# Payment Card Wallet Design

Bernard Parenteau

bern@floridalogic.com bparente@fit.edu

October 2018

v2 March 2019

## Payment Card Wallet

The Payment Card Wallet is a cryptocurrency software wallet for devices that can read information from a physical item such as a card, enabling the wallet to receive and hold cryptocurrency “coins” with multi-signature transactions. One of those signatures will have to be generated using information on the physical item or card. That item or card may be read by the software wallet by any means available to the device on which the wallet is running, such as reading an RFID chip.

A multi-signature transaction requires n of m signatures, where n and m can each be any relatively small number (given that n<=m). So coins can be received in such a way that more than one signature is required to spend them.

Many existing cryptocurrency wallets support these multi-sig transactions and they can be used for additional security such as requiring two devices to spend, or for multi-party accounts. With the Payment Card Wallet, all the receiving addresses that it generates will represent multi-sig transactions.

A user of the wallet will receive a physical item such as a credit card-sized smart card (the user would probably want multiple copies of the card). The item or card will contain cryptographic codes that can be read by the wallet (e.g. via RF).

To generate an address to receive funds, the wallet will use a multi-signature transaction using both a key held by the wallet and a key held (partially or fully) by the item or card.

To spend funds received at such a generated address will require at least two signatures. One signature will be done in the usual way using the software wallet’s own keys. For an additional signature, the wallet will read from the item or card to obtain the key information necessary to generate that additional signature.

So the payment card must be in physical proximity to the device on which the wallet is running (e.g. a mobile device) for any coins to be spent.

Unlike some other security schemes, the Payment Card Wallet is more resilient to attacks in which phone service and phone numbers are hijacked, because a physical card is required to spend the coins.

## Payment Card Cryptographic Details

(The CardWalletDiagram document depicts the data flow)

The card contains 2 random bit strings

* a source value (S)
* a randomizer value (R)

Upon setup, the wallet reads the values.

The user is asked to generate some randomness (e.g. by moving their mouse around or using captcha clicks) and that randomness is digitized (D).

The randomizer value R is appended to D, and the result is hashed via SHA512 producing H.

(note; R is used to increase the chances that the source of H is actually random, even if the user-generated randomness is not particularly random)

The hash result H is saved locally and a passphrase is generated from it for the user to save in case the wallet needs to be rebuilt.

From the SHA512 result H;

-the left 32 bytes is the hash key HK

-the right 32 bytes are for starting key position (SKP)

The card’s source value S is hashed via HMAC-SHA512 using HK, producing the master private extended key (XK).

The left 32 bytes of XK is the master private key, while the right 32 bytes is the chaincode.

S is not saved by the wallet and must be handled in such a way that it is not inferable by forensically examining the wallet’s code execution.

The hash key (HK) is saved locally by the wallet.

The result is that the master private key is random and the card alone cannot be used to generate the master private key, nor can the wallet alone generate the master private key.

To sign future transactions, the source value S is read from the card and hashed as above to get the master private key.

To generate keys for use by the wallet, use the standard HD hierarchy as described:

<https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>

<https://medium.com/bitcraft/hd-wallets-explained-from-high-level-to-nuts-and-bolts-9a41545f5b0>

<https://github.com/satoshilabs/slips/blob/master/slip-0044.md>

From the extended master private key, child private keys and child public keys can be generated.

Receiving addresses will be generated from multisig transactions (within a script transaction) where one of the public keys is a child public key generated from the card as above, and the other is from the software wallet’s HD keys. Rather than starting at 0 for the child private key sequence, the wallet will start at SKP, and the HD key from the software wallet in any transaction will always be the public key from the matching key position. This will make the key pairs (and script hashes) predictable, thereby enabling recovery.

The result is that neither the wallet alone nor the card alone could be used to generate the necessary signatures to sign transactions received at wallet addresses.

When cards are created, there should be multiple copies produced to provide backup in case a card is lost or damaged.

## Wallet Information, Backup. Restore

The wallet will keep an internal persistent file of the Payment Card public keys received and associated information related to transactions hashes used and the information related to each script hash. Some of that information (the Payment Card public key) will enable the wallet to build the multi-sig script and the P2SH script in the spending transaction. Additional information would be the status (unused, pending, holding funds, used-spent) and associated transaction info. This data can be externally generated for backup.

If the wallet or its internal persistent storage is deleted, damaged, or unreadable, and backup is unavailable, it would have to be rebuilt. The wallet could be reinstalled if necessary and in any case would need to go through a procedure similar to new setup, starting after the point that the hash result H was saved and generated a key phrase. The rebuild process would instead ask the user for the key phrase to generate the hash result H. The setup process would continue from there in the same way as the initial setup.

The last step would be a procedure that would search the UTXO for any matching script hashes. The wallet will generate the script hash addresses for multisig transaction scripts with public keys from both the wallet and the card, with matching sequence numbers, starting at the SKP (starting key position). Subsequent transactions would start using a key position an arbitrary number (e.g. 10) greater than that used by the last discovered transaction.