科技部資訊安全實務研發計畫系統測試報告

System Integration and Testing Document

先進駕駛輔助系統之雲端輔助設計優化
子計畫三：植基於免疫系統之安全的物聯網應用開發方法 -
以先進駕駛輔助系統為例

MOST 105-2221-E-156 -004

主持人蘇維宗
真理大學資訊工程學系(所)

Department of Computer Science and Information Engineering
Aletheia University, Taiwan

2017/06/10

目錄

圖目錄

表目錄

版本更變記錄

修改日期	版本	文件狀態	描述	負責人
2017/06/10	1.0	Release	測試報告完稿確認	蘇維宗
2017/6/5	0.2	Beta	加入效能測試結果	陳韋丞
2017/05/25	0.1	Draft	初稿	陳韋丞

1. Introduction

本子計畫第一年要研究的議題是如何讓物聯網裝置所產生的資料能夠在確保安全性與隱私權的條件下細緻地(fine-grained)給適當的使用者進行存取。由於未來物聯網裝置數量急遽增加的情況下，傳統的存取控制清單(Access Control List, ACL)方法將因為其延展性(scalability)不佳而不適合用來在物聯網中實現存取控制。從近代資訊安全技術的發展來看，植基於身份的加密機制(Identity-Based Encryption，以下簡稱 IBE)似乎是一個符合對透過物聯網交換的資料進行加密並同時提供存取控制的解決方案。然而，傳統的 IBE 要運用在物聯網的環境下仍會遇到許多問題，例如裝置身份與安全性金鑰的管理。

本次測試將依據適用於物聯網之植基於身份加密方法(IoT Identity-Based Encryption，I^2BE)之軟體模組，主要為圖 1 的輕量化 IBE 引擎(Light-Weight IBE Engine)，進行功能與效能測試。本子計畫所開發之輕量化 IBE 引擎之原始碼(bee-cpabe-sdk-0.1 專案)目前已發布在 GitHub 平台上(https://github.com/adas-most/adas-security-mqtt)。

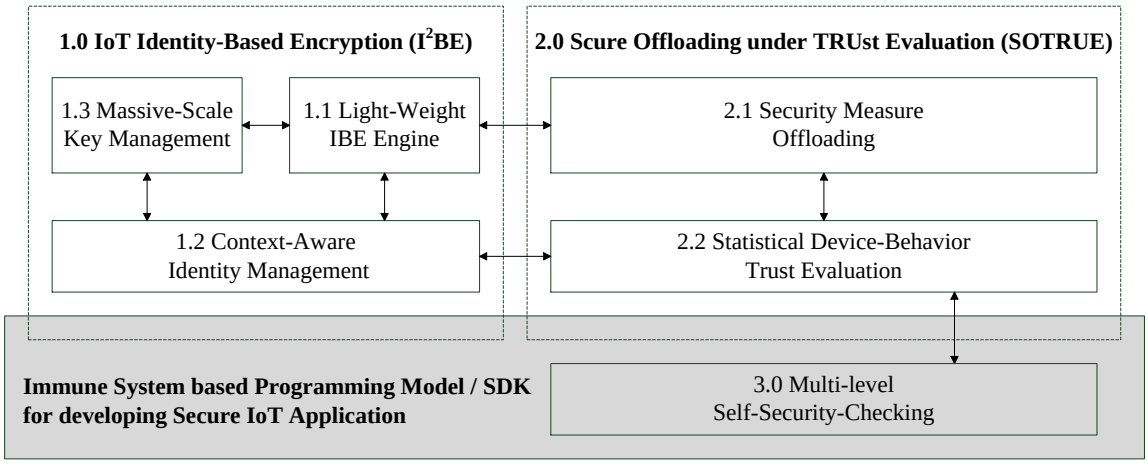

圖 1. 本計畫擬開發系統之架構圖(規劃三年完成)

1.1. Scope of Testing

本文件內容將依據系統需求規格書與系統設計文件，描述關於整合測試的相關計畫與內容。在確認本系統整合前，必須先確認所有的設計之子系統均能正確無誤的運作，

因此著重於整合系統測試(Integration Test)及接受測試(Acceptance Test)，並透過此文件之描述與實踐，達到順利進行測試工作之目的。

1.2. Acceptance Criteria

本測試計劃需要滿足下列的測試接受準則：

- 本系統需要對所有列為必要(Critical、Important、Desirable)之需求作完整測試。
- 測試程序需要依照本測試計畫所訂定的程序進行，所有測試結果需要能符合預期測試結果方能接受。
- 以測試案例為單位，當測試未通過時，需要進行該單元的測試，其接受的準則與前一項規定相同。

2. Testing Environment

2.1. Operational Environment

本文件主要測試一支援屬性加密 4 個項目，即 1 個系統含有 4 個功能，其不同之操作概念測試，說明如後。Bee 加密系統如圖 1，使用者可以依據本身需求要用到的功能以方便的方式來達到保護資料安全性的目的。

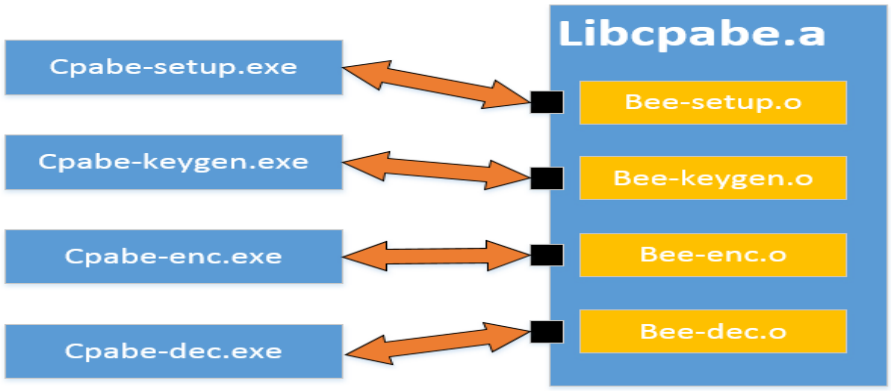

圖 2. 整體軟體架構圖

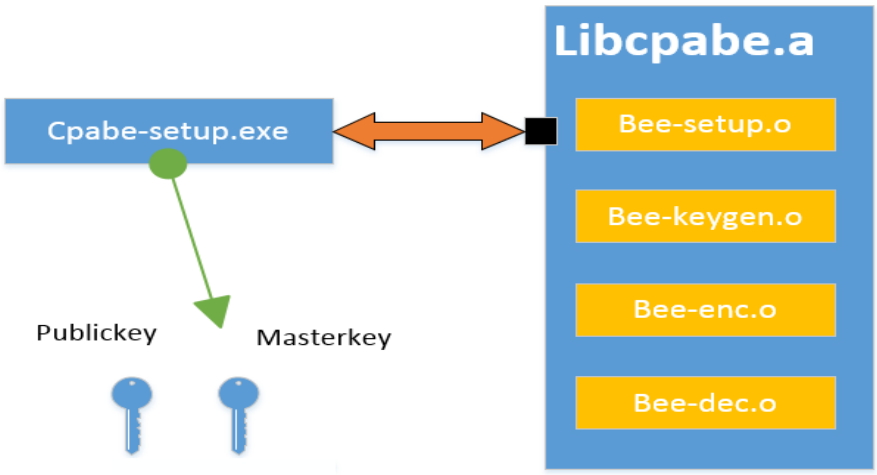

圖 3. setup()功能架構圖

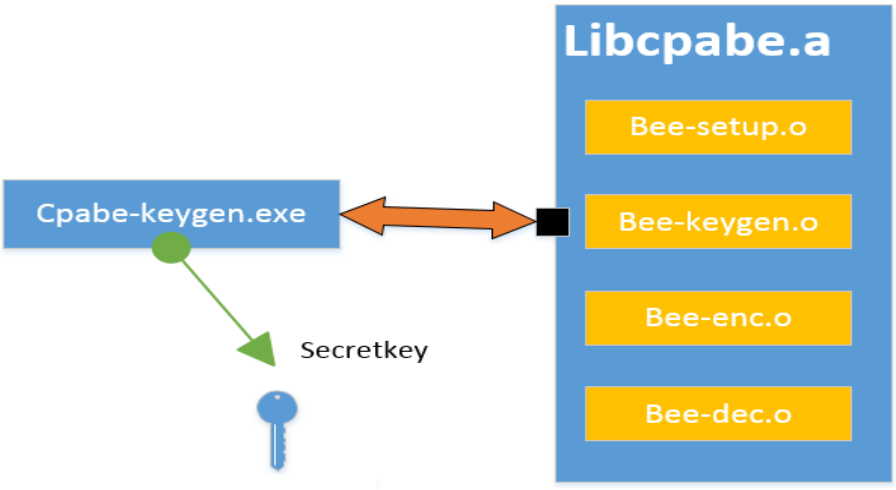

圖 4. keygen()功能架構圖

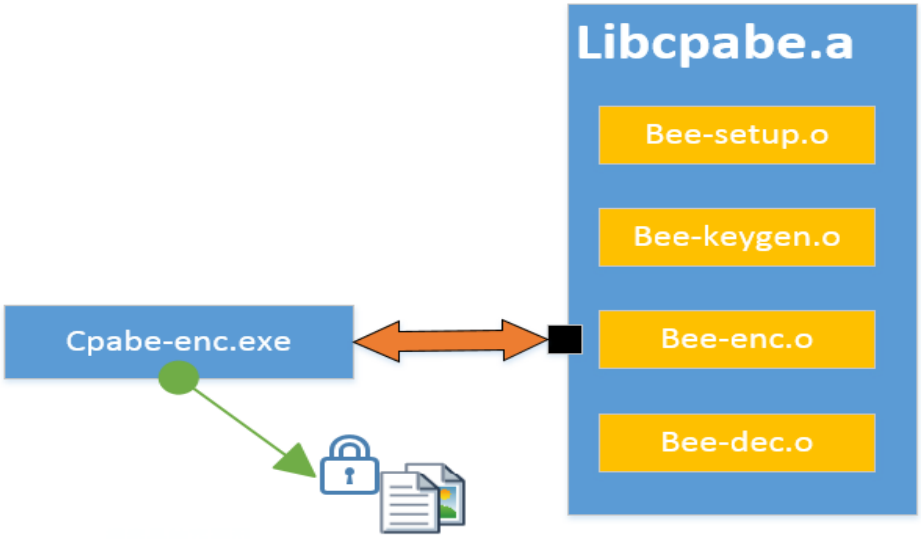

圖 5. enc()功能架構圖

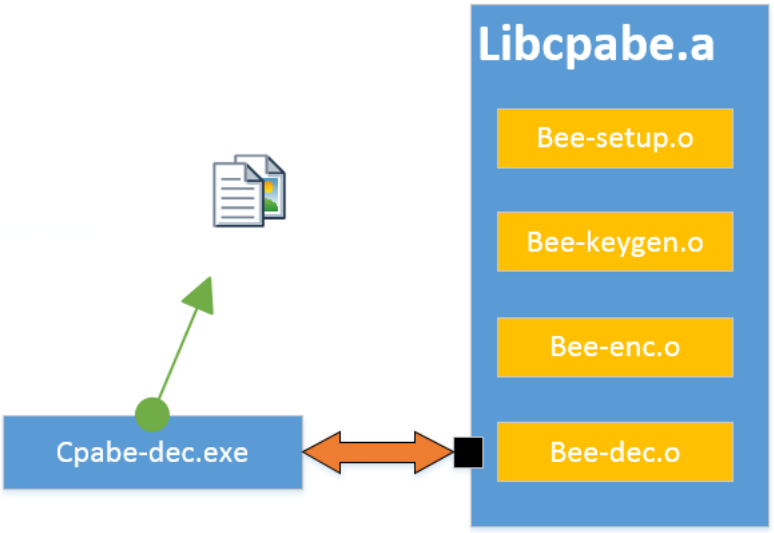

圖 6. dec()功能架構圖

2.2. Hardware Specification

依據測試環境內容，關於測試環境所需的硬體規格說明，如表格 1 所示:

表格 1. 硬體規格

CPU	Intel(R)Core(TM) i5-4200H CPU@ 2.80GHz 2.79GHz
RAM	8.00GB
System type	64Bit

2.3. Software Specification

依據測試環境內容，關於測試環境所需的軟體規格說明，如下所示:

- CentOS Linux (64 bits)與相關開發套件(如 gcc、make 等)
- 基本函式庫套件，包含 m4、flex、bison、libssl-dev、libgmp-dev(>4.0.0)與 glib(>2.0.0)
- 屬性加密函式庫套件(libpbc、libbswabe、與 cpabe)
- 本計畫開發之函式庫套件(libcpabe)

2.4. Test Data Sources

關於測試所需的測試資料，如下

- 多種文件資料，如*.txt 與*.doc 等。
- 多種圖像資料，如*.jpg 等。
- 多種影音資料，如*.mpg,與*.mp3。

3. Testing Schedule, Procedure, and Responsibility

3.1. Testing Schedule

根據專案執行規劃書計畫書，測試時程 106 年 4 月起至 106 年 5 月止，詳細時程說明如下。

- 時程
  - 各子功能之內部元件整合測試(106/4/10 ~ 106/4/23)
  - 系統功能測試(106/4/24 ~ 106/5/14)
  - 系統效能測試(106/5/15 ~ 106/6/3)
- 查核點
  - 各子功能之內部元件整合測試確認(106/4/24)
  - 系統功能測試確認(106/5/15)
  - 系統效能測試確認(106/6/4)

3.2. Testing Procedure

3.2.1. Subsystems Validation

各子功能之內部元件測試，由各子系統開發負責人員完成，在此我們著重於所有子功能完成後之整合測試。

3.3. Personnel and Responsibility

表格 2. 測試人員角色與責任

Testing Cases	Testing Target	Personnel
FT01	setup()	蔡鎧澤
FT02	keygen()	賴英綸
FT03	enc()	魏祥宇
FT04	dec()	魏祥宇
PT01	setup()執行效能	陳韋丞
PT02	keygen()執行效能	陳韋丞
PT03	enc()執行效能	陳韋丞
PT04	dec()執行效能	陳韋丞

4. Test Cases

4.1. Integration Testing Cases

4.1.1. FT01 Function Test Case

目的: 驗證 setup 功能且完整的獲得執行的結果。

表格 3. setup()功能測試案例說明

Identification	FT01
Name	驗證 setup()功能
Assumption	整合(libcpabe-01.a 已整合)
Pre-condition	整合(libcpabe-01.a 已整合)
Target	呼叫 setup()函式能夠正確執行或在發生錯誤時回報錯誤代碼
Test Object	setup(公開金鑰產生路徑, 主要金鑰產生路徑)
Severity	Critical
Test Source	TS 1. 兩個路徑都存在 setup(“./pubkey”,”./mstkey”) TS 2. 公開金鑰的路徑不存在 setup(“/error/pubkey”,”./mstkey”) TS 3. 主要金鑰的路徑不存在 setup(“./pubkey”,”/error/mstkey”) TS 4. 公開金鑰與主要金鑰的路徑都不存在 setup(“/error/pubkey”,”/error/mstkey”)
Instructions	Step 1. 編輯 bee-setup 程式並依 TS1 ~ TS4 修改 setup()函式的參數 Step 2.清除資料(make clean) Step 3.重新編譯並產生執行檔(make) Step 4.執行 bee-setup 程式(./bee-setup)並觀察執行結果
Expected Result	TR 1. Return 0 顯示 SETUP SUCCESS，在指定目錄產生公開金鑰與主要金鑰 TR 2. Return -1 顯示 can't write file: /error/pubkey SETUP ERROR! TR 3. Return -1 顯示 can't write file: /error/mstkey SETUP ERROR! TR 4. Return -1 顯示 can't write file: /error/pubkey SETUP ERROR!

4.1.2. FT02 Function Test Case

目的: 驗證 keygen 功能且完整的獲得執行的結果。

表格 4. keygen()功能測試案例說明

Identification	FT002
Name	驗證 keygen()功能
Assumption	整合(libcpabe-01.a 已整合)已驗證完 setup()
Pre-condition	整合(libcpabe-01.a 已整合)已產生公開金鑰與主要金鑰
Target	呼叫 keygen()函式能夠正確執行或在發生錯誤時回報錯誤代碼
Test Object	keygen(私密金鑰路徑, 公開金鑰路徑, 主要金鑰路徑, 屬性個數, …)
Severity	Critical
Test Source	TS 1. 正確產生私密金鑰 keygen(“./seckey”, ”./pubkey”, ”./mstkey”, 4, ”jackie”, ”boy”, ”a = 12”, ”s = 1000”) TS 2. 私密金鑰路徑不存在 keygen(“/error/seckey”, ”./pubkey”, ”./mstkey”, 4, ”jackie”, ”boy”, ”a = 12”, ”s = 1000”) TS 3. 公開金鑰不存在 keygen(“./seckey”, ”/error/pubkey”, ”./mskey”, 4, ”jackie”, ”boy”, ”a = 12”, ”s = 1000”) TS 4. 主要金鑰不存在 keygen(“./seckey”, ”./pubkey”, ”/error/mstkey”, 4, ”jackie”, ”boy”, ”a = 12”, ”s = 1000”) *TS 5. 屬性個數與實際給予的屬性數量不一致 keygen(“./seckey”, ”./pubkey”, ”./mstkey”, 3, ”jackie”, ”boy”, ”a = 12”, ”s = 1000”) TS 6. 未給予屬性數量 keygen(“./seckey”, ”./pubkey”, ”./mstkey”, ”jackie”, ”boy”, ”a = 12”, ”s = 1000”) TS 7. 未給予屬性參數 keygen(“./seckey”, ”./pubkey”, ”./mstkey”, 3, ””) TS 8. 參數屬性亂設定 keygen(“./seckey”, ”./pubkey”, ”./mstkey”, 1, ”swr**** 2121= ”)
Instructions	Step 1. 編輯 bee-keygen 程式並依 TS1 ~ TS4 修改 keygen()函式的參數 Step 2. 清除資料(make clean) Step 3. 重新編譯(make) Step 5. 執行 bee-keygen 程式(./bee-keygen)並觀察執行結果
Expected Result	TR 1. Return 0 顯示 KEYGEN SUCCESS，在指定目錄產生私密金鑰 TR 2. Return -1 顯示 can't write file:/error/seckey KEYGEN ERROR! TR 3. Return -1 顯示 can't read file:/error/pubkey KEYGEN ERROR! TR 4. Return -1 顯示 can't read file:/error/mstkey KEYGEN ERROR! TR 5. Return 0 顯示 KEYGEN SUCCESS，在指定目錄產生私密金鑰 TR 6. Segmentation fault(core dumped) TR 7. Segmentation fault(core dumped) TR 8. Return -1 顯示 error parsing attributes” swr**** 2121=” (note that numerical attributes are unsigned integers) KEYGEN ERROR!

4.1.3. FT03 Function Test Case

目的: 驗證 enc 功能且完整的獲得執行的結果。

表格 5. enc()功能測試案例說明

Identification	FT03
Name	驗證 enc()功能
Assumption	整合(libcpabe-01.a 已整合)已驗證完 setup()與 keygen()
Pre-condition	整合(libcpabe-01.a 已整合)已產生公開金鑰、主要金鑰與私密金鑰
Target	呼叫 enc()函式能夠正確執行或在發生錯誤時回報錯誤代碼
Test Object	keygen(公開金鑰路徑, 明文檔路徑, 存取政策, 密文檔路徑、公開金鑰路徑)
Severity	Critical
Test Source	TS 1. 對 jpg 檔案進行加密 enc(“./pubkey”, ”./cat.jpg”, ”jackie and s >= 1000”, ”default”) TS 2. 對 xlsx 檔案進行加密 enc(“./pubkey”, ”./105.xlsx”, ”jackie and s >= 1000”, ”default”) TS 3. 對 mp3 檔案進行加密 enc(“./ pubkey”, ”./b.mp3”, ”jackie and s >= 1000”, ”default”) TS 4. 對 mp4 檔案進行加密 enc(“./pubkey”, ”./a.mp4”, ”jackie and s >= 1000”, ”default”) TS 5. 對 docx 檔案進行加密 enc(“./pubkey”, ”./Bee.docx”, ”jackie and s >= 1000”, ”default”) TS 6. 公開金鑰不存在 enc(“/error/pubkey”, ”./Bee.docx”, ”jackie and s >= 1000”, ”default”) TS 7. 加密參數為空值 enc(“./pubkey”, ”./Bee.docx”, ””, ”default”) TS 8. 加密參數設置錯誤 enc(“./pubkey”, ”./Bee.docx”, ”jackie ands >= 1000”, ”default”) TS 9. 加密時指定輸出檔案名稱 enc(“./pubkey”, ”./cat.jpg”, ”jackie and s >= 1000”, “./catt.jpg”)
Instructions	Step 1. 編輯 bee-enc 程式並依 TS1 ~ TS9 修改 enc()函式的參數 Step 2. 清除資料(make clean) Step 3. 重新編譯(make) Step 4. 執行 bee-enc 程式(./bee-enc)並觀察執行結果
Test Result	TR 1. Return 0 顯示 ENC SUCCESS，在指定目錄產生 cat.jpg.cpabe TR 2. Return 0 顯示 ENC SUCCESS，在指定目錄產生 105.xlsx.cpabe TR 3. Return 0 顯示 ENC SUCCESS，在指定目錄產生 b.mp3.cpabe TR 4. Return 0 顯示 ENC SUCCESS，在指定目錄產生 a.mp4.cpabe TR 5. Return 0 顯示 ENC SUCCESS，在指定目錄產生 Bee.docx.cpabe TR 6. 顯示 can't read file:/error/pubkey ENC FAIL TR 7. 顯示 error parsing policy: syntax error Segmentation fault(core dumped) TR 8. 顯示 error parsing policy: syntax error Segmentation fault(core dumped) TR 9. Return 0 顯示 ENC SUCCESS，在指定目錄產生 catt.jpg(加密過)

4.1.4. FT04 Function Test Case

目的: 驗證 dec 功能且完整的獲得執行的結果。

表格 6. dec()功能測試案例說明

Identification	FT04
Name	驗證 dec()功能
Assumption	整合(libcpabe-01.a 已整合)已驗證完 setup()、keygen()與 enc()
Pre-condition	整合(libcpabe-01.a 已整合)已產生公開金鑰、主要金鑰、私密金鑰與密文
Target	呼叫 dec()函式能夠正確執行或在發生錯誤時回報錯誤代碼
Test Object	dec(公開金鑰路徑, 私密金鑰路徑, 密文檔路徑)
Severity	Critical
Test Source	TS 1. 對 jpg 檔案進行解密 dec(“./pubkey”, ”./seckey”, ”./cat.jpg.cpabe”) TS 2. 對 xlsx 檔案進行解密 dec(“./pubkey”, ”./sekey”, ”./105.xlsx.cpabe”) TS 3. 對 mp3 檔案進行解密 dec(“./pubkey”, ”./seckey”, ”./b.mp3.cpabe”) TS 4. 對 mp4 檔案進行解密 dec(“./pubkey”, ”./seckey”, ”./a.mp4.cpabe”) TS 5. 對 docx 檔案進行解密 dec(“./pubkey”, ”./seckey”, ”./Bee.docx.cpabe”) TS 6. 公開金鑰不存在 dec(“/error/pubkey”, ”./seckey”, ”./cat.jpg.cpabe”) TS 7. 私密金鑰不存在 dec(“./pubkey”, ”/error/seckey”, ”./cat.jpg.cpabe”) TS 8. 私密金鑰屬性不符合存取政策 dec(“./pubkey”, ”./seckey2”, ”./cat.jpg.cpabe”) TS 9. 對加密時指定輸出檔案名稱進行解密 dec(“./pubkey”, ”./seckey”, ”./catt.jpg”)
Instructions	Step 1. 編輯 dec 程式並依 TS1 ~ TS9 修改 dec()函式的參數 Step 2. 清除資料(make clean) Step 3. 重新編譯(make) Step 5. 執行 bee-dec 程式(./bee-dec)並觀察執行結果
Test Result	TR 1. Return 0 顯示 DEC SUCCESS，在指定目錄解密成 cat.jpg.cpabe TR 2. Return 0 顯示 DEC SUCCESS，在指定目錄解密成 105.xlsx.cpabe TR 3. Return 0 顯示 DEC SUCCESS，在指定目錄解密成 b.mp3.cpabe TR 4. Return 0 顯示 DEC SUCCESS，在指定目錄解密成 a.mp4. TR 5. Return 0 顯示 DEC SUCCESS，在指定目錄解密成 Bee.docx. TR 6. Return -1 顯示 can't read file:/error/pubkey DEC FAIL TR 7. Return -1 顯示 can't read file:/error/seckey DEC FAIL TR 8. Return -1 顯示 cannot decrypt, attributes in key do not satisfy policy DEC FAIL TR 9. Return 0 顯示 DEC SUCCESS，在指定目錄解密 catt.jpg

4.1.5. PT01 Performance Test Case

目的：測試 setup 的平均執行時間。

表格 7. setup()效能測試案例說明

Identification	PT01
Name	測試 setup()的平均執行時間
Tested target	setup()的執行效能
Severity	Critical
Instructions	Step 1. 執行 FT01 Step 2. 紀錄執行一次的時間 Step 3. 重複執行 100 次 Step 4. 計算平均值行時間
Expected result	執行 setup()的平均時間必須小於 1 秒
Cleanup	None

4.1.6. PT02 Performance Test Case

目的： 測試 keygen()在不同屬性個數時的執行時間

表格 8. keygen()效能測試案例說明

Identification	PT02
Name	測試 keygen()在不同屬性個數時的執行時間
Tested target	setup()的執行效能
Severity	Critical
Instructions	Step 1. 執行 FT02 並將屬性個數設定為 n (n 從 1 開始) Step 2. 紀錄執行一次的時間 Step 3. 重複執行 100 次 Step 4. 計算平均值行時間 Step 5. 如果 n <= 30，n = n + 1 回到 Step 1
Expected result	屬性個數 30 以下時執行 keygen()的平均時間必須小於 2 秒
Cleanup	None

4.1.7. PT03 Performance Test Case

目的：測試在不同的存取政策樹高度時，enc()的平均執行時間。

表格 9. enc()效能測試案例說明

Identification	PT03
Name	測試 enc()在不同存取政策樹高度時的執行時間
Tested target	enc()的執行效能
Severity	Critical
Instructions	Step 1. 執行 FT03 並將存取政策樹高度設定為 n (n 從 1 開始) Step 2. 紀錄執行一次的時間 Step 3. 重複執行 100 次 Step 4. 計算平均值行時間 Step 5. 如果 n <= 5，n = n + 1 回到 Step 1
Expected result	存取政策樹高度 5 以下時執行 enc()的平均時間必須小於 1 秒
Cleanup	None

4.1.8. PT04 Performance Test Case

目的：測試在不同的存取政策樹高度時，dec()的平均執行時間。

表格 10. dec()效能測試案例說明

Identification	PT04
Name	測試 dec()在不同存取政策樹高度時的執行時間
Tested target	dec()的執行效能
Severity	Critical
Instructions	Step 1. 執行 FT04 並將存取政策樹高度設定為 n (n 從 1 開始) Step 2. 紀錄執行一次的時間 Step 3. 重複執行 100 次 Step 4. 計算平均值行時間 Step 5. 如果 n <= 5，n = n + 1 回到 Step 1
Expected result	存取政策樹高度 5 以下時執行 dec()的平均時間必須小於 1 秒
Cleanup	None

5. Test Results and Analysis

5.1. Functional Test

Test Case	Result (Pass / Fail)	Comment
FT01	Pass	
FT02	Conditional Pass	TR 5 指定屬性數量與給定屬性數量不同也可以產生金鑰 TR 6 記憶體錯誤 TR 7 記憶體錯誤
FT03	Pass	
FT04	Conditional Pass	TR 7 記憶體錯誤 TR 8 記憶體錯誤
Rate	90%	

5.2. Performance Test

Test Case	Expected Result	Test Result	Comment
PT01	< 1 秒	Pass	詳細測試結果請參閱圖 7
PT02	< 2 秒	Pass	詳細測試結果請參閱圖 8
PT03	< 1 秒	Conditional Pass	詳細測試結果請參閱圖 9 當樹高為 5 時會超過 1 秒
PT04	< 1 秒	Pass	詳細測試結果請參閱圖 10

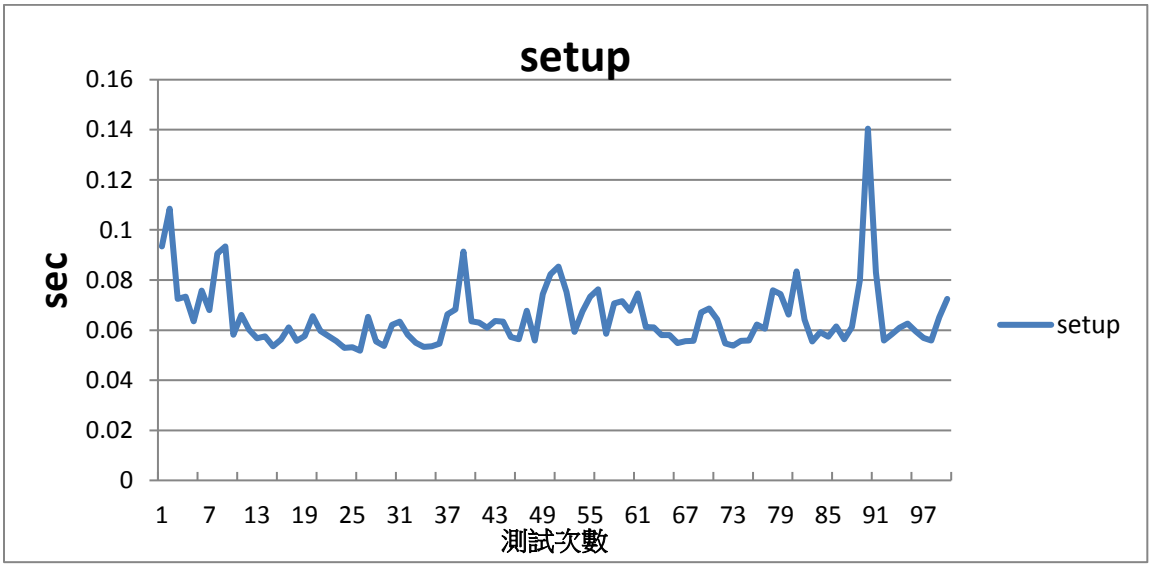

圖 7. setup()效能測試圖

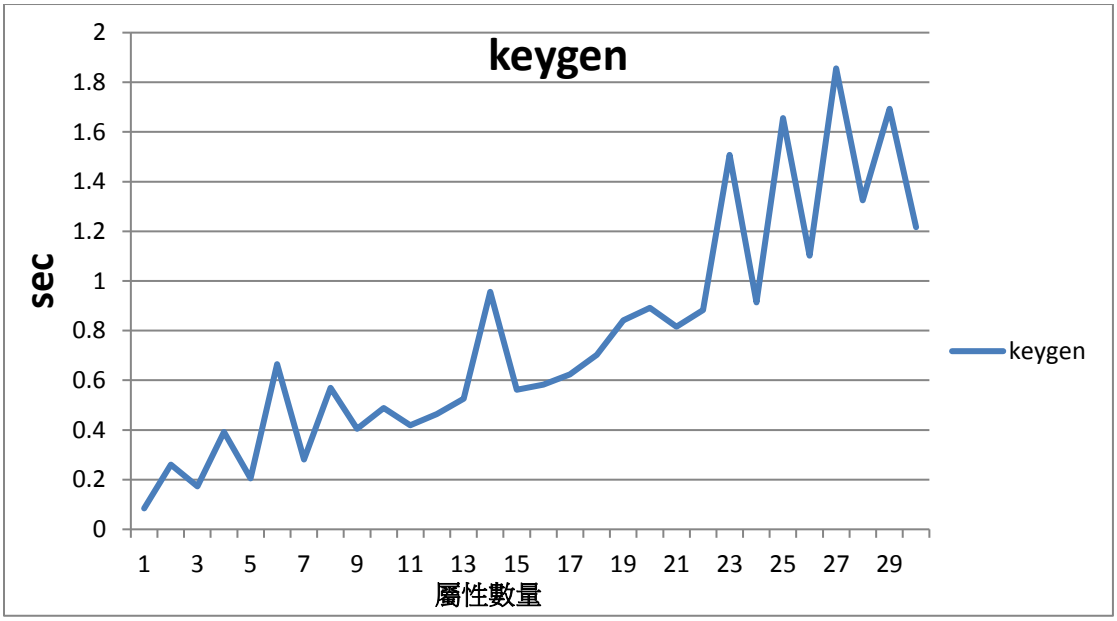

圖 8. keygen()效能測試圖

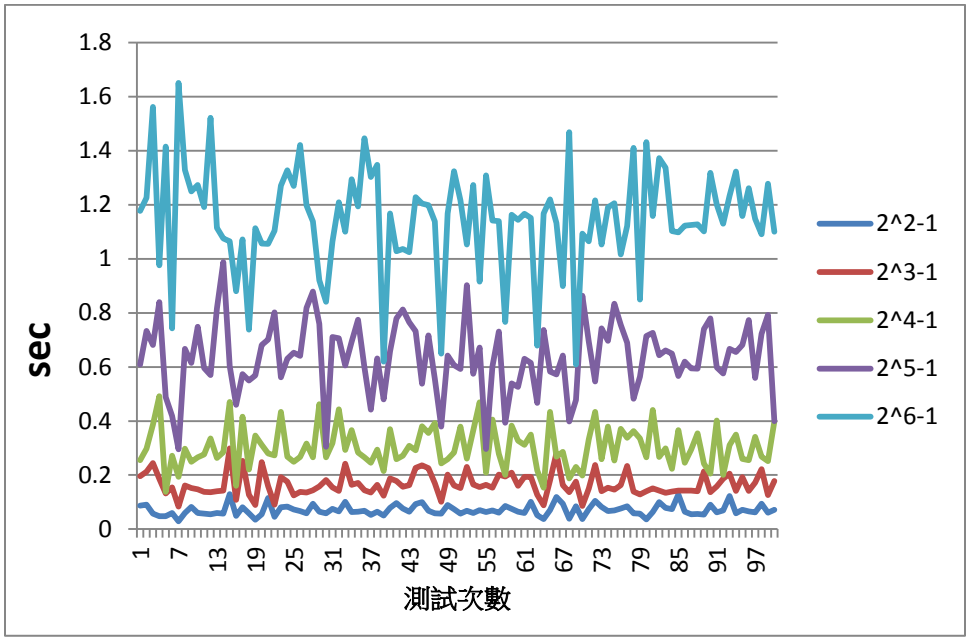

圖 9. enc()效能測試圖

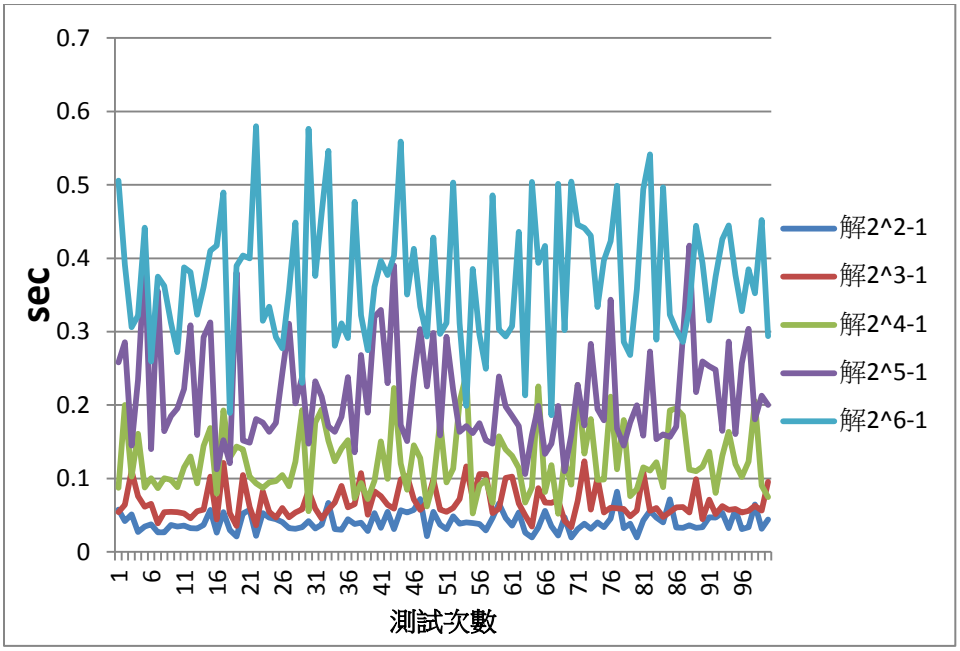

圖 10. dec()效能測試圖

Appendix A: Traceability

表格 11. 系統模組與測試案例的追溯矩陣

Test Cases / Module	FT01	FT02	FT03	FT04	PT01	PT02	PT03	PT04
1.1 Light-Weight IBE Engine	◎	◎	◎	◎	◎	◎	◎	◎
1.2 Context-Aware Identity Management		◎						
1.3 Massive-Scale Key Management	◎	◎			◎	◎		