UNIVERSITY of WASHINGTON | BOTHELL

SCHOOL OF STEM

Computing & Software Systems

CSS 337 Secure Systems

Assignment 2

Due date: Monday 4 Mar

The Smart Wallet App is designed to run on a Trusted Platform. Each instance of the App will securely maintain the following parameters:

- Wallet ID (WID): A unique number identifying the Wallet instance (Set to last 3 digits of your Student ID).
- Balance: Initialized to 0.
- Bank Secret Key (K_{Bank}): An AES 256 bit key issued by the Bank. Used to transfer funds between wallets.
- Wallet Secret Key (K_{Wallet}): An AES 256 bit key (unique per wallet) generated as follows:

  K_{Wallet} = SHA-256(Student_ID)

  For example SHA-256(56789859) = B751237F9A7ED3530A67CEBD5E18816387C0A372A9E1D424A5F276A997FB6252
  Input is in ASCII, output is in hexadecimal

- A table to store list of Synchronized wallets.

The wallet should implement the following UIs:

UI1: Receiving funds from the bank

The bank issues an Electronic Money Draft (EMD) which is entered on the Wallet UI to deposit the funds. The EMD is simply the amount encrypted using the Wallet Secret Key (K_{Wallet}).

EMD = AES-256(Amount, K_{Wallet})

For example AES-256(00000000000000000000000000000123 ,
CF959C7BFC4FB5792AA25457578EF9E8B78E3558A8B7BF6A92338397B5F4639D) =
D8D9752ED5CCCB9F460FB8D6EB86A984
All numbers are in hexadecimal. Amount is $123.

The UI will prompt the user to enter the EMD, decrypt it using the Wallet Secret Key, and update the amount in the Wallet.

UI2: Synchronizing two wallets

Wallets will need to register with each other before they can transfer funds. The synchronization protocol is described as follows:

1. Wallet A will generate a token X_{AB} = AES-256(WID_A, WID_B, amount=0, counter=0) and display it on the screen (in Hex). Please refer to Token Structure in Appendix I.
2. Wallet B will collect X_{AB} from the user and add a record to the internal table used to store synchronized wallets. The record contains the WID_A and the incremented counter value which is (0+1).
3. Steps 1 -2 are repeated for Wallet B to Wallet A synchronization.

UI3: Sending Funds

The user of Wallet A will use this interface to transfer funds to Wallet B. The user will enter the amount and the receiving Wallet ID. The wallet balance will be updated accordingly.

A token will be generated X = AES256(WIDA, WIDB, Amount, C_B), where C_B is the counter value associated with WID_B in Wallet A's table. C_B is incremented by 1 afterwards.

UI4: Receiving Funds

The user of Wallet B will enter the value of token X on this UI. The token is decrypted and verified by making sure that WID_B is in the receiver's field and that the counter matches that in the record associated with WID_A. The wallet balance is updated accordingly and C_A is incremented by 1.

Appendix I: Token Structure

Token X is generated by encrypting a single block using AES256. The following table summarizes the structure of the block:

Bytes	Description
1 - 4	Sender's Wallet ID
5 - 8	Receiver's Wallet ID
9 - 12	Amount
13 - 16	Counter

The following key is used to encrypt the block and generate the token. Token is communicated to the user by displaying it on the screen.

K_{Bank} = F25D58A0E3E4436EC646B58B1C194C6B505AB1CB6B9DE66C894599222F07B893

Example: Wallet # 444 sending $21 to Wallet # 333 for the first time. The token is calculated as follows:

*AES-256(00000444**00000333**00000021**00000001** ,*
F25D58A0E3E4436EC646B58B1C194C6B505AB1CB6B9DE66C894599222F07B893) =
965390DFD8B18BCD419CA0583896218A
All numbers are in hexadecimal.