PV204 Project Presentation

Ananya Chatterjee, UCO:459203

Rajesh Kumar, UCO:459195

Rajesh Chandrakant Mehta, UCO:459194

Github Repo: **https://github.com/rktiff/pv204javacard**

Introduction

- Chosen Application: Universal Password Manager (UPM)
  - http://upm.sourceforge.net/
  - Author: Adrian Smith
  - Open-source password manager written in Java
  - User credentials, passwords and notes are stored in credential database (CDB) files
  - CDB are encrypted using password based encryption
  - Password (max. 16B) is taken from User to generate the Key
  - UPM uses AES-CBC-256 for encryption
  - Encryption Key generated from Master Password + Random Salt

Motivation

- Key is generated and stored locally
    - Physical access to PC gives full control to attacker
    - Single factor authentication
- Encryption of Credential Data Base (CDB)
    - Relies completely on strength of master password (16 byte)
    - Key generation algorithm is known to public
    - Attempt is made to slow down brute-force attack by implementing key stretching
    - Insufficient against dictionary attacks

Scope and Assumptions

- Scope
  - Replacement of Local Key Generation with JavaCard while retaining original UPM functionalities
  - Mutual Authentication for JavaCard and user
  - Generation and management of encryption key inside JavaCard
  - Ensure secure transmission of encryption credentials
  - Analysis of possible attack scenarios

- Assumptions
  - Initial setup of JavaCard for the user is done in a secure environment.
  - User PIN cannot be extracted from the JavaCard.
  - One JavaCard is handled by one user only

Proposed Approach

- Random encryption key generated on the JavaCard and stored on the card
- Provided to UPM application from card on user request through secure channel
- Key never leaves the card in clear
- Mutual authentication between application and JavaCard
- Protection against brute-force attack by limiting number of user attempts

Design

- Our solution uses the tuple:

  `<FileHandle, MasterPassword, EncryptionKey>`

- For creating new database, FileHandle and EncryptionKey are generated by the JavaCard one-time

- For opening existing database, the JavaCard compares the MasterPassword and FileHandle provided by the UPM application. If it matches, the 256 bit AES key is sent to the application by the JavaCard over Secure Channel

- The Applet allows the user 3 attempts to enter correct MasterPassword. This prevents against brute-force attacks.

Secure Channel

- Intent: Provide confidentiality, integrity and authentication of communication between UPM application and JavaCard in the presence of an active attacker.
- Approaches
  - Use of GlobalPlatform libraries
    - Integrated successfully with UPM application over JavaCard simulation environment
    - However, this was not supported by real card
  - Implementation of variant of symmetric key Needham-Schroeder protocol

Secure Channel Design

Initial Setup (Secure Environment assumed)

1. Long term Symmetric Key (LSK) Distribution

   UPM App $\xrightarrow{\text{Password+Salt}}$ JavaCard

   The distribution of symmetric key is handled by running the same algorithm on both sides with exchanged password and salt. The algorithm follows PBKDF2 and uses SHA-1 1000 times on MasterPassword and Salt for key generation. This step replaces the trusted server in the original Needham-Schroeder protocol.

Establishment of Secure Channel

2. UPM App generates Nonce-PC using Secure RNG, concatenates it with ID-PC and sends it to JavaCard after encryption with LSK.

   UPM App $\xrightarrow{E_{LSK}(\text{Nonce}_{PC}, \text{ID}_{PC})}$ JavaCard

Contd..

3. JavaCard decrypts incoming transmission and extracts the Nonce-PC.
4. Card generates Nonce-Card using Secure RNG, concatenates with ID-Card and Nonce-PC and sends it to PC after encryption with LSK..

   JavaCard $\xrightarrow{E_{LSK}(Nonce_{PC}, Nonce_{Card}, ID_{Card})}$ UPM App
5. UPM App decrypts incoming transmission and extracts the nonces and verifies the value of Nonce-PC. On success, the card is authenticated to the PC.
6. UPM App then encrypts Nonce-Card and sends it to JavaCard

   UPM App $\xrightarrow{E_{LSK}(Nonce_{Card})}$ JavaCard
7. JavaCard decrypts incoming transmission and extracts the nonce and verifies the value of Nonce-Card. On success, the PC is authenticated to the JavaCard. This ensures mutual authentication.
8. JavaCard then sends encrypted Acknowlegement of Success to UPM App for generation of session key (SK)

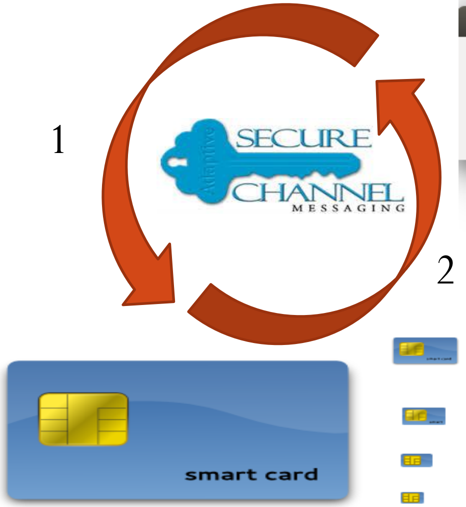
Contd..

9. SK = Hash (Nonce-PC | | Nonce-Card)

   This ensures participation of both parties in the protocol.
10. For creation of new database, UPM asks for FileHandle and Key from JavaCard
11. JavaCard applet generates the FileHandle and Secure Random Key, encrypts it using the session key SK and sends it to the UPM App.
12. UPM App uses this key for encryption and decryption of a single database.

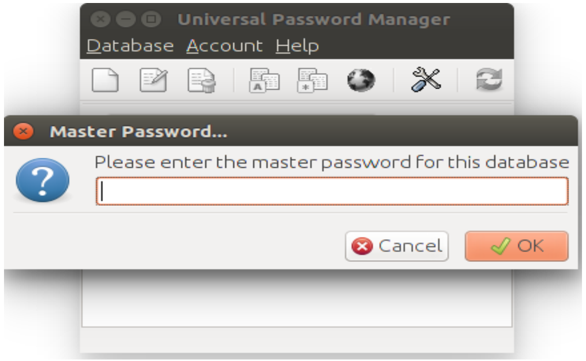
Implementation

MasterPassword+FileHandle



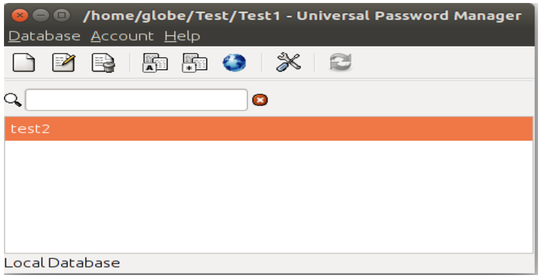
1

2



Please enter the master password for this database

3



test2

Local Database

Security Analysis

- The proposed solution is secure against dictionary and brute-force attacks because the database files are encrypted using cryptographically random key from the JavaCard
- The keys are stored and processed in the protected environment of JavaCard and transmitted to PC application through a secure channel once verification of user MasterPassword is confirmed.
- This gives us 2-factor authentication of 'something we know' and 'something we have'. A potential attacker needs to gain access to the card and also the MasterPassword .
- More than 3 incorrect attempts prevents brute force attacks on MasterPassword .
- Establishment of secure channel between the applet and UPM application protects the MasterPassword and Key against an active attacker.
- Implementation of Secure channel provides mutual auhentication between PC and JavaCard, preserves confidentiality of Key, protection against MITM attacks by participation of both parties for generation on SK.

Contribution

All project members actively participated in design, development and implementation of the project

1. Integration of UPM application with JavaCard simulation environment – Rajesh Kumar, Rajesh Mehta
2. Integration with Real Card - Rajesh Kumar, Rajesh Mehta
3. Design of Secure Channel – Ananya Chatterjee
4. Development and Integration of Secure Channel – Rajesh Mehta, Ananya Chatterjee
5. Final Integration with Real Card – Rajesh Kumar
6. Testing and Security Analysis – Ananya Chatterjee, Rajesh Kumar, Rajesh Mehta

References

[1] Adrian Smith, http://upm.sourceforge.net/

[2] Roger M. Needham, Michael D. Schroeder, Using encryption for authentication in large networks of computers, 1978

[3] Google Images