CSE5095-010: Blockchain Technology

Lecture 11

Ghada Almashaqbeh
UConn - Fall 2020

Outline

- Key management in cryptocurrencies.
  - Hot and cold storage.
  - Hierarchical wallets.
  - Cold info storage.
  - Splitting and sharing keys.
  - Online wallets.

Spending Coins

- Recall that coins in any cryptocurrency are virtual.
  - Strings of bits.
  - One owns all unspent coins, which are transaction outputs that direct coins to their address(s).
- Spending any amount of coins requires:
  - Some public information from the blockchain.
  - The secret key associated with the address (or public key) that owns the coins.
    - Needed to provide digital signatures.
- Losing the secret keys means losing these coins; no one will be able to spend them.

Key Management

- Storage and retrieval of secret keys.
  - Involves also when and how to generate new keys for future transactions.
- Goals:
  - Security; only the legitimate owner get to spend a given coin.
  - Availability; coins owners can spend them whenever they wish.
  - Usability; it is relatively easy for the average user to store/retrieve/use secret keys.
- We will focus on Bitcoin in the slides.
  - The introduced concepts can be applied to any other cryptocurrency.

Wallet Software

- User (or client) software that keeps track of coins that a client owns.
- Provides a convenient user interface to simplify operations.
  - Issuing transactions.
  - Tracking total balance.
  - Generate keys when needed.
  - Bookkeeping of these keys.
- Encode addresses as text strings (Base58) or in the form of QR codes.
  - Simplifies sharing addresses with others.
- A large number of wallets is available.
  - Different flavors; desktop, mobile, web applications, etc.
  - Different vendors; metamask, jaxx, coinbase, etc.
  - Security is a driving factor of which one to choose.

Naive Solution

- Store the keys in a file on your laptop or smartphone.
  - This is just like carrying all life savings in your wallet; if you lose the wallet you lose all your money.
  - Security is tied to your device; breaking into the device allows an attacker to steal your keys.
- This is called hot storage.
  - Easy to use but risky.

Hot vs. Cold Storage I

- Hot storage.
  - Storing secret keys on a device that is connected to the Internet and used frequently for variety of applications.
  - Usable or coventient.
- Cold storage.
  - Offline storage, like on a machine or memory device that is stored in some safe location.
  - Less convenient.
- The majority of the coins are in the cold storage with a few in the hot storage.

Hot vs. Cold Storage II

- Seperate keys are needed for each.
  - Otherwise, compromising hot storage will compromise cold storage.
- But both should be aware of their addresses to allow transferring currency.
  - Cold storage stores its cold secret keys, cold addresses, and hot public addresses.
  - Hot storage stores hot secret key and public addresses, as well as cold public addresses.

New Addresses of Cold Storage?

- A good practice in Bitcoin to break transaction linkability is to create a new address for each new transactions.
- How can a hot wallet learn the addresses of a cold wallet?
  - Remember cold wallet is offline, no internet connectivity.
- Even generating a large chunk of addresses at the beginning will not work.
  - Cold storage needs to connect whenever a new batch of addresses is generated.

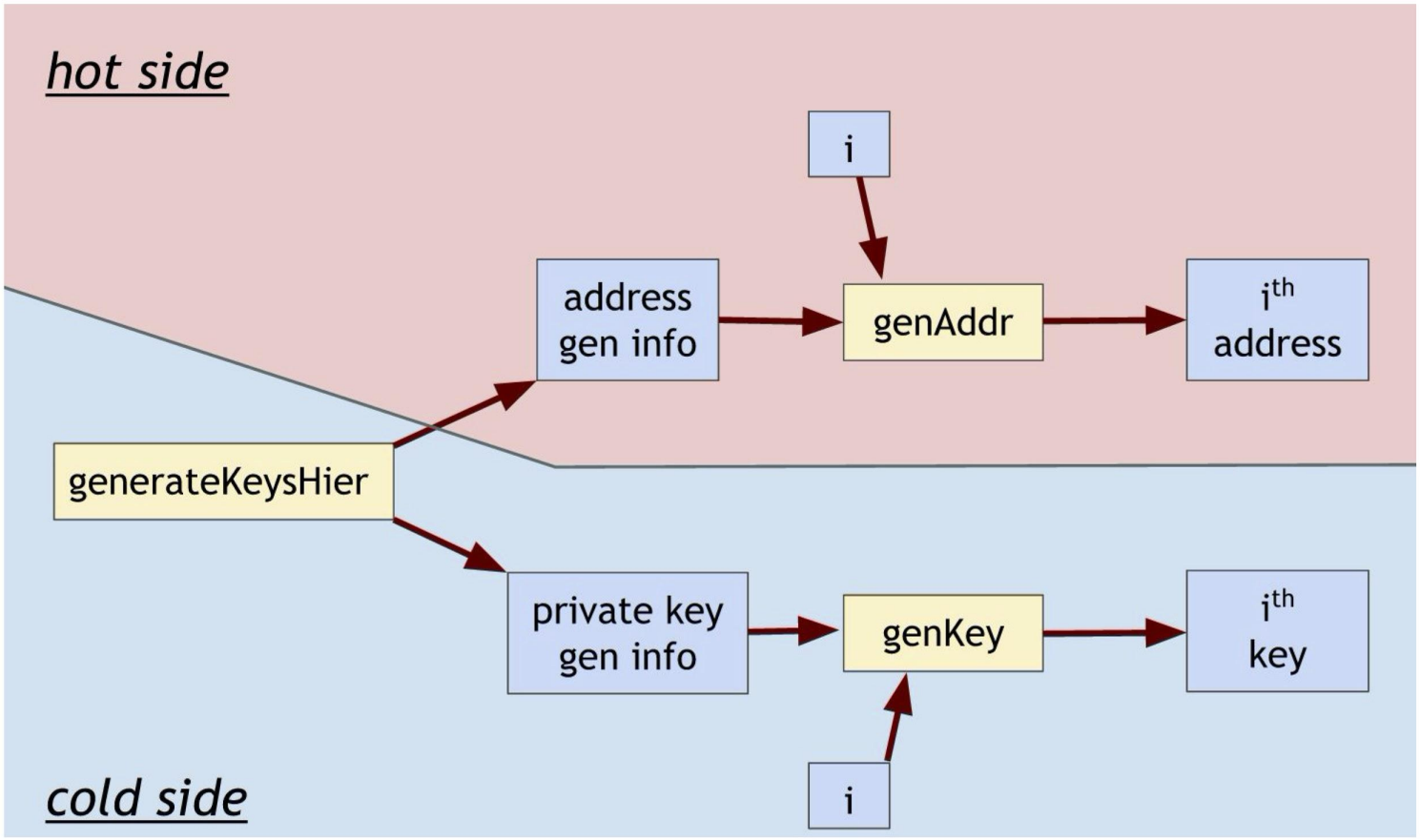
Hierarchical Key Generation I

- Allows cold storage to generate an infinite number of key pairs (or addresses).
- Usual keyGen algorithms generate a single key pair; public and private.
- Hierarchical keyGen instead generates some info that allows generating all future keys in a deterministic way.
  - Usually called master public key and master secret key.
- The master public key does not reveal any info about the master secret key or all future generated secret keys.

Hierarchical Key Generation II

- By having the operation indexed with some integer i, the master key allows generating the ith future key.
  - Works for both public and secret keys.
- Hot storage has the a copy of the master public key, while the master secret key is known only to the cold storage.
- Not all digital signature schemes support this hierarchical approach.
  - ECDSA supports that, details come later.

Hierarchical KeyGen - Pictorially



*From Ch 4, Bitcoin and Cryptocurrency Technologies book.

Hierarchical KeyGen in ECDSA (BIP32)

- A group G, in which DDH is believed to be hard, of order p (where p is some primer) and a group generator g.
- KeyGen is extended into 3 algorithms:
  - keyGenHer(1^n): msk = x, mpk = g^x, Hash function H.
    - n is the security parameter.
    - x is some integer selected at random from Z_p.
  - addrGen(mpk, i): r = H(i || mpk), pk_i = mpk*g^r = g^{x+r}, $addr_i$= H(pk_i)
  - keyGen(msk, i): r = H(i || mpk), sk_i = msk + r = x+r
- A hot wallet will be able to generate the ith address ad a cold storage will be able to generate the ith private key.

Security Issue I

- No forward or backward security.
  - Given i, ski, mpk, it is easy to determine msk.
    - Assume an attacker compromised the hot wallet (got hold of mpk), and one secret key of the cold storage has been leaked.
    - This allows the attacker to compute msk, and hence, all previous as well as future secret keys.
    - ***Exercise:*** track the algorithms and see how to msk can be computed.
- Source of vulnerability; the way randomness r is computed.

Security Issue II

- BIP32 gives an alternative construction that preserves forward and backward security if a private key is leaked.
  - However, no hierarchical addresses anymore, single address but hierarchical secret keys.
  - Transaction linkability!!
- Guteso et al. [Guteso et al., 2015] developed a bitcoin hierarchical wallet that tolerate leakage of m keys.
  - Drawbacks:
    - m must be fixed in advance.
    - Size of the mpk grows with m.

References

- [Guteso et al., 2015] Gutoski, Gus, and Douglas Stebila. "Hierarchical deterministic bitcoin wallets that tolerate key leakage." In International Conference on Financial Cryptography and Data Security, pp. 497-504. Springer, Berlin, Heidelberg, 2015.