Term Project Report

Hardware Security Module for CryptBox (HSMCB) for File Encryption

04 May 2017

BIDARE VENKATESH GURUPRASAD, UCO:459196
GILLELA MARUTHI, UCO:459206
GOVIND PRASHANTH REDDY, UCO:459207
TANGUTURI RAMU, UCO: 459204

Faculty of Informatics, Masaryk University, Brno, Czech Republic
http://www.fi.muni.cz

Table of Content

1. Introduction

As a part of the term project, to implement the secure channel functionality using a Hardware Security Model (for ex., Java card), an extensive search for a basic framework which handles the user password was carried out (Jpass, Cryptobox, UPM). Subsequent to the study and evaluation of several frameworks, the crypt box was found to be suitable for the proposed activity. The cryptbox was downloaded from the following website:

https://www.codeproject.com/articles/546069/cryptbox

2. Objective of the project

Following are the objectives of the goal:

- *Selection of existing open-source application which requires password or key from user*
- *Creation of applet which will provide secure key storage or processing*
- *Creation of secure channel between original app and car*

3. Git Repository location

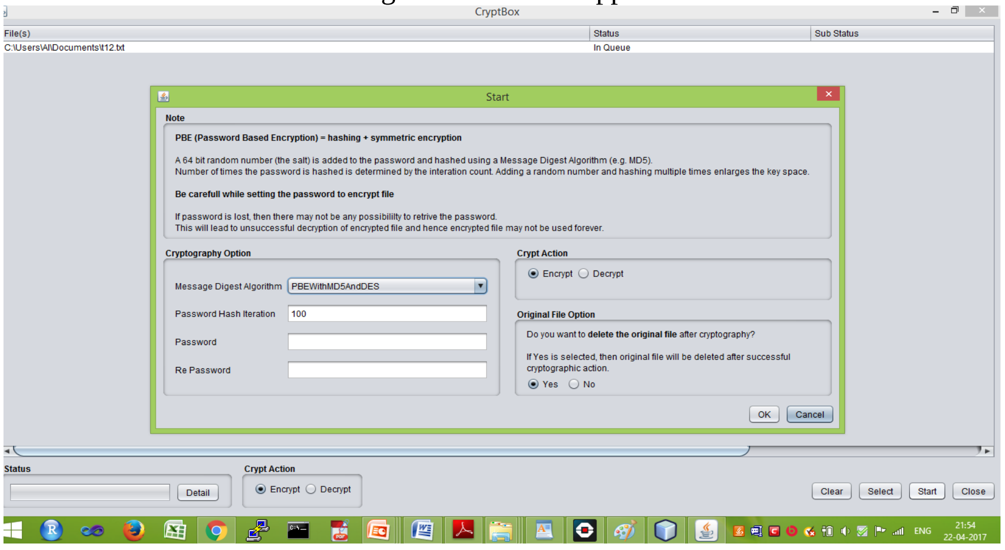
https://github.com/ramucs108/HSMCryptbox

4. Brief Description

As a first step for project development, an analysis of the Cryptbox open source framework using code walk through (16 man-hours effort (approx)) was carried out. Complete java source code was analyzed . During the analysis comments were made in source file during code walkthrough. It was observed that several Input validations were not done.

4.1. Original Cryptbox frame work functionalities:

- The original cryptbox project is based on File Encryption and Decryption using Password Based Encryption (PBE) where in standard APIs of the Java Crypt libraries (for ex., javax.crypto.Cipher, javax.crypto.SecretKey, javax.crypto.SecretKeyFactory, javax.crypto.spec.PBEKeySpec etc.,) are being used.
- The user selects Mode (Enc/dec) and File to be Enc/Dec. Also, user enters Password & re-assword and PDKDF number of Iterations. After taking these inputs from the user, the application generates a Random Number (Salt) – 8 bytes
- It uses password and salt to generate PBKDF for user selected number of times using "PBEWithMD5AndDES" class of java.
- Finally, the application Encrypts/Decrypts the user selected file using DES algo.
- The screen shot of basic original framework application is as shown:



File(s)	Status	Sub Status
C:\Users\Al\Documents\t12.txt	In Queue	

Note

PBE (Password Based Encryption) = hashing + symmetric encryption

A 64 bit random number (the salt) is added to the password and hashed using a Message Digest Algorithm (e.g. MD5).
Number of times the password is hashed is determined by the interation count. Adding a random number and hashing multiple times enlarges the key space.

Be carefull while setting the password to encrypt file

If password is lost, then there may not be any possibililty to retrive the password.
This will lead to unsuccessful decryption of encrypted file and hence encrypted file may not be used forever.

Cryptography Option

Message Digest Algorithm: PBEWithMD5AndDES
Password Hash Iteration: 100
Password:
Re Password:

Crypt Action

[✓ Encrypt] Decrypt

Original File Option

Do you want to **delete the original file** after cryptography?

If Yes is selected, then original file will be deleted after successful cryptographic action.

[✓ Yes] No

Status

Crypt Action

[✓ Encrypt] Decrypt

4.2. Hardware Security Module for CryptBox (HSMCB) functionalities:

4.2.1. High level Architecture:

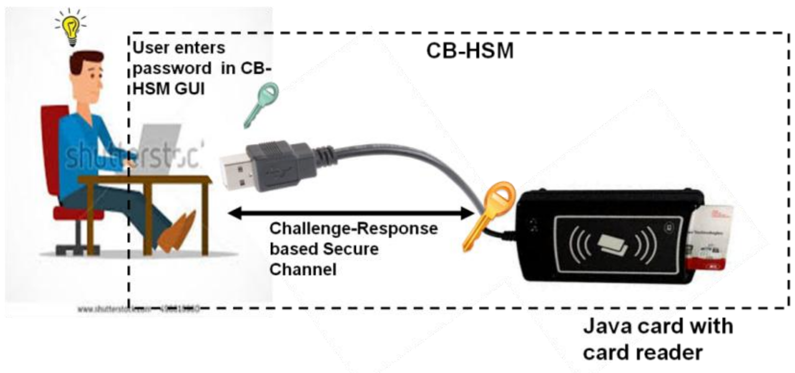

User enters password in CB-HSM GUI
CB-HSM
Challenge-Response based Secure Channel
Java card with card reader

Fig 1. High level Architecture of HSMCB

As shown in the Fig 1, the original Cryptbox framework is enhanced to realize the HSMCB functionalities. The user host machine (Laptop/Desktop) is integrated with a Java card reader having a java card which works as Hardware Security Model (HSM).

4.2.2.Development strategy:

The HSMCB was developed in two phases, first using simulator running completely on simulator and finally running on the trusted HSM i.e., Java card. The features of the HSMCB are described below:

- Secure Channel Communication
  - Cryptographic primitives used
    - AES128, MAC, Random Numbers
    - Padding of the message is carried out to make multiple of 16 bytes
  - Security mechanisms
    - Authentication
    - Confidentiality
    - Integrity
    - Reply Protection
  - Assumptions
    - Attacker can eavesdrop between PC and Java Card
    - The final encryption and decryption of the file will be carried out by PC application

4.2.3. Secure communication Channel message details:

The Table-1 gives the details of the messages flowing between host and Java card and vice-versa.

Sl. No.	Message Name	From	To	Remarks
1.	Challenge1 Message	Host	JC	Freshness ensured by random value RanHost of Host
2.	(Challenge1 Response + Challenge2) Message	JC	Host	Freshness ensured by random value RanJCof JC
3.	Challenge2 Response	Host	JC	Freshness ensured by random value RanHost+1 of Host
4.	ENC_KEY_FOR_FILE Response Message	JC	Host	Finally JC sends the key to be used for enc/dec the file

Table 1. Messages between Host and JC

4.2.4. Secure communication Channel message descriptions:

The Table-2 and Table-3 gives the details of the individual messages flowing between host and Java card and vice-versa.

Table-2 gives the messages for initial transaction between Host and JC wherein host sends its random number to JC and JC increments it and adds its own random number.

MESSAGE DETAILS	Action Taken
1. Message from Host to JC	Host computes the encryption of (PIN+RanHost) using public key of Java Card and adds the MAC of the packet. JC receives the packet, computs MAC and compares with incoming MAC. If success, sends message 2
2. Message from JC to Host	JC computes the encryption of (PIN+RanJC) and adds MAC of the packet. Host verifies the H_RanVal1 for negation, it negates J_RanVal2 and replies with MSG_ID 0X03

Table 2. Messages between Host and JC - stage1

Table-3 gives the messages transaction between JC and Host wherein JC verifies whether the host has incremented the random number. If verification is success, it sends the final encryption key to be used for file encryption.

MESSAGE DETAILS	Action Taken
3. Message from Host to JC	Host computes the encryption of (PIN+RanHost) using public key of Java Card and adds the MAC of the packet. JC receives the packet, computs MAC and compares with incoming MAC. If success, sends message 2
4. Message from JC to Host	JC computes the encryption of (ENC $_{K1}$(PIN + (RanHost+3) + +ENC_KEY)) and adds MAC of the packet. Host verifies the RanHost+3 and accepts ENC_KEY

Table 3. Messages between Host and JC - stage2

4.2.5. Description of major modules:

- Init_javacardStart_Encryption
  - Input parameter:
    - Nil
  - Output parameter:
    - Return 1 if initialization is success, else throws exception

    - Description:
        - This module initializes the java card for communication using the function connect to takes the FILE_ENC_DEC_KEY provided by the simulator/Java card subsequent to secure communication, opens the user selected file, carries out encryption or decryption as the case may be.

- JavaCardProtocol
    - Input parameter:
        - Nil
    - Output parameter:
        - Return 1 for success, 0 for failure
    - Description:
        - This module establishes the secure channel communication between host and java card using multiple transaction as described in the above sections message format. Finally, it returns if the communicaiton is success, else returns 0.

- decryptJCResponse()
    - Input parameter:
        - byte buffer returned from Java card
    - Output parameter:
        - 1 for success, 0 for failure
    - Description:
        - This module verifies the returned response from the java card and checks for various conditions viz., whether the random number is incremented by 1 or not. If success, returns 1 else returns 0.

- genPassword(APDU apdu)
    - Input parameter:
        - APDU containing the input random number from the host
    - Output parameter:
        - Return the final password if success, else throws the error
        - 0 for failure, 1 for success
    - Description:
        - This module takes the random number from the incoming APUD subsequent to secure channel communicaton. It finally generates the encrypted key. This key will be sent host which uses this key for the encryption/decryption of the user selected file.
        -
- computeMAC(APDU apdu)
    - Input parameter:
        - APDU containing the input for which MAC is to be computed
    - Output parameter:
        - if success, returns the final MAC appended to APDU, else 0 for failure
    - Description:
        - This module takes the data payload from the incoming APDU and computes the MAC of the same using inbuilt MAC library. If the computation is success, it appends the MAC to the end of the APDU and returns. Else, it returns 0 for failure.
    -

5. Conclusion

The Hardware Security Module using CryptBox (HSMCB) is designed, developed and tested successfully on a Java Card. The user selected file is tested successfully for both encyption and decryption.