DES 演算法，同時在處理 SAM 之運算時，所有的基碼均受保護。

2.3 公開金鑰基礎建設（Public Key Infrastructure，PKI）

為了要確保資料傳輸的隱私性及安全性，加密系統成為必備的輔助工具。早期的加密系統採用對稱式加密法，也就是加密金鑰與解密金鑰是完全相同，抑或可以相互推算。即使對稱式演算法不斷地推陳出新，由 DES 到 3DES，再到安全性極高的 AES，但卻無法解決兩個最根本的問題，那就是如何安全地將金鑰告訴對方，以及後續的金鑰安全管理及保存問題。

為解決上述問題，遂有非對稱式加密系統（又稱公開金鑰系統）的問世，而 PKI 公鑰基礎建設即為採用該加密系統理論，並用於網路資料加密及數位簽名等服務的統一架構。該架構下的使用者同時擁有一對成雙的公鑰及私鑰，私鑰只有使用者自己才有，所以理論上具有獨一無二的極度安全性，公鑰則對外公開給任何人。所以只要用對方的公鑰（任何人皆可獲得）加密的資料，也只有對方的私鑰才能解得開，如此就解決了金鑰分配及安全管理的問題了。

更重要的是，PKI 是目前唯一可以同時達到身份認證、隱私性、完整性、不可否認性及存取控制等五大資通安全目標及應用的安全架構。我們將探討 PKI 組成的關鍵元素，以及這些元素所扮演的角色及重要性。

一、一對相匹配的公／私鑰

由於 PKI 是基於非對稱金鑰密碼系統所建立的，所以一雙成對的公／私鑰是最基本的元素，因為唯有透過公／私金鑰，才能去數位憑證中心，申請個人專屬的數位憑證，或透過私鑰產生數位簽章，或對資料進行加解密的動作。

二、數位憑證（Digital Certificate）

在網際網路上要如何辨識彼此的確切身份，以保護彼此的權益，對此 PKI 提供了數位憑證的機制來確認彼此的身份。數位憑證是由公正的第三方單位－憑證中心所簽發的。使用者在申請憑證之前，必須先產生一組成對的公鑰及私鑰，接著並以公鑰副本及身份證明向 CA 申請，審核通過後，CA 會以自己的私鑰對使用者提供的資訊加密並產生數位簽章。所以一隻數位憑證中包含了使用者的身份識別、公鑰及 CA 的簽章，它提供了一個具公信力、無法冒用的身份憑證。

三、憑證中心（Certificate Authority，CA）

在 PKI 架構環境中，是由政府、銀行或法人等具公信力之第三方來擔任憑證中心的角色。憑證中心可謂 PKI 架構中的核心執行機構，扮演了如同人體大腦及電腦 CPU 的重要地位，其主要任務包括數位憑證的申請、安全發送及管理，以及憑證廢止列表（CRL）、用戶憑證黑名單發佈等作業。

在網際網路環境中，即使透過公鑰及私鑰來進行資料的加解密或數位簽證作業，但仍無法確認公／私鑰使用者的確實身份，而 PKI 架構中的數位憑證機制即可解決這個問題。因為使用者必須帶著自己的公鑰去向憑證中心申請數位憑證，如此一來不但有第三方公正單位證明你的身份，同時也將你的公／私鑰與數位憑證綁在一起。網路上任一方只要透過你的公鑰去驗證數位憑證，即可確認你的真實合法身分，如此便能確保彼此的權益。至於公鑰分享的問題，在非 PKI 環境中，若要獲得別人的公鑰，只有要求對方告知一途。但在 PKI 環境中，透過數位憑證機制即可解決，因為每個

人的數位憑證中都附有專屬的公鑰。此外，也可向憑證中心查詢獲得。

目前台灣的政府公開金鑰基礎建設(GPKI)，乃採用階層式憑證管理架構，最上層中心是隸屬於行政院研考會的政府憑證總管理中心(Government Root CA；GRCA)，其直屬下轄有五大第一級憑證機構，包括電子工商憑證管理中心（MOEACA）及內政部憑證管理中心（MOICA)，前者負責公司行號憑證之簽發，後者主要負責大家耳熟能詳的自人然憑證的發放管理作業。原本公司行號、財團法人及自然人憑證都是由政府憑證管理中心（GCA）總管簽發工作，如今已陸續移轉到其他專責憑證中心，見表 2.3。

表 2.3 政府各憑證管理單位表

憑證中心	英文簡稱	簽發的憑證種類	主管機關
政府憑證總管理中心	GRCA	GRCA 自簽憑證 CA 交互認證憑證	研考會
政府憑證管理中心	GCA	政府機關(構)單位憑證 伺服器應用軟體憑證	研考會
政府測試憑證管理中心	GTestCA	GPKI 技術規範所列的各種一般用戶憑證	研考會
電子工商憑證管理中心	MOEACA	公司、分公司及商號憑證	經濟部
內政部憑證管理中心	MOICA	個人憑證	內政部
組織及團體憑證管理中心	XCA	學校、財團、社團法人及非法人團體憑證	研考會

四、憑證註冊申請中心（Registration Authority，RA）

基本上 RA 是 CA 隸屬下的一部分，為憑證簽發及管理的延伸機構，主要負責數位憑證的申請註冊、審核、簽發及後續管理等業務。

五、X500/LDAP 目錄伺服器

輕巧型目錄存取通訊協定（Lightweight Directory Access Protocol；LDAP）主要針對早期 X500 目錄存取協定的複雜性及開發上的難度等問題，而在資訊、命名、功能及安全等模型上做了許多程度上的改善。其具備高效率查詢功能、樹狀資訊管理模式、分散式部署架構，以及靈活的存取控制等特點，所以已然成為網際網路目錄服務的通用標準。在 PKI 體系中，LDAP 被廣泛應用於憑證管理發佈、授權管理、憑證廢止列表（Certificate Revocation List；CRL）資訊發佈等方面。換句話說，用戶可以採用 LDAP 協定上網查詢自己憑證、他人公鑰或黑名單資訊等。

六、金鑰和憑證的更新機制

憑證是有時效性的（例如國內工商憑證有效期限為五年），如此可降低金鑰被破譯的可能性藉此提昇憑證的安全性。憑證到期後，PKI 系統多半會自動完成更新動作。但政府機構簽發憑證若到期則需要重新申請辦理。用戶的憑證、私鑰或認證裝置（如 USB Token、智慧卡）難免會有遺失的情形發生，這時可向 CA 憑證中心申請憑證廢止作業，並重申請全新的憑證。懷疑憑證或私鑰被他人破譯，或一時找不到憑證，亦可申請憑證暫停作業，若憑證失而復得便可再向 CA 申請續用恢復作業。有 CA 提供了定期更新、廢止、暫停、恢復等機制，用戶得以安心地享受認證、加密及簽章的好處。

七、CRL 憑證廢止清冊發佈機制

CA 會將過期或作廢的憑證「黑名單」，定期公佈在 CRL 憑證廢止清冊上，以供用戶線上交易查詢之用，確保線上交易的安全。

八、憑證歷史檔案庫

憑證庫是網上的公共資訊庫，可供用戶進行開放式查詢作業，用戶可藉此獲得公鑰，或查詢某憑證是否已過期或作廢。

九、認證裝置

所謂認證裝置也就是實現 PKI、進行認證、資料加解密及數位簽章等活動的具體裝置或載具，目前最常見的認證裝置有智慧卡（Smart Card）、USB Smart Token（或稱 USB Key），以及軟碟片，不同的認證裝置各有優缺點。早先 PKI 系統多半只有軟體，所以加解密動作會耗費主機 CPU 資源，若改以硬體裝置中的晶片來進行加解密就比較快捷有效率，同時硬體認證裝置也解決了憑證及私鑰安全地保存及隨身攜帶的問題。

十、交叉認證（Cross-Authentication）

如果要和日本某個公司或個人進行線上交易，但由於兩國各屬不同的 PKI 網域，你要如何確認對方的合法身分，以及所持有的憑證是否有效（過期或作廢與否），而交叉認證就是為了解決這個問題而誕生的解決機制。目前為了因應各式不同的 PKI 認證架構及型態，所以有許多不同的交叉認證模型，常見的有樹狀認證模型、網狀認證模型、橋接式模型、信任列表模型、相互認可模型、鑒定認可模型、代理路徑查詢與驗證模型等。

第三章　系統架構與安全模組晶片

爲了解決企業內部資料安全問題並提供使用者真正的行動式安全服務、落實辦公室行動化的理念，本研究構想將行動裝置(如手機、PDA）加裝此安全模組成為可信賴之個人裝置如圖 3.1 所示。

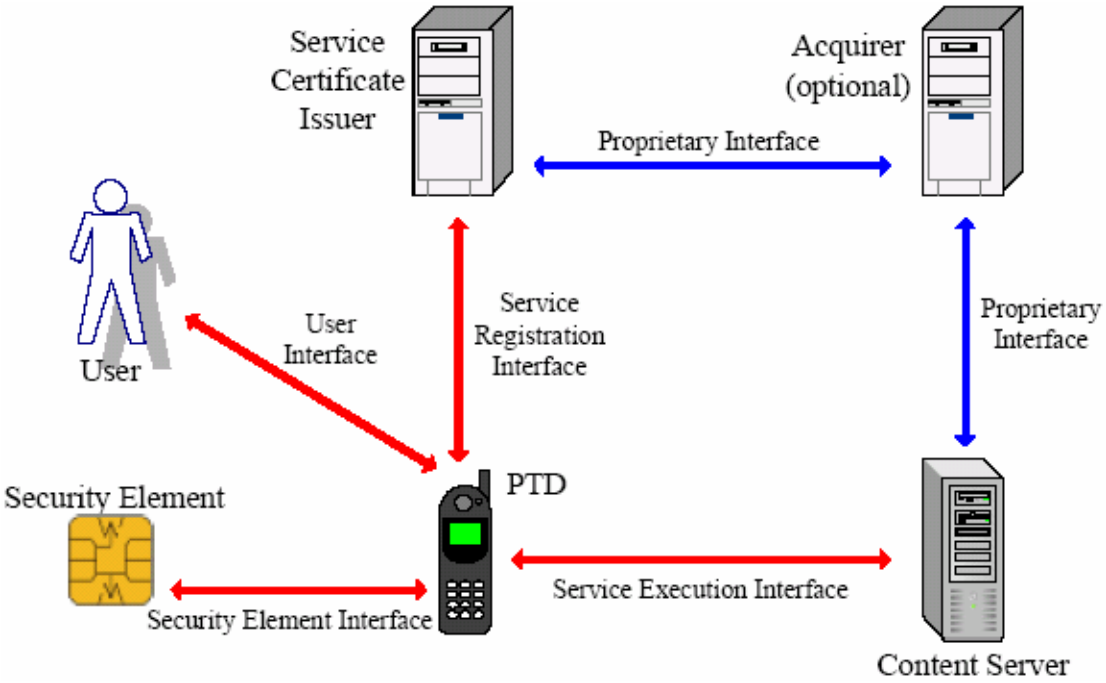

圖 3.1 個人信賴裝置

個人信賴裝置（PTD）的概念為：個人化，由某一人來控制和使用且大部分時間由此人所攜帶。其內部包括一個安全元件，用來保護重要的資料，例如私密金鑰且必須具有數位簽章以確保交易的安全功能，安全連線、認證和授權。並擁有一個應用平台，提供交易的使用者介面，如銀行、付款、票務服務。。

PTD 會用來作使用者的驗證，已驗證 PTD 所屬的使用者。在通過使用者

驗證之後，PTD 才可以被用來作交易。安全元件執行使用者驗證，例如，一張晶片卡，只有在收到使用者輸入一組特定的 PIN 碼才能開始運作，PTD 也可採用此種模式。為了能夠存取多種服務，PTD 使用一個憑證資料庫，裡頭存放特定的服務憑證。此資料庫裡可能有實際的憑證或關於實際的憑證存放地點的指標。此外，PTD 可能包括一個交易資料庫，合約和收據、票券，在此架構下透過相關安全機制進行資料的存取、身分認證與對遠端文件從事數位簽章。

3.1 安全模組晶片

本研究中認為最佳之系統安全性應為由硬體方式實現加解密存取模組，如此不僅能加速更能有比軟體為佳的保護。此安全加解密晶片的提出，使得存取資源與資料交換時有了一個完善的保護措施，可確保公司與使用者的隱私、資料的安全，並利用數位簽章達到限制組織內部裝置存取的功能。在現今 PDA、行動電話及各種行動裝置紛紛出現與越來越普及的同時，雖然其具備體積小、方便攜帶的特性，但只要透過適當的設備，就可以取得其中的資訊，所以更需要將此安全控制晶片用來保護需要更高安全性的裝置，例如是 PDA 上的應用之類或是像是 WLAN、Bluetooth 這類需要交換資訊又有暴露危機的連接方式當能考慮以此安全晶片來作保護。在本研究中將此類需要高安全性的個人裝置稱做個人信賴裝置（PTD），安全模組晶片更是個人信賴裝置的安全核心以此來確保行動裝置之安全環境。

個人信賴裝置構想源自於結合安全模組晶片與行動裝置的優點，實作

內含加解密運算及數位簽章等功能的安全晶片的構想則來自智慧卡中少部份以硬體實現的安全存取模組（SAM）。安全模組晶片與行動裝置結合的應用更希望能用於 PKI 之認證讓不管是資料傳輸或是網路交易都能在 PTD 中實現，具備實用性及安全性。企業或組織也能借此一架構避免內部資料被非授權人士竊取的問題，以達到建構安全行動辦公室環境的目標。

在綜合了上一章所述之各種資訊保密技術後，我們將提出一個包含完整加解密系統的安全晶片架構並依照晶片設計流程，利用 FPGA 實驗板加以實作。在此晶片中包含了三種系統：對稱式密碼系統，非對稱式密碼系統以及訊息摘要。圖 3.2 將表示利用這三種系統將資料明文加密之過程、圖 3.3 則表示利用這三種系統將資料明文解密之過程。

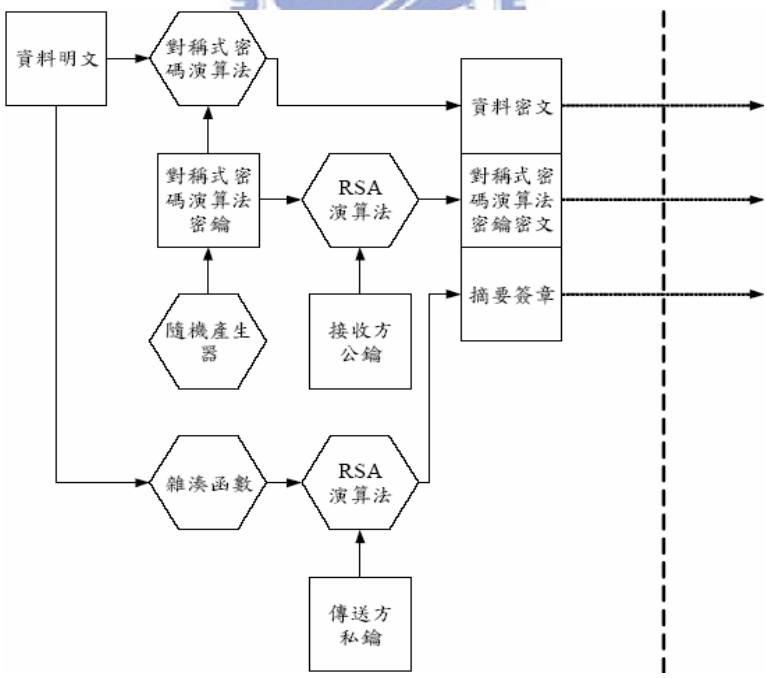

圖 3.2 安全模組晶片加密過程

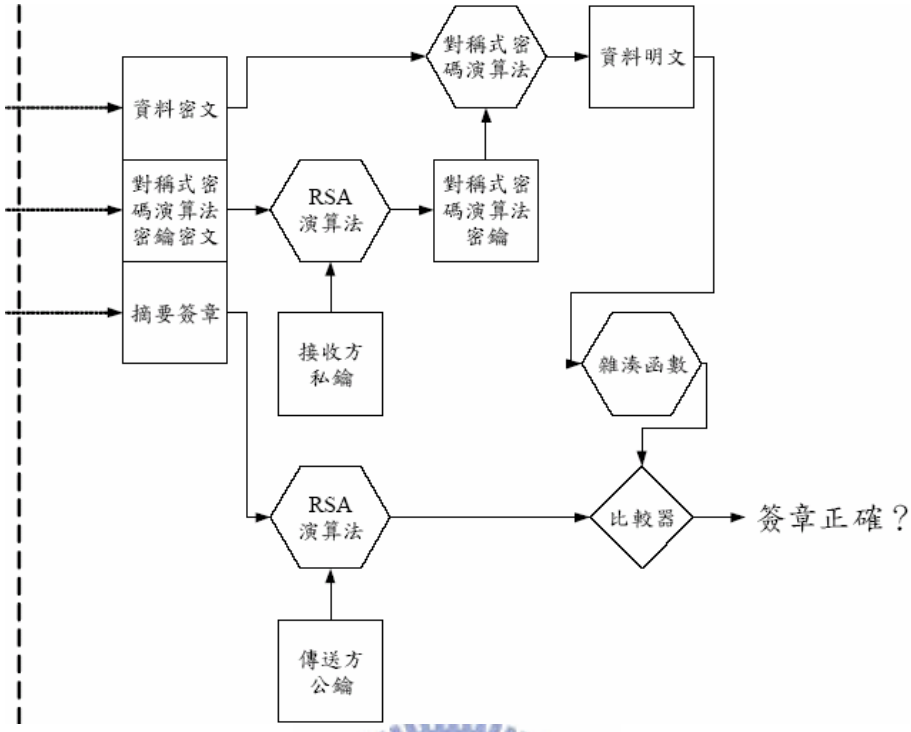

圖 3.3 安全模組晶片解密過程

其中對稱式密碼系統採用 AES 演算法；非對稱式密碼系統採用為現行最著名的 RSA 演算法；訊息摘要則採用 MD5 演算法；亂數產生則使用 LFSR 的方法。由 OPENCORES 網站取得這些免費 IP 後加以改寫為適用本研究的。

實際資料進行加密的其實是 AES 系統，而公開金鑰加密技術 RSA 演算法只針對 AES 之金鑰作加解密的動作。相對於實際資料（明文）的長度，AES 金鑰的長度相對的較短，因此對整體的效率而言，利用 RSA 對 AES 金鑰進行加密，並不會造成太大的影響，這就是所謂結合祕密金鑰與公開金鑰的資料加密方法。這樣的方法主要的優點為：實際加密資料的是計算速度較快的 AES 演算法，而且因為使用公開金鑰演算法對 AES 金鑰進行加密，因此整體系統不需要另外透過安全的管道傳送 AES 金鑰。這將是一個兼具效率與安全的資料保密方法。此外，為了提高資料傳輸的安全性，每次加密的過程中，可以更換不同的 AES 金鑰，藉以降低 AES 金鑰洩漏的風險。

3.2 安全模組晶片次模組

本研究欲實作之安全模組晶片使用到五個次模組分別為：U_prng、U_aes、U_iaes、U_rsa、U_md5，其分別使用到利用 LFSR 方法的亂數產生器、AES 之加密演算法、AES 之解密演算法、RSA 演算法、MD5 訊息摘要演算法。在本節將簡介各次模組並說明其矽智財（Silicon Intellectual Property）之來源。

3.2.1 虛擬亂數產生器（Pseudo Random Number Generator，PRNG）

虛擬亂數產生器 PRNG（Pseudo Random Number Generator）在對稱式密碼器的金鑰產生上，扮演一個十分重要的角色，用來達到隨機亂度的最佳策略。一般常見的亂數產生器有下列幾種：

1. 線性反饋移位暫存器(Linear Feedback Shift Register，LFSR)
2. 線性同餘產生器(Linear Congruence Generator，LCG)
3. 非線性亂數產生器
4. 截切亂數產生器
5. 利用數學方法之亂數產生器

在本研究中所採用的是線性反饋移位暫存器(Linear Feedback Shift Register，LFSR)。線性反饋移位暫存器是許多金鑰位元串產生器的基本要

件。因為線性反饋移位暫存器它們能夠產生周期很大的金鑰位元串且所產生的金鑰位元串有很好的統計特性。另外由於它們的結構關係使得能夠用代數特性來輕易的分析。線性反饋移位暫存器也非常適合於硬體方面的應用，故本研究以此方法來產生所需要的虛擬亂數。

我們的亂數產生器模組為 Xilinx CoreGen 之免費 IP，以 LFSR 方式完成 Pseudo Random Number Generator。藉此產生 session key。本研究所使用次模組 U_prng 之輸出與出入介面如下圖 3.4。

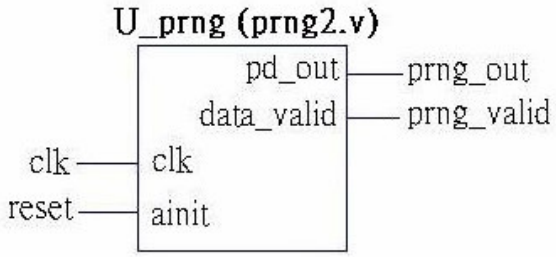

圖 3.4 U_prng 輸入與輸出訊號

3.2.2 高等加密標準演算法（Advanced Encryption Standard）

在現今網路上傳輸使用之對稱式加密系統許多仍採用 DES 或 Triple DES 來進行加解密。DES 具有金鑰大小過短等等不敷現在大量資料加密使用的缺點，Triple DES 雖然夠安全但加解密速度都需要較長的時間。由於 AES 的運算速度及安全性都比 Triple DES 好，且已被美國政府訂為國家標準來取代 Triple DES，故本研究中所採用之對稱式密碼系統選用 AES 演算法。

AES 演算法是由美國國家標準與技術協會（National Institute of Standards and Technology，NIST）與工業界與許多密碼學研究團體合作，

計劃發展與制定的。美國國家標準與技術協會廣邀全球密碼學研究團體進行相關的研究與發展時，提出幾點 AES 所必須符合的要求：

A. 經由公開的程序對外徵求。

B. 是對稱性的金鑰加密法。

C. 祕密金鑰的長度是可變的。

D. 可以同時由硬體及軟體來實作。

E. 無專利的限制或必須符合 ANSI 的專利政策，可以自由使用。

其中 Rijndael 演算法經過美國國家標準與技術協會的驗證與篩選後脫穎而出，成為 AES 演算法的標準。由於中選的候選密碼器放棄其版權，可全世界自由使用。Rijndael 之提出者也提供檢測值與參考原始碼，提供分析和描述及三種不同語言版本的軟體實現(reference and optimized in C，optimized in Java)以協助實作。

Rijndael 演算法是由 Joan Daemen 和 Vincent Rijmen 共同發明，為一區塊加密演算法（block cipher），其加密區塊與金鑰長度允許變動。Rijndael 演算法可使用 128 位元、192 位元或 256 位元長度的金鑰，對於 128 位元、192 位元或 256 位元的加密區塊進行加密（金鑰跟加密區塊長度不需要相等共九種組合）。特別是 Rijndael 不論用於回授或非回授模式，它在各種不同之硬體或軟體計算環境下，皆表現非常出色。它的鑰匙安裝(key setup)時間是最短的，而鑰匙快算(key agility)之彈性也相當好。Rijndael 所需記憶空間很少，極適合於空間限制型之環境，如 IC 卡，經測試也確實展示出其優秀性能。Rijndael 之運作是屬於較易防制電源及時序攻擊者，它可用一些防禦措施來對抗這些攻擊，而不會對其性能有明顯影

響。以下本文將 Rijndael 演算法統一以 AES 演算法代稱。

本研究採用 128 位元長度的金鑰，對於 128 位元的加密區塊進行加密，整個 AES 演算法包含了三大部份：Key Expansion 電路以及加密、解密運算電路。全部電路如圖 3.5 所示。

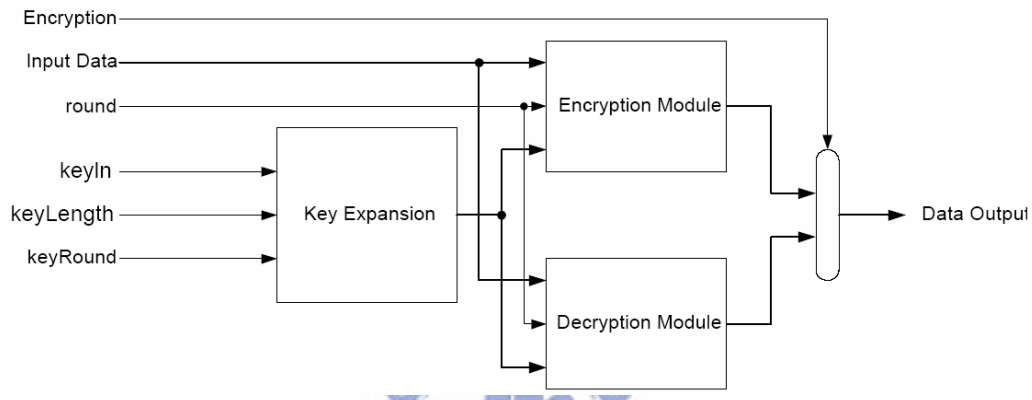

圖 3.5 AES 演算法全部電路

我們將 AES 演算法加密及解密部份分為兩個次模組：U_aes、U_iaes。代表加密電路與解密電路的 U_aes 以及 U_iaes，其 IP 來自 OPENCORES 網站為免費 IP，兩個次模組的輸出與輸入介面圖示如下圖 3.6。

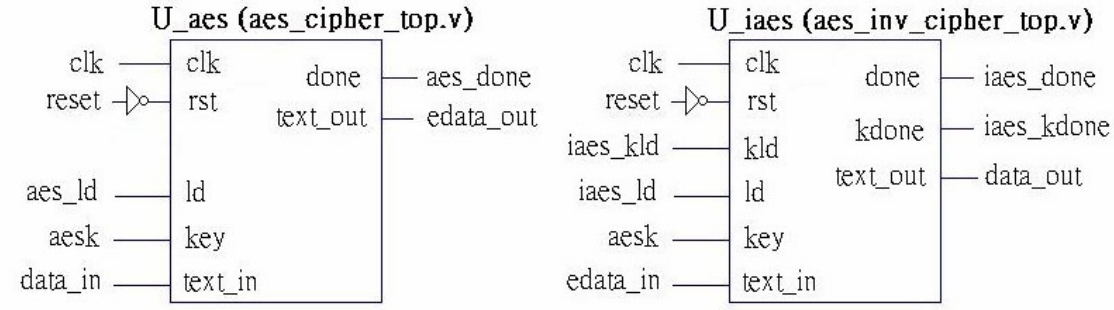

圖 3.6 U_aes、U_iaes 之輸入與輸出訊號

3.2.3 RSA 演算法

RSA 演算法是由 Rivest、Shamir 及 Adleman 三位教授於 1978 年所提出，適用於公開金鑰密碼系統的演算法，同時也是最簡單的演算法之一，因為其加密程序及解密程序為一次模指數運算即可，同時，其安全性係建立在分解已知整數 N 的質因數，當已知整數 N 越大，則分解的困難度越高。

A. 產生金鑰：(Public Encryption Key & Private Decryption Key)

1. 任意選擇兩個不同的大質數(p,q)，其乘積即為模數 $N = p \times q$。
2. $\phi(N)$為 N 的尤拉商數，即小於 N 且與 N 互質的正整數個數。
3. 選擇e使得$\gcd(e,\phi(N)) = 1$，計算 d 使得$e \times d \equiv 1 (\mathrm{mod}\, \phi(N))$。
4. 公開加密金鑰由(e,N)所組成，私密解密金鑰由(d,n)組成。

B. 加密程序與解密程序：(Encryption & Decryption) 如圖 3.7

1. 明文 (Message)：M，密文 (Cipher-text)：C
2. 加密：$C \equiv M^e (\mathrm{mod}\, N)$，解密：$M \equiv C^d (\mathrm{mod}\, N)$。

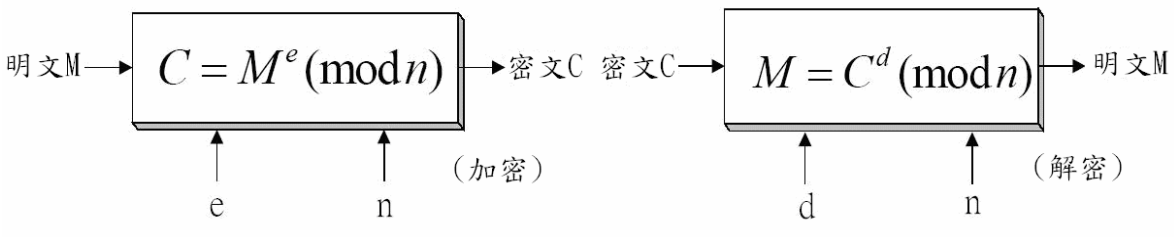

圖 3.7 RSA 加密與解密程序

C. 簽章與認證：(Signature & Verification)

1. 明文 (Message)：M，簽章 (Signature)：S

2. 簽章：$S \equiv \mathrm{M}^{\mathrm{d}} (\mathrm{mod}\, N)$，認證：$M \equiv \mathrm{S}^{\mathrm{e}} (\mathrm{mod}\, N)$。

一般相信 RSA 密碼系統的安全性等若於分解整數問題;若能分解模數 N 得知 p、q，即可得到$\phi(N) = (p-1) \times (q-1)$，進而得到解密金鑰 d，破解 RSA 密碼系統。然而，隨著模數 N 的長度增加，質因數分解是一件非常困難的事。當然每一種密碼都無法達到不管破密者截獲多少密文 C，加以分析後的結果與直接猜明文 M 的方式是一樣的，即所謂的理論安全。不過只需要達到實際安全的層度就可以了。而實際安全的層度即一系統之工作特徵大到使得具備有限計算能力及記憶體的破密者，無法在合理的時間內破解此系統，則此系統即可稱為實際安全或計算上安全的密碼系統。DES 編碼比 RSA 保護更弱，為什麼還是可以使用呢？其實就是因為破密者無法在合理的時間之內、使用合理的資源來破解密碼系統。

根據應用上的不同或加解密的速度考量，改變 RSA 密碼系統的安全等級，只需要改變模數長度即可，這樣的彈性設計也可算是 RSA 密碼系統的一項優點。

表 3.1 顯示各類密碼法之處理效率比較。其中可明顯發現對稱式金鑰的處理效率較非對稱式金鑰密碼法為快，如 DES/AES 的處理效率為 RSA 的一千倍，故我們可以預期本研究之加解密過程中 RSA 演算法將暫用大部分之時間。然非對稱式密碼法則提供不可否認性的能力，且於訊息加密前，其不需如對稱式金鑰密碼法必須秘密協調出通訊雙方的通訊金鑰等優點。因此，在安全與速度的雙重考量下，本研究將此兩密碼進行結合，即訊息傳送方將通訊使用的通訊金鑰以接收方之公開金鑰加密，之後的大量通訊資料則以共同擁有的通訊金鑰進行加解密處理。

表 3.1 各類密碼法及網路連結處理效率比較表

密碼技術	處理效率（處理數量/秒）
非對稱式金鑰密碼法（1024bit RSA）	2
對稱式金鑰密碼法（DES/AES）	2，000
單向雜湊函數（MD5/SHA）	20，000
網路連線（TCP/Internet）	1，000

本研究所使用之 RSA 演算法來自 OPENCORES 網站，並將 VHDL CODE 改寫為 VERILOG CODE。為了硬體實作之方便和速度起見將金鑰的長度暫定為 128 位元。雖然本程式金鑰的長度為可更改之設計，長度可以長達 1024 位元以確保所需高度的安全性但需要修改其他模組加以配合。次模組 U_rsa 之輸入與輸出介面如下圖 3.8 所示。RSA 演算法在本研究中可說佔極為重要的地位，首先 RSA 必須負責先加密 Session Key、之後又有簽章、解密 Session Key 以及解密簽章等動作必須再用到 RSA 演算法。

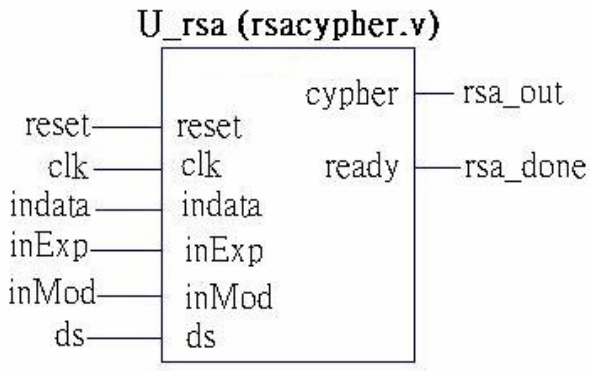

圖 3.8 U_rsa 之輸入與輸出訊號

3.2.4 MD5(Message Digest)演算法

MD5(Message Digest)訊息摘要演算法是由提出 RSA 演算法中的 MIT 教授 Ron Rivest 所發展出來，可用以驗證資料內容的完整性，其輸入任何長度的訊息產生輸出一 128 位元的訊息摘要，輸入訊息會被分成好幾個 512 位元的區塊來處理。雖然較長的訊息摘要長度可以減少雜湊值碰撞的機率，但相對的運算時間會較長。

不同大小或是不同內容的兩個檔案，所產生的 MD5 碼都不會相同。因此，MD5 可以被用在數位簽章上，用以判斷數位簽章的完整性。在 MD5 之前還有 MD2 和 MD4 演算法，MD2 演算法較適於用在 8 位元的系統上，而 MD4 和 MD5 則適於用在 32 位元的系統上。MD5 是由 MD4 延伸出來，不過可以提供比 MD4 更佳的安全性。本研究所使用之次模組 U_md5 取自 OPENCORES 網站，其輸出與出入介面如下圖 3.9。

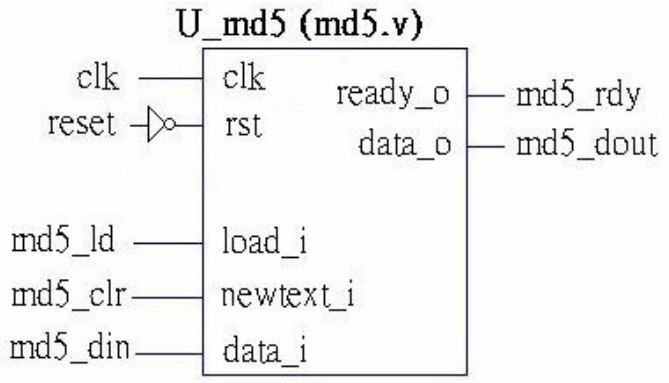

圖 3.9 U_md5 之初入與輸出訊號

3.3 系統運作之流程

本研究欲透過加裝安全模組晶片使行動裝置，甚至 PDA 或行動電話成為可信賴之防竄改裝置，安全模組晶片之中提供資料加解密以及認證、簽章等功能。更能進一步使 PTD 成為 PKI 架構下，進行認證、資料加解密及數位簽章等活動的具體裝置或載具。

1. PTD 與 PC（主機）間之資料傳送

傳送方（如圖 3.10）：

1. 亂數產生加密本文的 Session Key。
2. 利用 Session Key 加密本文，產生加密文件。
3. 用接收方的 Public Key 加密 Session Key。
4. 利用傳送方的 Private Key 對本文訊息摘要做簽章的動作。
5. 將加密過的 Session Key、簽章、加密文件，傳給接收方。

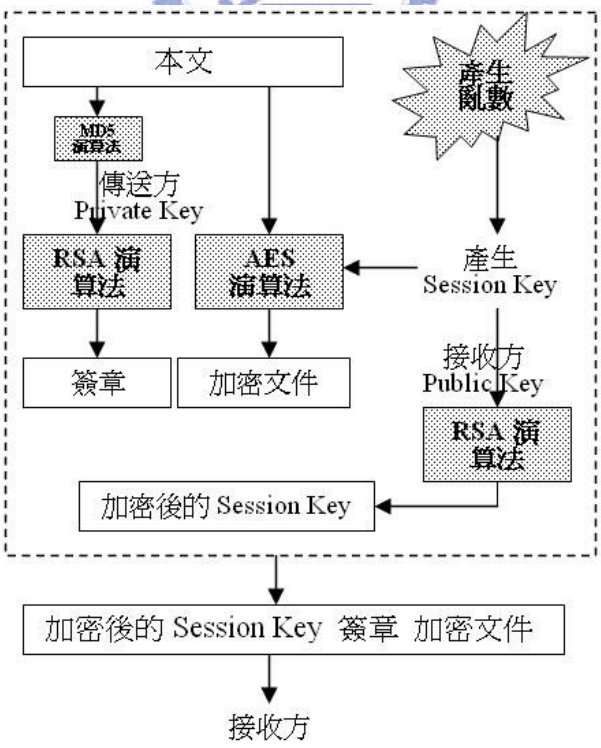

圖 3.10 傳送方之動作

接收方（如圖 3.11）：

1. 接收方的動作恰與傳送方相反
2. 利用接收方的 Private Key，將 Session Key 解密。
3. 利用 Session Key，將加密文件解密。
4. 利用傳送方的 Public Key 驗證簽章，如果驗證通過則完成本文的交換，反之，則退回給傳送方。

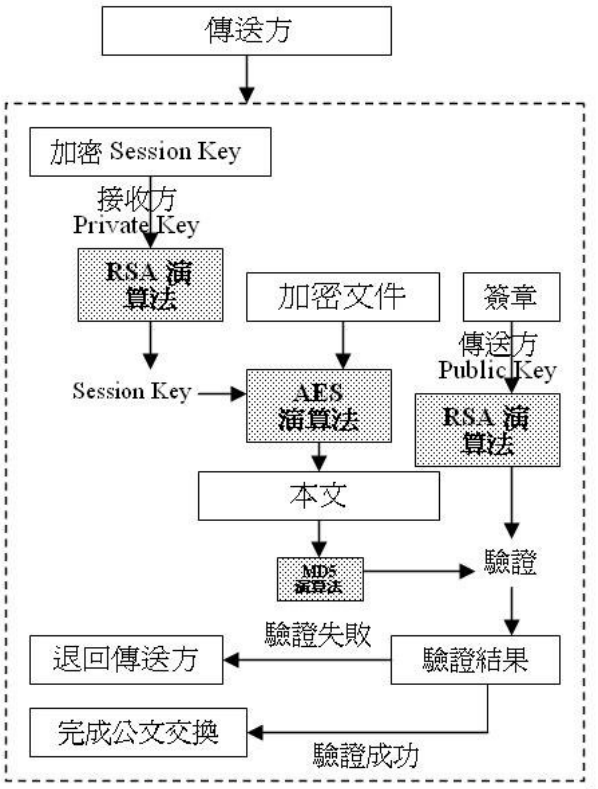

圖 3.11 接收方之動作

2. PTD 自身之資料存檔與讀取（如圖 3.12）

存檔－

1. 亂數產生加密本文的 Session Key。
2. 利用 Session Key 加密本文，產生加密文件。

讀檔－

1. 載入 Session Key
2. 利用 Session Key，將加密文件解密。

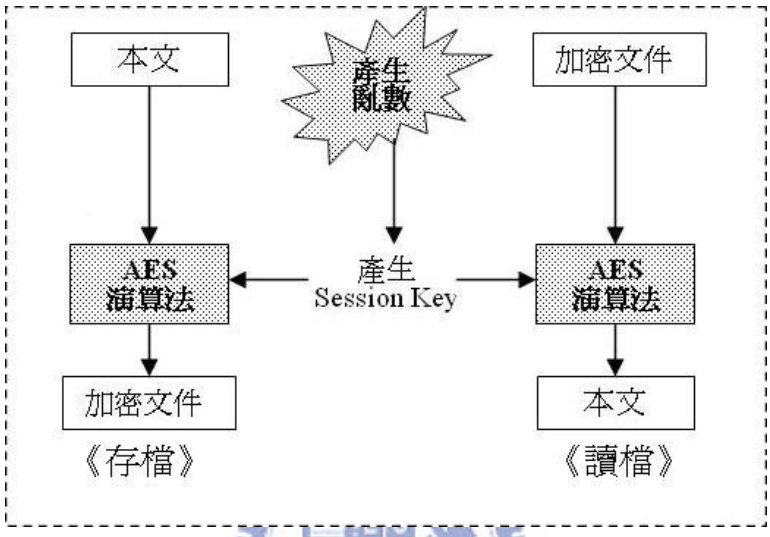

圖 3.12 個人信賴裝置自身存取

此處之 Session Key 可設置為自己方便記憶之密碼再亂數產生器產生剩餘位數，如此一來可以方便記憶。或是不另外設定全由亂數產生 Session Key，存檔或讀檔時可另外使用 PIN 碼進行身份認證。

第四章 實作與評估

本研究提出 PTD 的概念，其安全性主要仰賴於內部的安全元件，此安全元件即我們實作的安全模組晶片。採用 AES 對稱式加解密演算法，RSA 非對稱式加解密，亂數產生（Pseudo Random Number Generator）以 LFSR 來完成，另有 MD5 來做訊息摘要，RSA 做數位簽章，唯一功能完整之整合晶片如下圖 4.1 所示。

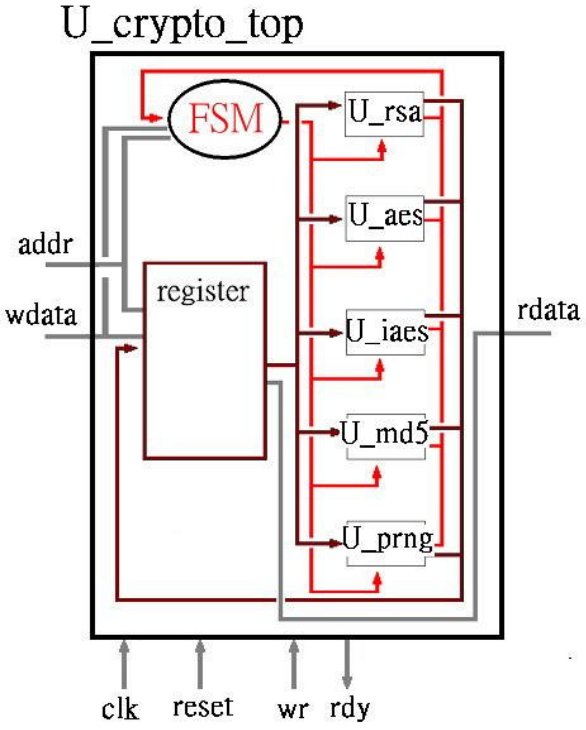

圖 4.1 主模組之輸入與輸出訊號

我們將以現場可程式化閘陣列(Field Programmable Gate Array，FPGA)依晶片設計之流程來實作安全模組晶片。並在本章的最後評估加裝安全模組晶片後可確保哪些安全。

4.1 安全模組晶片實作

A. 功能描述

此模組設計成 controller 的形式，預期將接在匯流排（bus）上由 CPU 來填資料。本模組可經由填寫參數與利用不同的命令，加速一同時使用對稱式與非對稱式加解密模組的公文系統。此模組可隨機產生 session key，並且在使用與傳輸過程中 session key 不會有任何外洩的危險。

B. I/O 介面

本模組的 I/O 介面為 32 bits，支援的 key length 為 128 bits。圖 4.2 為本模組之 I/O 介面圖示，可看見最上層模組所有輸入與輸出信號。其中 Address bus 之長度為 8 bits，Write data bus 及 Read data bus 長度皆為 32 bits。

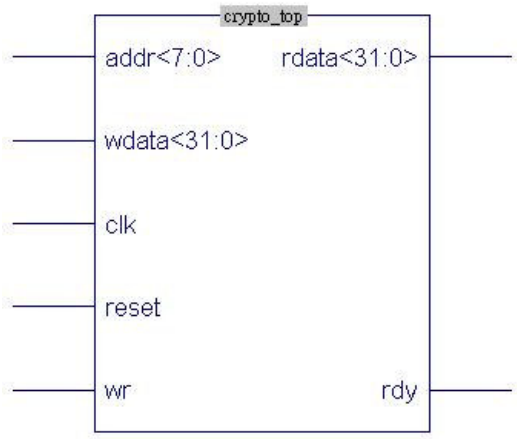

圖 4.2 I/O 介面

表 4.1 輸入/輸出介面

Name	Direction	Active	Description
Reset	Input	High	重置訊號，pull high 時資料歸零
Clk	Input		時脈訊號
Addr[7:0]	Input		Address bus
Wdata[31:0]	Input		Write data bus
Rdata[31:0]	Output		Read data bus
Rdy	Output	High	Ready 信號：當模組處於可以接受 command 的狀態時為 high。執行命令時 rdy 為 low。

C. 架構:

表 4.2 次模組功能描述

次模組	功能描述
U_rsa	負責 RSA 加解密運算。
U_aes	負責 AES 加密運算，key 可直接 apply。
U_iaes	負責 AES 解密，須先進行 load key 動作才能進行資料的解密。
U_prng	以 LFSR 完成亂數產生，藉此產生 session key。
U_md5	負責 MD5 hash function。

本模組共包含五個次模組: U_rsa，U_aes，U_iaes，U_prng，U_md5，已於前一章簡介其功能。而位在五個次模組下之其它功能模組如圖 4.3 所

示。除五個次模組之外另有儲存 public key，private key，data 以及 signature 的 register 若干個，以及用來接受 command 以控制流程的有限狀態機（Finite State Machine，FSM）如圖 4.4 所示。

圖 4.3 所有模組及更下層模組

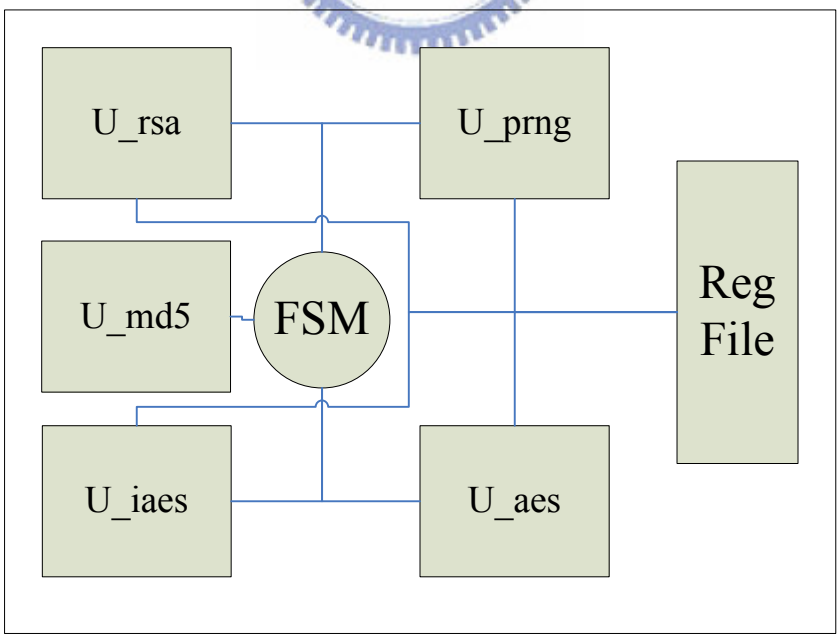

圖 4.4 有限狀態機

此有限狀態機將根據某些情況或指令轉換其狀態如圖 4.5 所示，其共有八種狀態，所有狀態簡介如表 4.3。

表 4.3 FSM 之狀態

0	IDLE	Idle
1	GKEY	Get AES key
2	ECDA	Encrypt data
3	SIGN	Signature
4	DLKY	Deliver Load key
5	DCKY	Decrypt AES key
6	DCDA	Decrypt data
7	VRFY	Verify

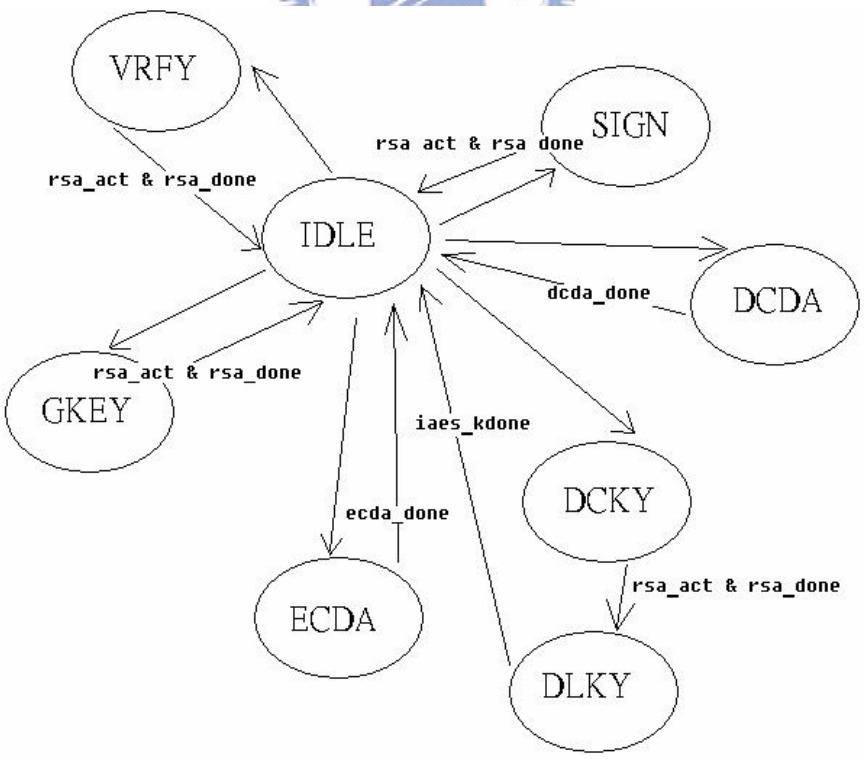

圖 4.5 狀態轉換圖

D. 位址映射（Memory Map）

Memory Map 因為每個 field 都是 32 bit，因此位址以 0，4，8，c 為單位作存取。

例如當要填入 Plain text 做加密時，address 為 0x00，0x04，0x08，0x0c，共 32x4 = 128 bit 的 data，其中 0x00 為 data[31:0]，0x04 為 data[63:32]，0x08 為 data[95:64]，0x0c 為 data[127:96]。系統為 little endian。

表 4.4 位址映射

0x00 - 0x0c	RW	Plain text for encryption， cypher for decryption
0x10 - 0x1c	WO	Private key d
0x20 - 0x2c	WO	Private key n
0x30 - 0x3c	WO	Public key e
0x40 - 0x4c	WO	Public key n
0x50 - 0x5c	RW	Cipher for encryption Plain text for decryption
0x60 - 0x6c	RW	Encrypted key
0x70 - 0x7c	RW	Digest of signature
0x80	WO	Command register
0x80	RO	Status register

E. 使用方式

圖 4.5、圖 4.6 顯示加解密所需流程及需要填寫的參數。其中藍色方格為下達的 command，橢圓區塊為完成命令之後產生的結果值，灰色方塊則是下達命令前需要填寫的參數。下達 command 後，rdy 訊號會 pull low，必須要等待 rdy 訊號再度 pull high 才表示命令完成。

加密流程（如圖 4.6）：

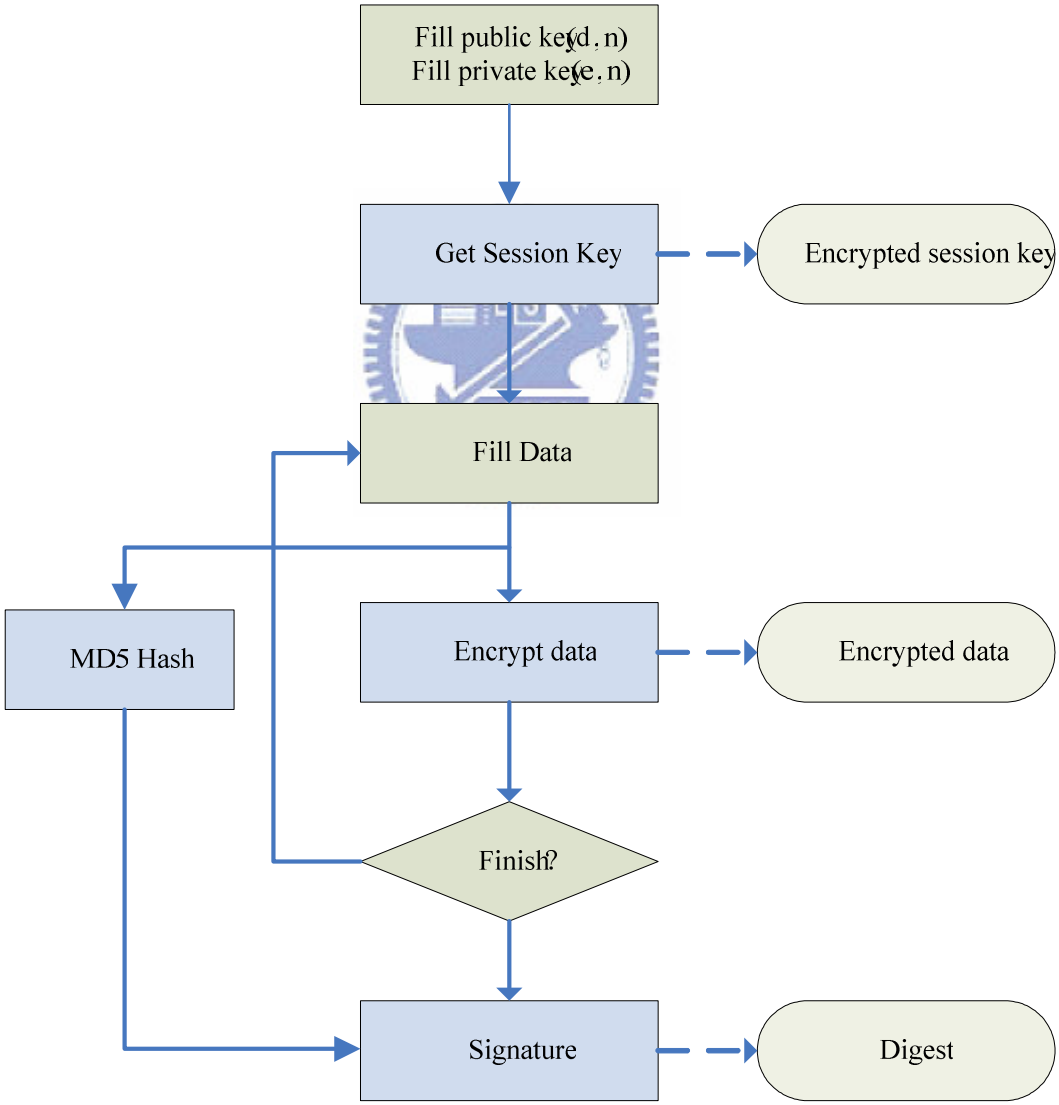

圖 4.6 加密流程

解密流程（如圖 4.7）：

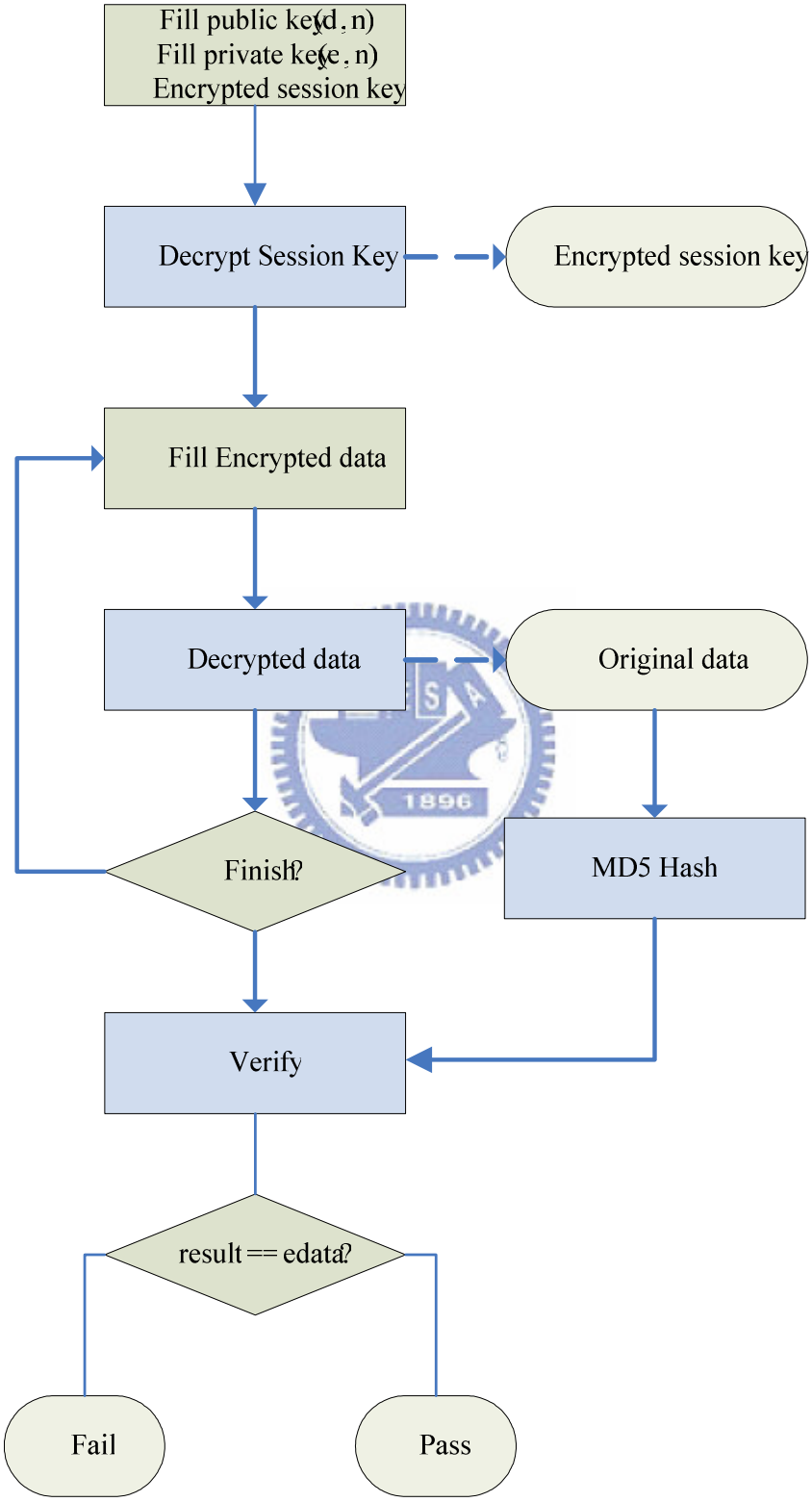

圖 4.7 解密流程

F. 命令描述:

此處描述 command register 對應的命令值如表 4.5 所示。

表 4.5 命令描述

Command	[2:0]	運算命令	0x0 Get AES key 0x1 Encrypt data 0x2 Decrypt AES key 0x3 Decrypt data 0x4 Generate signature 0x5 Verify
	[3]	停止	對此 bit 寫一時，可馬上停止進行的運算，讓模組回到等待接收命令的狀態，但產出資料無法控制。
Status	[0]	驗證結果	0: 表示驗證通過 1: 表示驗證失敗

G. 時序圖

寫入 command（如圖 4.8）:

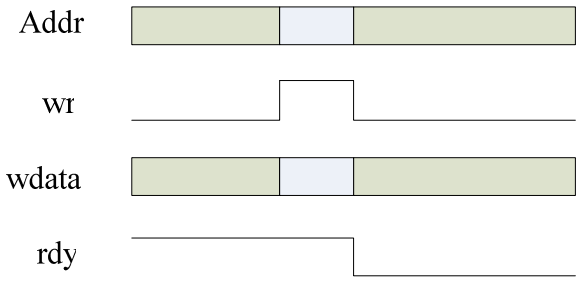

圖 4.8 寫入 Command

寫入 pattern（如圖 4.9）：

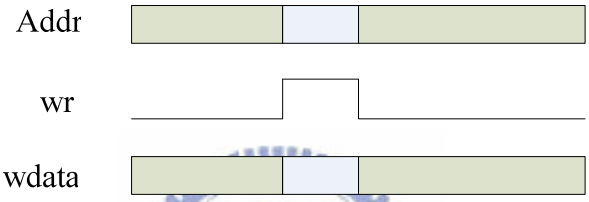

圖 4.9 寫入 pattern

讀取（如圖 4.10）：

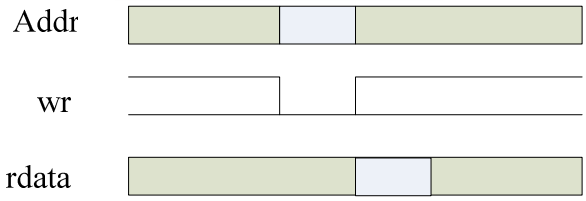

圖 4.10 讀取

H. 設計環境

1. 軟體環境：

- Simulation：Mentor Graphic Modelsim
- Synthesize：Xilinx cor. XST

◆ Implementation：Xilinx cor. ISE

2. 硬體環境：Xilinx ML310 實驗板（如圖 4.11 所示）

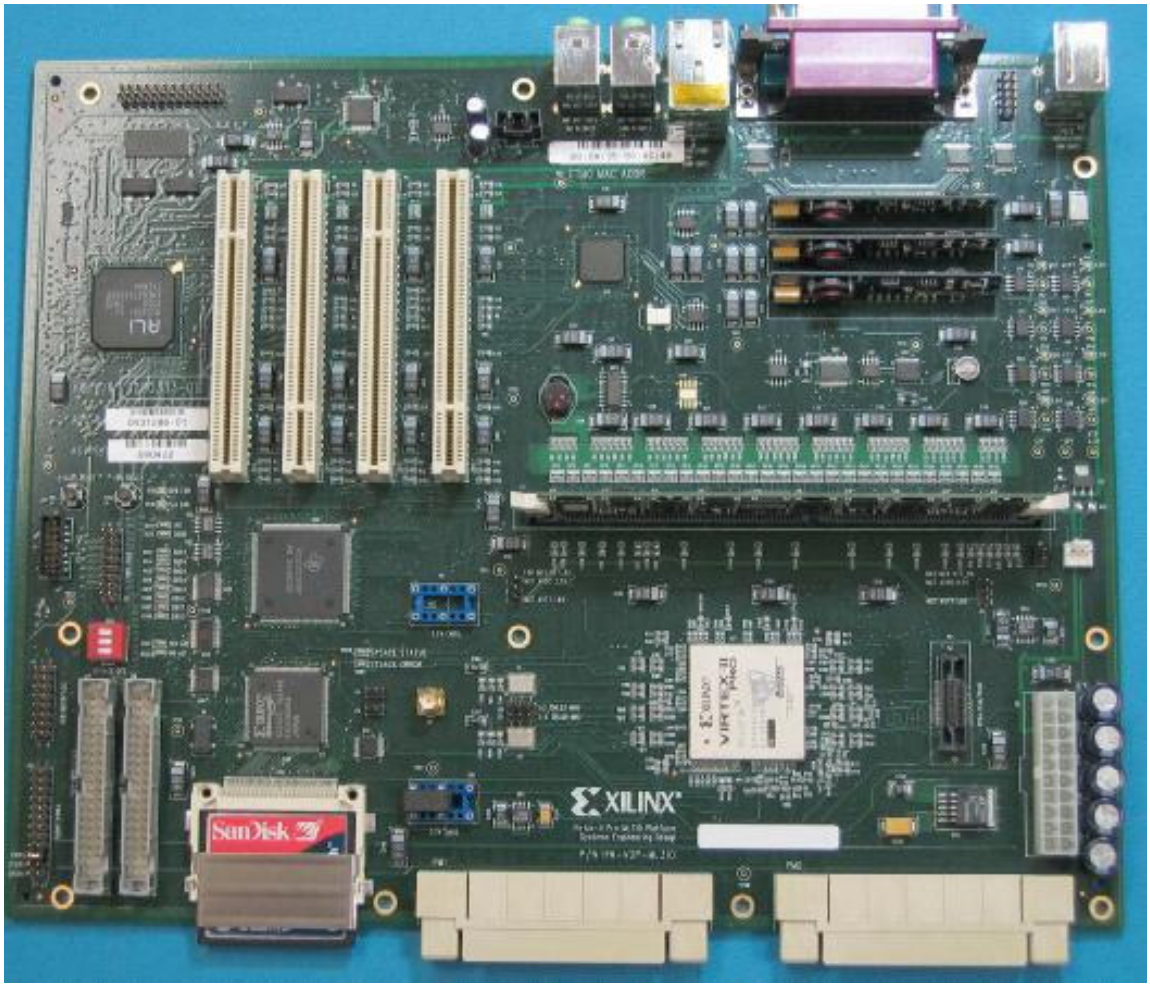

圖 4.11 Xilinx ML310

4.2 系統驗證與測試

本模組有兩種方式驗證:其一使用 functional test bench 進行 behavior simulation。其二為撰寫一個 synthesizable test pattern

generator，reset 完就會自動把 pattern 跑到完，到時候一起合成弄到系統裡去，download 至 FPGA 執行。

Functional test bench 與 pattern generator 的時序是相同的皆為 300ns。運作順序為寫入 pattern，取得 session key，加密資料，簽章。完成加密動作之後，改做解密程序：解密 session key，解密資料，驗證。Pattern generator 並有一 test_end 訊號表示測試程序結束，以及一 verify_ok 訊號表示驗證簽章結果無誤如圖 4.12。

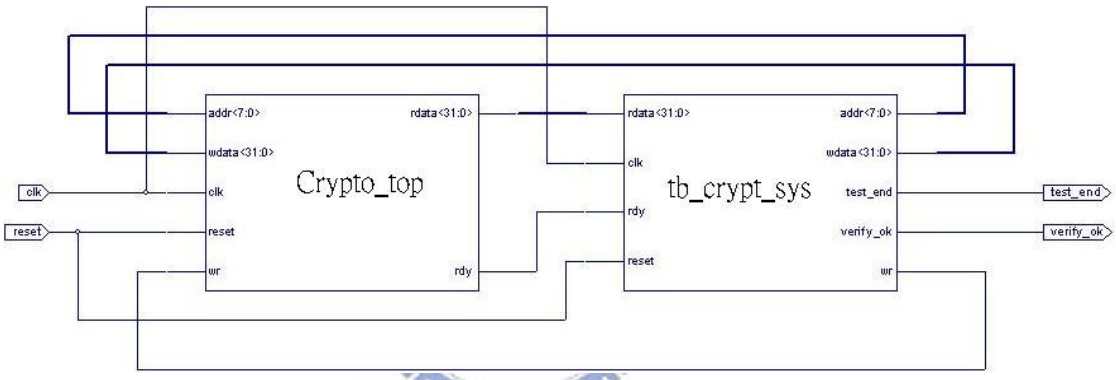

圖 4.12 Testbench

本研究所使用之實驗板為 Xilinx ML310 開發板（2VP30 FF896-6）。唯本實驗板雖功能強大足以實作此安全晶片如下圖 4.13 所示。但 Xilinx ML310 沒有顯示燈號，沒有按鈕、開關造成驗證上的不便。因此在確認程式沒有錯誤且可合成在板子上後，改以前文中第一種方法利用 ModelSim SE 5.8 模擬工具來觀看所設計的 functional test bench 執行後的結果。

```
Device utilization summary:
---------------------------

Selected Device : 2vp30ff896-6

 Number of Slices:                     8263  out of  13696    60%
 Number of Slice Flip Flops:           5302  out of  27392    19%
 Number of 4 input LUTs:              15057  out of  27392    54%
 Number of bonded IOBs:                  75  out of    556    13%
 Number of GCLKs:                         1  out of     16     6%
```

圖 4.13 Synthesize 後之 Final Report

我們將所有需要的程式碼和測試碼放在同一目錄下，以 ModelSim SE 5.8 的模擬工具，去模擬整個流程並觀察簽章及資料加密、解密的正確與否以及相關波形。

Modelsim 使用方法：

1. 開啟 Modelsim 後，在 command shell 底下切換到解開目錄的所在位置

   >> cd e:\RSA

2. 確認所在位置正確：用 dir 應該會出現目錄下檔案（如圖 4.14）

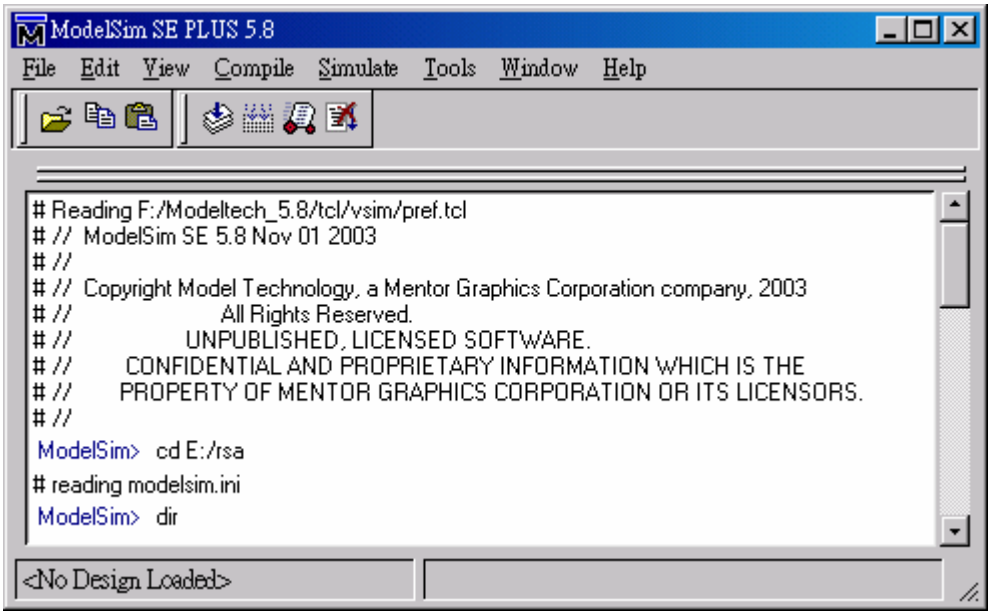

圖 4.14 模擬流程 a

3. 執行 script（如圖 4.15）：

   >> do run.do

4. waveform 出現（如圖 4.16）

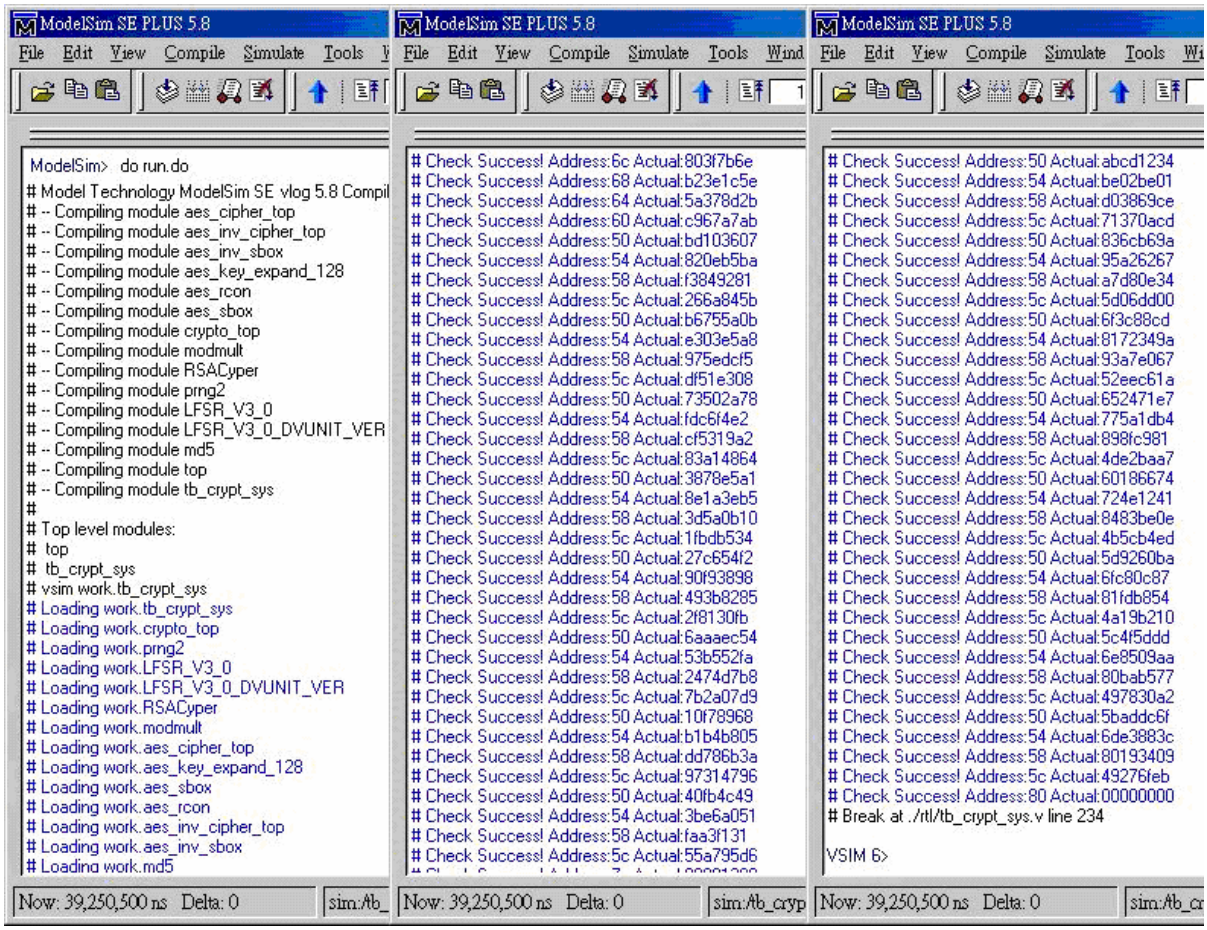

圖 4.15 模擬流程 b

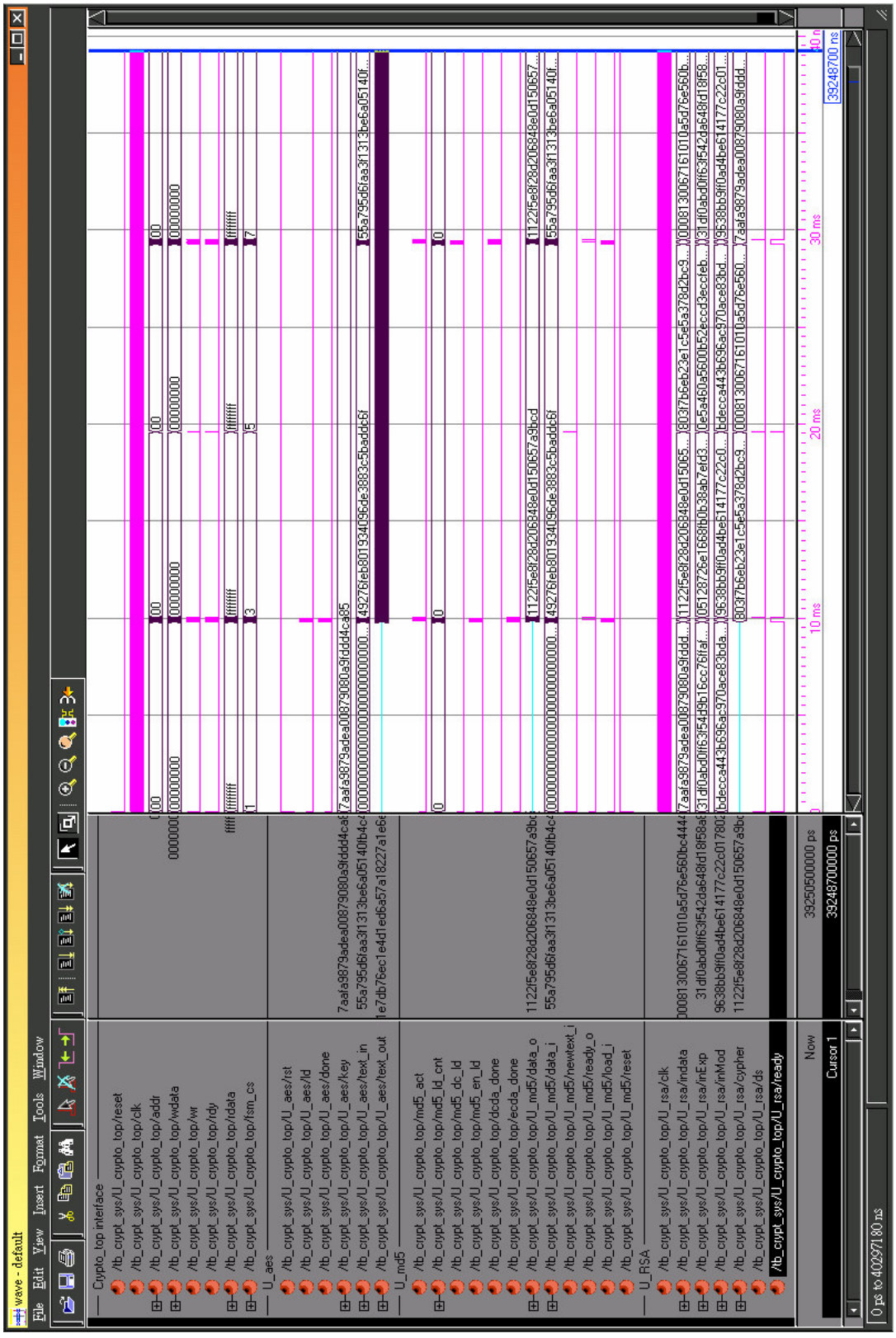

圖 4.16 波形圖（反白效果）

由圖 4.16 之波型圖可知本程式如預期般大部份之時間皆花費在 RSA 演算法之運算，所花費時間不到 40 毫秒（ms）其中有大多時間處在 FSM 的第 1、3、5、7 的狀態。而 FSM 的第 1、3、5、7 的狀態正好是 Get Session Key、Signature、Decrypt Session Key、Verify，皆為 RSA 演算法的運算。

我們檢視 test bench 來判斷程式是否正確。在 functional test bench 中將資料做初始化產生一筆雜亂無意義之資料如圖 4.17 所示，並設計在做 R_single 這個動作時即對預期讀出資料與實際讀出資料做比對，且若比對不同將顯示 ERROR 訊息，比對相等則將顯示出 SUCCESS 訊息如圖 4.18 所示。

```
//data initialization
data[0] = 32'habcd1234;

for(i=0;i<31;i=i+1) begin
    data[i+1] = data[i] + 32'h1235abcd ;
  if(i%4 == 2)
    data[i+1] = data[i+1]>>1;
end
```

圖 4.17 資料初始

```
task R_single;
        input [7:0] addr_v;
        input [31:0] rdata_v;
        output [31:0] actual_v;
        begin
        ->clear_reg;
        @(posedge clk)
        addr = addr_v;
        wr = 1'b0;
        @(posedge clk)
        if(rdata != rdata_v)
$display("Error! Value of Address:%x isn't as expected! Expect:%x Actual:%x", addr, rdata_v, rdata);
        else
$display("Check Success! Address:%x Actual:%x", addr, rdata);
       actual_v = rdata;
end
endtask
```

圖 4.18 R_single

我們可以在 functional test bench 發現做了 R_single 共 73 次（如圖 4.19 所示）。而做第 41 次 R_single 時正好為初始之 data[0]經過加密後再解密所得之結果，所以我們可以預期若正確無誤資料與實際資料比對 73 次，而第 41 次將是 data[0]的初始值：abcd1234。結果顯示確實顯示了 73 次的 SUCCESS 訊息，並且第 41 次顯示的也的確是 data[0]的初始值：abcd1234，程式也在驗證完後停止動作如圖 4.20 所示。

```
    // Write command: Getkey
    W_single(8'h80, 32'h0000_0000);
    ->clear_reg;
    wait(rdy);
    R_single(8'h6c, 32'hx, ekey3);
4次 R_single(8'h68, 32'hx, ekey2);
    R_single(8'h64, 32'hx, ekey1);
    R_single(8'h60, 32'hx, ekey0);

    // Write command: Encrypt
    for(i=0;i<32;i=i+4) begin
      W_single(8'h00, data[i]);
      W_single(8'h04, data[i+1]);
      W_single(8'h08, data[i+2]);
      W_single(8'h0c, data[i+3]);

      W_single(8'h80, 32'h0000_0001);
      ->clear_reg;
      wait(rdy);
      R_single(8'h50, 32'hx, edata[i]);
32次  R_single(8'h54, 32'hx, edata[i+1]);
      R_single(8'h58, 32'hx, edata[i+2]);
      R_single(8'h5c, 32'hx, edata[i+3]);
    end

    // Write command: Sign
    W_single(8'h80, 32'h0000_0004);
    ->clear_reg;
    wait(rdy);
    R_single(8'h7c, 32'hx, sig3);
4次  R_single(8'h78, 32'hx, sig2);
    R_single(8'h74, 32'hx, sig1);
    R_single(8'h70, 32'hx, sig0);
    ->clear_reg;
```

```
    // Write Signature
    W_single(8'h70, sig0);
    W_single(8'h74, sig1);
    W_single(8'h78, sig2);
    W_single(8'h7c, sig3);

    // Write command: Decrypt key
    W_single(8'h80, 32'h0000_0002);
    ->clear_reg;
    wait(rdy);

    // Write command: decrypt data
    for(i=0;i<32;i=i+4) begin
      W_single(8'h00, edata[i]);
      W_single(8'h04, edata[i+1]);
      W_single(8'h08, edata[i+2]);
      W_single(8'h0c, edata[i+3]);

      W_single(8'h80, 32'h0000_0003);
      ->clear_reg;
      wait(rdy);
      R_single(8'h50, data[i], edata[i]);
32次  R_single(8'h54, data[i+1], edata[i+1]);
      R_single(8'h58, data[i+2], edata[i+2]);
      R_single(8'h5c, data[i+3], edata[i]);
    end

    // Write command: Verify
    W_single(8'h80, 32'h0000_0005);
1次  ->clear_reg;
    wait(rdy);
    R_single(8'h80, 32'h0000_0000, status);

    $stop;
end
```

圖 4.19 R_single 之次數

ModelSim SE PLUS 5.8
File Edit View Compile Simulate Tools Window Help
ModelSim> do run.do
Model Technology ModelSim SE vlog 5.8 Compil
-- Compiling module aes_cipher_top
-- Compiling module aes_inv_cipher_top
-- Compiling module aes_inv_sbox
-- Compiling module aes_key_expand_128
-- Compiling module aes_rcon
-- Compiling module aes_sbox
-- Compiling module crypto_top
-- Compiling module modmult
-- Compiling module RSACyper
-- Compiling module ping2
-- Compiling module LFSR_V3_0
-- Compiling module LFSR_V3_0_DVUNIT_VER
-- Compiling module md5
-- Compiling module top
-- Compiling module tb_crypt_sys
Top level modules:
top
tb_crypt_sys
vsim work.tb_crypt_sys
Loading work.tb_crypt_sys
Loading work.crypto_top
Loading work.ping2
Loading work.LFSR_V3_0
Loading work.LFSR_V3_0_DVUNIT_VER
Loading work.RSACyper
Loading work.modmult
Loading work.aes_cipher_top
Loading work.aes_key_expand_128
Loading work.aes_sbox
Loading work.aes_rcon
Loading work.aes_inv_cipher_top
Loading work.aes_inv_sbox
Loading work.md5
Now: 39,250,500 ns Delta: 0
sim:/tb_
Check Success! Address:6c Actual:803f7b6e
Check Success! Address:68 Actual:b23e1c5e
Check Success! Address:64 Actual:5a378d2b
Check Success! Address:60 Actual:c967a7ab
Check Success! Address:50 Actual:bd103607
Check Success! Address:54 Actual:820eb5ba
Check Success! Address:58 Actual:f3849281
Check Success! Address:5c Actual:266a845b
Check Success! Address:50 Actual:b6755a0b
Check Success! Address:54 Actual:e303e5a8
Check Success! Address:58 Actual:975edcf5
Check Success! Address:5c Actual:df51e308
Check Success! Address:50 Actual:73502a78
Check Success! Address:54 Actual:fdc6f4e2
Check Success! Address:58 Actual:cf5319a2
Check Success! Address:5c Actual:83a14864
Check Success! Address:50 Actual:3878e5a1
Check Success! Address:54 Actual:8e1a3eb5
Check Success! Address:58 Actual:3d5a0b10
Check Success! Address:5c Actual:1fbdb534
Check Success! Address:50 Actual:27c654f2
Check Success! Address:54 Actual:90f93898
Check Success! Address:58 Actual:493b8285
Check Success! Address:5c Actual:2f8130fb
Check Success! Address:50 Actual:6aaaec54
Check Success! Address:54 Actual:53b552fa
Check Success! Address:58 Actual:2474d7b8
Check Success! Address:5c Actual:7b2a07d9
Check Success! Address:50 Actual:10f78968
Check Success! Address:54 Actual:b1b4b805
Check Success! Address:58 Actual:dd786b3a
Check Success! Address:5c Actual:97314796
Check Success! Address:50 Actual:40fb4c49
Check Success! Address:54 Actual:3be6a051
Check Success! Address:58 Actual:faa3f131
Check Success! Address:5c Actual:55a795d6
Now: 39,250,500 ns Delta: 0
sim:/tb_cryp
ModelSim SE PLUS 5.8
File Edit View Compile Simulate Tools Window Help
Check Success! Address:50 Actual:abcd1234
Check Success! Address:54 Actual:be02be01
Check Success! Address:58 Actual:d03869ce
Check Success! Address:5c Actual:71370acd
Check Success! Address:50 Actual:836cb69a
Check Success! Address:54 Actual:95a26267
Check Success! Address:58 Actual:a7d80e34
Check Success! Address:5c Actual:5d06dd00
Check Success! Address:50 Actual:6f3c88cd
Check Success! Address:54 Actual:8172349a
Check Success! Address:58 Actual:93a7e067
Check Success! Address:5c Actual:52eec61a
Check Success! Address:50 Actual:652471e7
Check Success! Address:54 Actual:775a1db4
Check Success! Address:58 Actual:898fc981
Check Success! Address:5c Actual:4de2baa7
Check Success! Address:50 Actual:60186674
Check Success! Address:54 Actual:724e1241
Check Success! Address:58 Actual:8483be0e
Check Success! Address:5c Actual:4b5cb4ed
Check Success! Address:50 Actual:5d92260ba
Check Success! Address:54 Actual:6fc80c87
Check Success! Address:58 Actual:81fdb854
Check Success! Address:5c Actual:4a19b210
Check Success! Address:50 Actual:5c4f5ddd
Check Success! Address:54 Actual:6e8509aa
Check Success! Address:58 Actual:80bab577
Check Success! Address:5c Actual:497830a2
Check Success! Address:50 Actual:5baddc6f
Check Success! Address:54 Actual:6de3883c
Check Success! Address:58 Actual:80193409
Check Success! Address:5c Actual:49276feb
Check Success! Address:80 Actual:00000000
Break at ./itl/tb_crypt_sys.v line 234
VSIM 6>
Now: 39,250,500 ns Delta: 0
sim:/tb_crypt_sys

圖 4.20 程式正確無誤

4.3 系統安全性分析

本研究所提出之 PTD 概念，其安全是由內部之安全模組晶片所保護。在這裡討論之安全性主要是基於國內政府機關資訊處理共通規範的資訊安全標準，而上述標準所定義的資訊安全服務應包含下列四項，包含隱私性、認證性、完整性與不可否認性，在此將針對這四項進行個人信賴裝置安全性之分析，並說明為何 PTD 將具備此四項安全性：

A. 隱私性（Privacy）

為防止非法者從網路上得知通信內容，本研究使用對稱式的 AES 加密演算法對資料進行加密以確保資料的隱私性。假若不幸送修或被竊時：因為產生 AES key 之後，PTD 內儲存資料用 AES 演算法加密，所以即使同樣用傳輸線，敏感的資料也不會外流。

B. 認證性（Authentication）

認證性定義為甲能向乙証明自己，但是別人不能向乙証明他是甲。本研究採用個人信賴裝置（PTD）作為認證依據，已驗證 PTD 所屬的使用者。在通過使用者驗證之後，PTD 才可以被用來作通訊或交易。例如，需先輸入兩組 PIN 以驗證身分，若非法持有 PTD 人連續輸入三次錯誤的 PIN，機器將會被鎖住，而該識別碼也會變為無效，必須要由原認證中心才能解除等方法。可避免身分遭人偽造或遭暴力攻擊法的入侵，以達成認證上的安全。而 PTD 在安全的傳出用接受方 Public Key 加密過的 AES key 後，在企業或公司內部 PC 端就可以用其 Private Key 順利解出 AES key 而後再解出資料。

C. 完整性（Integrity）

資訊於傳輸的過程中，本身必須免於任何形式之竄改或損害。此項評估的目的在於檢查資料是否有被更改過，亦即在加密前根據原資料產生出該資料的檢查碼，在解密還原後再產生一次檢查碼，並核對前後二次的檢查碼是否一致。為確保接收方可確定接收到的訊息沒有被有意或無意的更改，及被部份取代、加入或刪除等，本研究中，傳輸的資料使用 MD5 演算法取得資料內容的訊息摘要，即雜湊值（hash value）。利用 MD5，可將原本的明文轉換為 128 位元的雜湊函數值，若訊息資料有誤，則檢查碼將會不正確。

D. 不可否認性（Non-repudiation）

其功能是在交易的過程中保留證據，以解決可能發生的爭議。由於本系統資料的交換，是利用雜湊函數（Hash Function）與數位簽章（Digital Signature）等方法完成，利用這些演算法對交易所產生的電子簽章，可用來作為交易雙方日後的授權依據。如此，除了擁有密鑰的使用者外，其餘使用者是無法進行簽章的更改的，這樣可以利用此種方法來實現資料的不可否認性。例如在企業內部 PTD 傳送資料到公司主機，PTD 可利用傳送方的 Private Key 達到簽章確認的手續。而 PC 端有 PTD 的 Pubic Key 可以用來驗證簽章相不相等，以此達到不讓非公司的 PTD 連上電腦，即使他可以取得公司的 Public Key。

第五章 結論與建議

隨著通訊技術的進步，生活中處處需要利用網際網路或其他通訊設備來交換資料。網路資料處理成為電腦系統所需解決的問題，這方面的問題在過去是由一般 CPU 來處理，而現在已發展成由專門的網路處理器(NPU)來提供高速的封包處理，以解決日亦嚴重的網路流量問題。而資訊加密的處理也跟著網路資料傳輸一樣在未來將會有越來越多的需求。因此在此篇論文中，設計出具前瞻性的密碼處理晶片來加速一般 CPU 或網路處理器對加密資料的處理並提出個人信賴裝置的概念。

5.1 結論

隨著 Internet 時代的來臨，網際網路的用途甚多，諸如全球資訊網（World Wide Web）、視訊會議（Video Conference）、即時播放（Video On Demand）、遠距教學（Distance Learning）、電子商務（E-Commerce）等，網路已經與我們的生活緊緊聯繫，無法分離。因此，網路安全的問題也越來越受到人們的重視。由於未加密過的資料在網路上是以明文的方式傳送，任何人只要能擷取到封包，即能知道傳送內容。為了能確保旁人無法得知我們傳送的資料，加解密的機制實有其必要。

現今一般的作法，是以傳統的一般用途處理器（general-purpose processor）來處理加解密的過程；這種方式當然可行，但其最常為人所垢病，即其處理效率低落。軟體方式來實現加解密過程在現今網路架構下是很容易做到的。但是，加解密的機制由於運算過程繁雜，耗費處理器的時間甚鉅。因此硬體方式仍應列入可行方案之一。

本論文實作之功能完整的加解密整合晶片，以硬體方式實現了 AES 演算法、RSA 演算法及 MD5 演算法。如此一來將會大大地減低處理器的負擔，並能有效提昇處理加解密的效率。光以 AES 演算法為例，以硬體方式實現其效能也是加解密及密鑰長度而比用 C language 撰寫相對應的程式提升了四千至六千倍之多。

此外，本論文提出之個人信賴裝置的概念可應用在行動通訊上，藉由個人數位助理器（PDA）或行動電話（Cell Phone），作為端末設備及無線通訊的工具，甚至因內建安全加解密晶片更能以加強使用電子現金的實用性。

5.2 後續研究建議

本研究實作之系統，礙於時間與複雜度上的考量，僅使用 modelsim 模擬器進行模擬與驗證作業，未來可進一步實作於行動電話上。

本研究之安全晶片可提供各種行動裝置不論是 PDA 或是行動電話較高的安全性。使得本研究在保密性、認證性、完整性、不可否認性上，都可提供安全的保障。若利用本研究之安全晶片並結合 SIM 卡，如此一來行動電話瀏覽電子公文，並進行數位簽章的運算，使得公文運作更有效率並獲得更高的機動性。

未來，隨著智慧型手機建置的成熟、價格趨於平穩，可提供更方便的瀏覽、輸入裝置，相信此行動裝置的應用將會更加普遍。在高速無線網路

基礎建設成熟、3G 無線寬頻網路標準成型之後，可透過本研究所提供之功能，進行影像、聲音等多媒體訊息內容的簽署，而持卡人身分驗證的部分，將來也可採用指紋辨識取代 PIN 碼的輸入，除了可防止使用者忘記 PIN 碼也能避免 PIN 碼操遭到洩漏。

此外，在本論文進行之中傳出山東大學王小雲教授破解了 MD5 及 SHA-1 兩個雜湊函數的消息。其實王小雲教授的方法縮短了找到碰撞的時間，是一項重要的成果。但她找到的是強無碰撞，要能找到弱無碰撞，才算真正破解，才有實際意義。根據密碼學的定義，如果內容不同的明文，通過雜湊演算法得出的結果（密碼學稱為資訊摘要）相同，就稱為發生了“碰撞”。雜湊演算法的用途不是對明文加密，讓別人看不懂，而是通過對資訊摘要的比對，防止對原文的篡改。強無碰撞是無法產生有實際意義的原文的，也就無法篡改和偽造出有意義的明文。通過強無碰撞偽造一個誰也看不懂的東西，沒有實際意義。所以王小雲教授找到強無碰撞這是個重要的事情，但不意味著密碼被破解。找到一對強無碰撞和找到有實際意義的碰撞，還是有本質區別的。由於時間上的考量就沒有改用其他雜湊函數代替本文中所使用的 MD5 雜湊演算法。如果能用更安全讓大眾沒有疑慮的雜湊演算法代替 MD5，也是未來可以後續研究的一個方向。

參考文獻

[1] National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES). Springfield, VA 22161: National Technical Information Service, Nov. 2001.

[2] Ocean LogicTM, OL_AES AES Core family Rev 1.4 http://www.ocean-logic.com/pub/OL_AES.pdf

[3] C. K. Koc, "RSA hardware implementation," RSA Laboratories, RSA Inc., vision.1, Aug. 1995.

[4] R.L.Rivest, "The MD5 Message Digest Algorithm," RFC 1321,April 1992.

[5] C. C. Yang, T. S. Chang, and C. W. Jen, "A new RSA cryptosystem hardware design based on Montgomery's algorithm," IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing, vol. 45, pp. 908–913, July 1998.

[6] V. R. J. Daemen, "The Block Cipher Rijndael," in Smart Card Research and Application, vol. 1820 of LNCS, pp. 288–296, Springer-Verlag, 2000.

[7] A. Dandalis, V. K. Prasanna, and J. D. P. Rolim, "A comparative study of performance of AES final candidates using FPGAs," in Cryptographic Hardware and Embedded Systems (CHES) 2000, vol. 1965 of LNCS, pp. 125–140, Springer-Verlag, Aug. 2000.

[8] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs," Proc. European Test Conference, pp.237-242, 1991.

[9] C.V. Krishna and N.A. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in Proc. Int. Test Conf., pp. 321 -330, Oct 2002.

[10] A. Nash, W. Duane, C. Joseph, D. Brink, 2001, PKI Implementing and Managing E-Security, McGraw-Hill, Berkeley.

[11] E. Kalligeros, X. Kavousianos, D. Bakalis, and D. Nikolos, "A new reseeding technique for LFSR-based test pattern generation," On-Line Testing Workshop, 2001. Proceedings. Seventh International, pp. 80 -86, July 2001.

[12] Mobile Electronic Transaction, "PTD Definition Version 2.0," Oct. 2002.

[13] S. William, “Cryptography and Network Security Principles and Practices,” Pearson Education, 3rd edition, 2003.

[14] S. Palnitker, “Verilog HDL: A Guide to Digital Design and Synthesis ” ,Prentice-Hall Inc., 1996.

[15] 賴溪松, 韓亮, 張真誠, “近代密碼學及其應用,” 旗標,2003

[16] 蔡志堅,“符合 PC/SC 個人電腦與智慧卡規格智慧卡安全機制之研究”,國立成功大學工程科學研究所碩士論文,2000 年 6 月

[17] 雷欽隆,尹奇恩,”智慧卡付款趨勢,”資訊安全通訊,第七卷,第四期,2001 年,第 46~60 頁

[18] 楊伏夷, “可證明安全的身份認證與存取控制 -- 使用 IC 智慧卡,” 國立中興大學應用數學系博士論文,2004 年 6 月

[19] CIC 訓練課程, Verilog Training Manual,國家晶片系統設計中心, 2002.

[20] 鄭信源 編著, “ Verilog 硬體描述語言數位電路-設計實務,” 儒林圖書公司, 2002 年 9 月再版, SIM 658

[21] 黃英叡等 編譯, “ Verilog 硬體描述語言(VerilogHDL: A Guide to Digital Design and Synthesis) ,” 全華科技圖書公司, 民國 88 年

[22] 財團法人資訊工業策進會 http://www.iii.org.tw

[23] OpenCard http://www.opencard.org

[24] 政府機關資訊處理共通規範 http://www.rdec.gov.tw/mis/standard92/ipcs/guide.htm

[25] 政府憑證管理中心(Government Certification Authority) http://www.pki.gov.tw/

[26] Information Technology Laboratory of NIST http://www.itl.nist.gov/

[27] RSA Security, Inc. http://www.rsasecurity.com/

[28] SET, Secure Electronic Transction , http://www.setco.org/

[29] Secure Sockets Layer(SSL) , http://wp.netscape.com/eng/ssl3/ssl-toc.html

[30] OPENCORES http://www.opencores.com/