Design and Analysis of Wallets for Selected Cryptocurrencies.

Ankush Arun Kawanpure  
Department of Computer Science  
Golisano College of Computing and Information Sciences  
Rochester Institute of Technology  
Rochester, NY 14586  
aak3988@rit.edu

Abstract—Cryptocurrency wallets are the software which bridges the gap between user and cryptocurrencies; they store the necessary information to complete transactions. Cryptocurrency wallets have three goals to achieve: availability, security, and convenience. While designing a wallet, it is challenging to incorporate all the three goals. Simple device wallets prioritize convenience over availability and security, whereas hierarchical deterministic wallets prioritize security and availability over convenience. Also, storing cryptocurrency on a device connected to the network is a security risk. Hierarchical deterministic wallets use a known seed for parent and child key generation and use elliptic curve cryptography and mathematics to calculate the public addresses without revealing the private keys.

Hierarchical deterministic wallets help to decrease the risk of online storage and improve the security by just publishing the public keys and keeping private keys offline. Hierarchical deterministic wallets are becoming a standard for storing cryptocurrency, but the complex configuration and cryptography involved makes them often inconvenient. This work aims at making hierarchical deterministic wallets more convenient without compromising security.

I. INTRODUCTION

In 2008, Satoshi Nakamoto [1] introduced peer-to-peer digital currency which uses cryptography to secure the transactions enables users to send payments across the world without any centralized authority. Since then many new cryptocurrencies come into existence, Bitcoin, Litecoin, Ethereum are some of the few with very high transactional volume 1. Low fees, faster transfer rate and ability to make virtually anonymous payments makes them popular. Manually performing cryptocurrency transaction and managing private keys very difficult. Cryptocurrency wallets provide an easy way to perform transactions and securely store the private keys. However, Wallets have limitations over improving security, and convenience.

Since 2008, many types of cryptocurrency wallets have been developed. Different kind of wallet served a different purpose and prioritized different design goals. Based on the key generation process wallets can be classified into following types.

1) Paper wallet: Simple way of storing cryptocurrency is to write the private keys on the paper and store it safely. Although such wallets provide excellent security, such wallets are very inconvenient to use. Also, any physical damage or theft may cause loss of keys and so as the cryptocurrency. Bitcoin paper wallet 2 provide a tool to generate private and public keys off-line.

2) Random independent wallet: These type of wallet uses a pseudo-random number generator to generate private keys. These wallets are very convenient as they can create private and public keys on the go. However, these wallets require a regular backup because device damage or theft may cause loss of keys. Bitcoin core 3 is the best known Random independent wallet.

3) Deterministic wallet: Deterministic wallet uses a random seed to generate private keys. Private key generation process can be explained as follows

\[ \text{privkey}_i = \text{Hash}(\text{seed}||i) \]  

The user only requires to backup the seed value, private keys and public keys are generated in the wallet deterministically. However, deterministic wallets stores public and private keys on the same device and it is a security risk for storing a large amount of currency.

II. PROBLEMS WITH WALLETS

Secure storing of cryptocurrency means keeping private keys safe.

A. Security issue

Storing private and public keys on the same device create the security risk. Also, many wallets store private keys in the unencrypted format, an attacker may try to get access to such device to get all of the private keys. In 2014 Mt. Gox’s unrealized theft of private keys resulted in Mt. Gox bankruptcy4.

B. Availability issue

The most secure way to store cryptocurrency is to store private keys off-line. But this makes cryptocurrency harder to access for a daily transaction. Also, damage or theft of such device causes loss of cryptocurrency.

1Transaction charts: https://bitinfocharts.com/  
2Bitcoin paper wallet Available: https://bitcoinpaperwallet.com  
4Mt. Gox bankruptcy timeline: https://medium.com/@jimmysong/mt-gox-hack-technical-explanation-37ea5549f715
C. Convenience issue

Securely encrypted wallets require proper back-ups to restore the wallet in case of software or hardware failure. Configuring such wallets may require expertise and incorrectly configured wallets may cause loss of coins. In 2011 someone lost 7208 Bitcoins while restoring wallet backup.5

III. COLD STORAGE WALLET

To solve the problem of storing public and private keys together, we can generate them in a batch and store them separately, private keys off-line device and public keys on the networked device. But this creates a problem of periodic transfer public and private keys between these devices.

Hierarchical deterministic wallet solves the problem of periodic transfer; it can generate public and private keys on separate devices independent of each other. Fig. 1 shows the basic block diagram of hierarchical deterministic wallet.

IV. HIERARCHICAL DETERMINISTIC WALLETS

Hierarchical deterministic wallets take advantage of homomorphism feature of the elliptic curve discrete-log based cryptosystem (ECDSA).

A. Elliptic Curve Digital Signature Algorithm

Bitcoin, Ethereum and all the cryptocurrency derived them uses the Koblitz curve secp256k1 to sign the transaction [1] [2]. Parameters specified for the Koblitz curve secp256k1 [3] are as follows:

\[F_p:\text{ finite field defined by}\]
\[p = FFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFF FFC2F\]
\[= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1\]

\[E:\text{ Curve over } F_p \text{ defined by}\]
\[y^2 = x^3 + ax + b \text{ where}\]
\[a = 0\]
\[b = 7\]

\[G:\text{ Base point in compressed form}\]
\[G = 0279BE667E F9DCBBAC 55A06295 CE870B07 92FCDDBC 2DCE28D9 59F2815B 16F81798\]

\[n:\text{ Order(G)}\]
\[n = FFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFBAAEDCE6 AF48A03B BFD25E8C D0364141\]

\[h:\text{ cofactor}\]
\[h = 01\]

The private key is a random large integer number and public is a point on the elliptic curve.

\[P(x, y) = k \cdot G(x, y)\]

where
\[k : \text{ Private key or random large integer}\]
\[p : \text{ Public key or point on the elliptic curve}\]
\[G : \text{ Base point on the elliptic curve}\]

According to homomorphism if we create a new private key by adding two private keys over a finite field, the same is true for corresponding public keys.

\[k_{new} = k_1 + k_2\]
\[P_{new} = k_{new} \cdot G\]

In equation 2 addition is arithmetic addition

\[P_{new} = (k_1 \cdot G) + (k_2 \cdot G)\]
\[P_{new} = P_1 + P_2\]

In equation 3 addition is elliptic curve point addition

By using this feature, we can separate the private keys generation from public keys generation. Private keys can be generated off-line and stored securely, in contrast corresponding public keys can be generated and stored on the networked device for monitoring the output transaction on the public keys.

B. Key Generation

In hierarchical deterministic wallets, we generate all the keys deterministically from a single seed value. Bitcoin Improvement Proposal 39 [4] provide an excellent way to generate a random phrase, this generated phrase combine with a user-provided secret used as a seed value.

1) Master Key Generation: Master key generation is the first step of the key generation process. The seed value is passed through the HMAC-SHA512 hash algorithm to give an output of 512bits. These 512 output bits are then separate into master private key and master chain code, also master public key is derived from master private key. From these three master keys, we generate the extended private Key which is a combination of master private key and master chain code. Also, we generate the extended public key which is a combination of master public key and master chain code as shown in Fig 2.

2) Extended Keys: Extended keys mainly consist of master public-private keys and master chain code which is useful for the child key generation. Extended keys also contain metadata information useful for creation of network specific compressed public addresses. The format of the extended key is described in fig 3 Version represents the network prefix for calculating the keys. Depth is the hierarchical level at which we are generating keys. Parent fingerprint the first 4 bytes of parent public key. Key index is the user account number. Chain code, public Key or private key are the parent keys.

5Bitcoin lost in backup mix-up https://bitcointalk.org/index.php?topic=11104.0
3) Child Private Key Generation: Fig 4 shows the child private key generation process. First parent private key converted into parent public key. Parent public key along with parent chain code and key index passes through the HMAC-SHA512 hash function to get 512 bit output. Out of 512 bit left 256 bits arithmetically added with parent private key over finite field G and used as child private key. Also right 256 bits used as child chain code.

4) Child Public Key Generation: Fig 5 shows the child public key generation process. Parent public key along with parent chain code and key index passed through the HMAC-SHA512 hash function to get 512 bit output. Left 256 bits from the output added with parent public key on the elliptic curve gives us child public key. Also right 256 bits used as child chain code.

5) Hardened Key Derivation: Child private key is derived from child public key and parent chain code as shown in fig. 4. However, if an attacker gets access to the child private key and parent chain code, attacker can solve the equation 4 for x to get the parent private key.

\[ x : x + left256_i = privkey_{i+1} mod(G) \]  (4)

Hardened keys are the child keys derived from parent private key instead of parent public key. Hardened keys provide a fixed hierarchy in the wallet and use to break the link between parent public key and child chain code.

C. Architecture & Design

Hierarchical deterministic wallet consists of two main components. Cold storage or Off-line storage to store the private key and sign the unsigned transaction and Networked wallet used to store public keys on the networked device.

1) Block Diagram: Fig. 7 represents the block diagram of the wallet. Complete wallet split into two hierarchical deterministic wallets, one wallet is on the network side to perform small transactions and one split between network and off-line side.

2) Wallet Architecture: Fig. 9 represents the complete architecture of the hierarchical deterministic wallet. The off-line wallet is responsible for the generation of extended private and public keys. Extended private keys are stored on the off-line, whereas Extended public keys are transferred on the networked side. This transfer can take place via QR-code or by manual entry. Along with extended public key, a hardened key is also moved to the network side with QR-code.

Networked wallet derives public keys and monitors for the outputs on the public keys. The hardened key is used to create
a separate wallet on the networked side capable of performing transactions on its own. This complete wallet helps to improve the convenience while using the wallet for small transactions.

Ideally, we must store a large sum of coins on the cold storage away from the network and a small fraction of coins on the network side wallet to perform small transactions. This architecture helps to reduce the security risk and improve the convenience of the wallet.

3) **API calls:** In this wallet, we are not storing blockchain locally, so to provide complete functionality, we are calling a REST API serving the full node. Fig. 8 give a detailed about the required api calls.

We are using Blockchain.info full node API in this project. The most important API call includes get unspent transactions and get transaction.

V. IMPLEMENTATION & RESULT

For the wallet implementation following libraries and software are used:

A. **Libraries and API**

1) **BitcoinJS library:** Actively maintained Javascript library\(^7\) for node.js and browser-based applications. Provide an easy way to interact with bitcoin protocol. This library handles essential elliptic curve operations like generating a public address from the private key. This library also provides abstraction while creating a signed transaction

---

\(^6\)Blockchain.info provides a robust api to query blockchain https://blockchain.info/api/blockchain_api

\(^7\)Github repository: https://github.com/bitcoinjs/bitcoinjs-lib
2) **Blockchain.info API**: Size of Bitcoin blockchain is more than 165GB\(^8\), and it is increasing. Blockchain.info provides a robust API for querying blockchain without storing it locally; this enables us to improve the storage requirement for our networked wallet. API calls include getting a list of unspent transactions, also getting balances, validating and broadcasting a signed transaction. Fig. 8 provides details about the API calls used in the implementation.

3) **Bitcoin Testnet 3**: Bitcoin testnet\(^9\) is an alternative bitcoin blockchain use for testing purposes. Bitcoin Testnet blockchain maintained by the developers and coins are worthless so as useful for testing. Bitcoin testnet has all the latest features present in the actual Bitcoin blockchain. Blockchain.info also provides API to access testnet blockchain.

\(^8\)Blockchain size according to blockchain.info https://blockchain.info/charts/blocks-size

\(^9\)Bitcoin testnet: https://en.bitcoin.it/wiki/Testnet

**B. Implementation**

Fig. 10 shows the implemented off-line wallet. This wallet is capable of generating a public and private key pair as well as extended public and private keys.

Fig. 11 shows the implementation of networked wallet. This wallet takes extended public or private key as input. If an extended public key is used, we are just capable of monitoring for outputs on the public keys and creating an unsigned transaction. To sign the transaction, we need to use the offline wallet. If we use an extended private key we can monitor as well can create signed transactions.

**C. Result**

Performing transaction using this wallet is the primary task. Fig. 12 shows the transaction JSON format. Transaction includes input transactions, output transactions, transaction hash and some other metadata information.
Fig. 13 shows the accepted transaction on the blockchain. Blockchain.info transaction explorer shows the block confirmation, acceptance time and transaction details.

VI. CONCLUSION

As more and more people are using cryptocurrency, there is a need for secure storage. Many commercial businesses are implementing Hierarchical deterministic wallets and offering them as a service. But the problem remains whether someone should trust such service. By implementing this wallet I learned challenges to improve the security as well as convenience. There is always a tradeoff between security and convenience, and it is important to find the right balance between the two.

VII. FUTURE WORK

In this project, we aimed to improve convenience as well as the security of cryptocurrency wallet. Future work for this project includes further enhancing the security of the wallet. Security improvements involve implementing Bitcoin Improvement Proposal 39[4] to generate mnemonics phrase which is a part of seed generation process. Shamir Secret Sharing Scheme[5] can be used to split the secret between two or more individuals, and M out of N confirmation needed to authenticate, which increase the security against theft. A multisignature transaction requires M out of N signatures to authorize a transaction. Feature to create multisignature transaction will be an improvement in the security of the
Fig. 10. Off-line Wallet

Fig. 11. Networked Wallet
wallet.
Also, developing a mobile client application for network wallet will help to improve the convenience of the wallet.

ACKNOWLEDGMENT

I would like to take this opportunity to thank my advisor Prof. Stanislaw P. Radziszowski. His support, guidance, and motivation plays an important role in completion of this project. I am thankful for getting an opportunity to present my work in front of SIRA group. I would also like to thank Prof. Joe Geigel for giving me helpful feedback on my project milestones presentations.

REFERENCES

Fig. 13. Confirmed Transaction on Blockchain