SoftwareDesignDocument (SDD)

Rev: A0-02

29March, 2017

“Bitcoin Vault”

From Concept to Customer

Revision History:

Date	Rev No.	Description	By
25-03-2017	A0-01	First Draft	VVDN
29-03-2017	A0-02	Second Draft	VVDN

Table of Contents

Table of Figures

1 Introduction

This document describes SW design details of "**Bitcoin Vault**" app (EDUK_SMOB).

This SDD is made for the reference of:

- Product managers at VVDN & Embedded Downloads to confirm the SW design & Architecture before implementation.
- Engineering Team at VVDN for implementation and validation of the EDUK_SMOB Project.
- System Integration and Verification team at VVDN/ Embedded Downloads.

1.1 Purpose of the document

Bitcoin Vault application will be used for secure Bitcoin transactions. Using this application, user can store bitcoins in wallet or vault

The key elements of this application are following:

- User will be able to transfer/receive Bitcoins from user's Vault to Wallet
- User will be able to transfer/receive Bitcoins from his wallet to another bitcoin address.
- User will be able to view transactions history along with the status of the respective transaction.
- User will be able to create new wallets. (To be discussed)

1.2 Scope of the Project

The Bitcoin Vault application is an application through which transfer/receiving of Bitcoins can take place. Users can access Wallets & Vaults for his account upon being successfully authenticated. User will be able to make transfers from Wallets to Vaults & vice versa.

2 Architecture Flow

The layered architecture of the application consists of three layers i.e. OS layer, SDK layer and App layer.

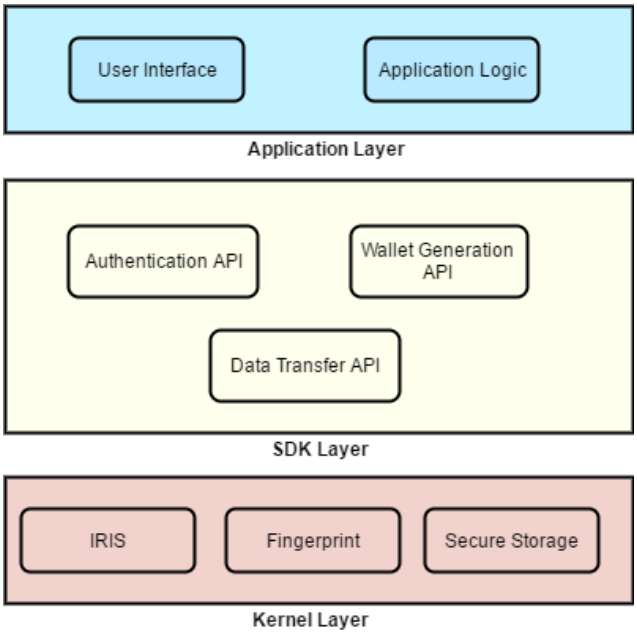

Figure 1: Layered Architecture Diagram

Requires the inclusion of the NFC card

Formatted: Normal, Left

2.1 Kernel Layer

This layer is first layer that is at the OS level. This provides access to Fingerprint/Iris and secure storage. Notifications needs to be added to inform the user that he has received Bitcoin even when the Bitcoin Vault application is closed.

2.2 SDK Layer

This layer will work as an interface between Application layer and Kernel layer. Using this layer, application can do following:

- SDK will expose an API that in turn calls kernel to generate keys and addresses.
- Authenticate user using the provided fingerprint/IRIS.

2.3 Application Layer

Application layer comprises of user interface and application logic

3 Application Flow and Design

This section provides you complete understanding of the application flow and design of the application.

3.1 Fingerprint Scan

This screen is used to authenticate user by scanning finger.

Figure 2: Fingerprint Authentication

Calling *AuthUser()* method will firstly display the UI where user put finger, fingerprint authentication will takes place by using SDK function *AuthUser(),* which in return will provide **Success/Failure**.

3.2 IRIS Scan

This screen is used to authenticate user by scanning iris.

Figure 3: IRIS Authentication

Can you please adjust the text “Put your eye on the circle” to “Please align your Bitvault until your eye is inside the circle”

Formatted: Normal, Left

When user puts eye, IRIS authentication will be the another input parameter to SDK function *AuthUserIris()*, which in return will provide **Success/Failure**.

3.3 Dashboard

Home Screen contains two options i.e. **Bitcoin-Wallet** and **Bitcoin-Vault**. Bitcoin-Wallet and Bitcoin-Vault will be created at the time of boot up. User can select any one of these by clicking on it.

Figure 4: Home Screen

3.4 Bitcoin Wallet

This screen is used to display all the wallets in the list and their current bitcoin balance.

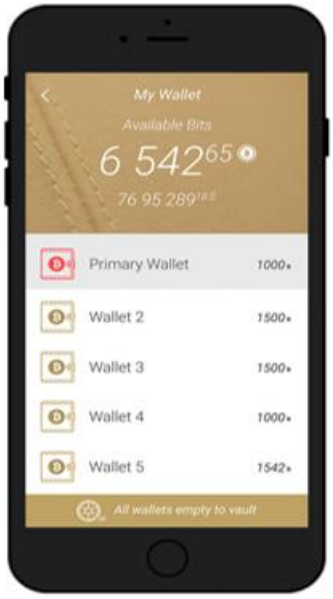

Figure 5: Wallets

There seems to be a problem with your calculation here:

1500bits = $76 96 289.18 Can you please fix this – It looks like you converted to rupees?

Should we rather have Bitcoin here in stead of Bits?

Formatted: Normal, Left

On this screen, following information is displayed

- **Available Bits**: This is sum total of all the bitcoins that are available in the wallets.
- **Amount in US Dollars**: This amount is equivalent of bitcoins in US dollars. This information is obtained using conversion rate function of SDK, which in turn will invoke block chain API.

***API - CONVERSION**:*

Block chain URL is used to get conversion value of dollar in bitcoin:

URL: http://<domainname>/BitConvertor?currency=USD&value=1

Type: GET

Return Type: JSON

Output: Available : value of BTC in 1 dollar

Input Parameters: currency,value.

Can you please clarify <domainname>? Where will this server be?

- **Available bitcoins in individual wallet :**This information is obtained using SDK function *getWalletBalance(String walletAddress*) in which we pass walletAddress as a parameter, which in turn invoke insight API.

API - SHOW BALANCE:

Insight.is URL is used to get balance for each address of a wallet:

URL: http://<domainname>/insight-api/addr /<WalletAddress>/balance

Type: GET

Return Type: JSON

Output: Available : amount in address

Input Parameters: Wallet address.

Can you please clarify <domainname>? Where will this server be?

- **All wallets empty to vault:** This will transfer all wallets bitcoins to vault.

3.5 Wallet Details

This screen is used to display the balance of the particular wallet with its transaction history.

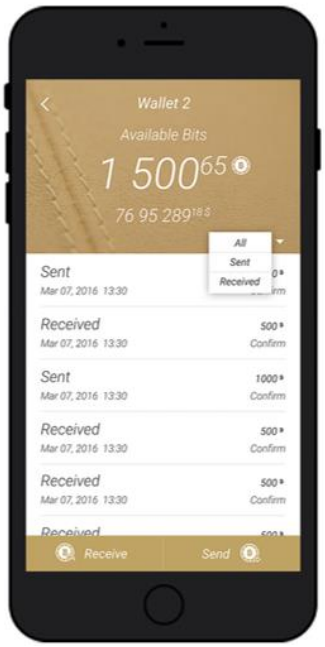

Figure 6: Wallet Details

There seems to be a problem with the calculation here – Can you please fix these screens. Did you convert to rupees instead of USD?

Will it be better to change Bits to Bitcoin?

Formatted: Normal, Left

On this screen, following information is displayed

- **Available Bits**: This information is obtained using SDK function *getWalletBalance(String walletAddress)* in which we pass **walletAddress** as a parameter, which in turn invoke insight API.

API - SHOW BALANCE

Insight.is URL is used to get balance for each address of a wallet:

URL: http://<domainname>/insight-api/addr /<WalletAddress>/balance

Type: GET

Return Type: JSON

Output: Available: amount in address

Input Parameters: Wallet address.

- **Transaction History**: This information is obtained by using SDK function, *GetTransactionHistory(wallet's address),* which in turn invoke insight API.

API - TRANSACTION HISTORY

Insight.is URL will be used for this feature implementation.

URL: http://<domainname>/insight-api/addr/<WalletAddress>

Type: GET

Return Type: JSON

Output: Date/Timestamp of transaction, Transaction ID, Transaction Amount

Input Parameters: Bitcoin address.

- **Filter Transaction**: This information will be displayed by filtering the transaction locally.
- **Receive**: This will opens a new screen on which wallet address of the wallet as well as QR code of the address is displayed.
- **Send**: This will opens a new screen which will provide a facility to enter or scan receiver address, amount to transfer and description for the transaction.

3.6 Send

This screen is used to send bitcoins from sender's wallet to receiver's wallet.

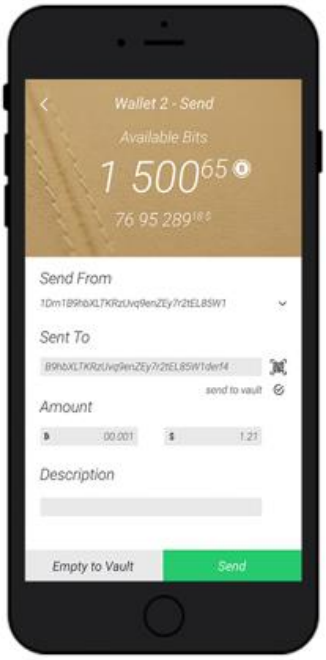

Figure 7: Send Bitcoins

On this screen, following information is displayed

- **Send From**: This will show the sender's current wallet address.
- **Sent To**: Sender either enter the wallet address of the receiver or scan the QR code of the receiver.
- **Amount**: Bitcoins to be transferred
- **Description:** It's an optional field, if sender wants to add to some description.
- **In the current SBIYP Application this field does not do anything –** It is not stored anywhere – Can we change this here to actually work?
- **Send :**This operation will follow following steps :
    - **Before the Send transaction can be completed, the user should scan his iris, fingerprint and NFC card.** Only once these three things have been successfully confirmed will the user's private key be generated - This is different to the way the current SBIYP application works!!
    - **Generating Hex**: This operation can be performed by using SDK function, *GenerateHex(unspentOutputs,Receiver's Address, Sender's restbitcoin's address,*

Formatted

sender's wallet address, amountToSend, miner fees, private key), which in turn provide hex value.

- **How is the private key sent across to the server**? Is this somehow encrypted?
- **Unspent Output**: This information will be obtained by using SDK function *GetUnspentCoins(wallet's address),* which in turn call insight API to provides unspentoutputs.
- **Once the send transaction is completed, the private keys should be deleted from the device and only be generated again once a send transaction is requested.**

Formatted: Font: Not Bold

Formatted: Font: Not Bold

API-Get Unspent Outputs

Insight.is URL will be used for this feature implementation.

URL: http://<domainname>/insight-api/addr/address/utxo

Type: GET

Return Type: JSON

Output: UnspentOutputs,

Input Parameters: address

- **Push Transaction:** This operation can be performed by using SDK function, *PushHexToBitcoinBlockchainServer(hex),* which in turn call insight API to get TransactionId

API - Push Transaction

Insight.is URL will be used for this feature implementation.

URL: http://<domainname>/insight-api/tx/send/

Type: POST

Return Type: JSON

Output: Transaction ID,

Input Parameters: raw transaction

As part of the process to transfer bitcoins between two users, a small token (1 bit) is also transferred to Embedded Downloads.

3.7 Confirmation

This screen is used to show message that transaction has been processed successfully.

Figure 8: Transaction Successful

The two buttons here are confusing – Make another transaction or Finish? I think the names of these buttons should be changed – Make another transaction should change to Make another Send Transaction – Finish Button should be called Home

Formatted: Normal, Left

On this screen, following options are displayed

- **Manage Another Transaction:** This option will navigate to the send screen to perform another transaction.
- **Finish:** This will navigate user on the home screen.

3.8 Receive

This screen is used to display the QR code and address of the particular wallet to receive the bitcoins.

Figure 9: Receive

Please ensure that the Address can be copied to text and pasted in other applications in the BitVault.

Formatted: Normal, Left

3.9 Bitcoin Vault

This screen is used to display the balance of the vault with its transaction history. To get the information same SDK functions and insight API will be used as with wallet.

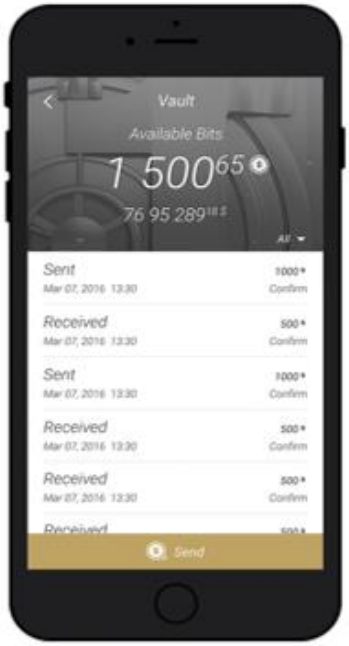

Figure 10: Vault Landing Screen

3.10 Send from Vault

This screen will be used to transfer bitcoins from vault to wallets. For sending bitcoins same SDK functions and insight api will be used, which are used to transfer bitcoins from one wallet to another wallet.

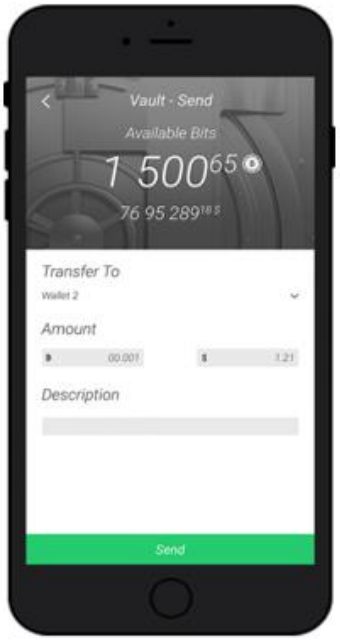

Figure 11: Vault Transaction

3.11 Empty to Vault

This screen will transfer all bitcoins from all wallets to vault.

Figure 12: Transfer from Vault to Wallet