Bitcoin key management

Anish Mathuria, DAIICT

Workshop on Blockchain Technology and Applications

Outline

- Basic concepts
- Wallets
- Deterministic key generation
- Multisignatures
- Two vulnerabilities (existing)

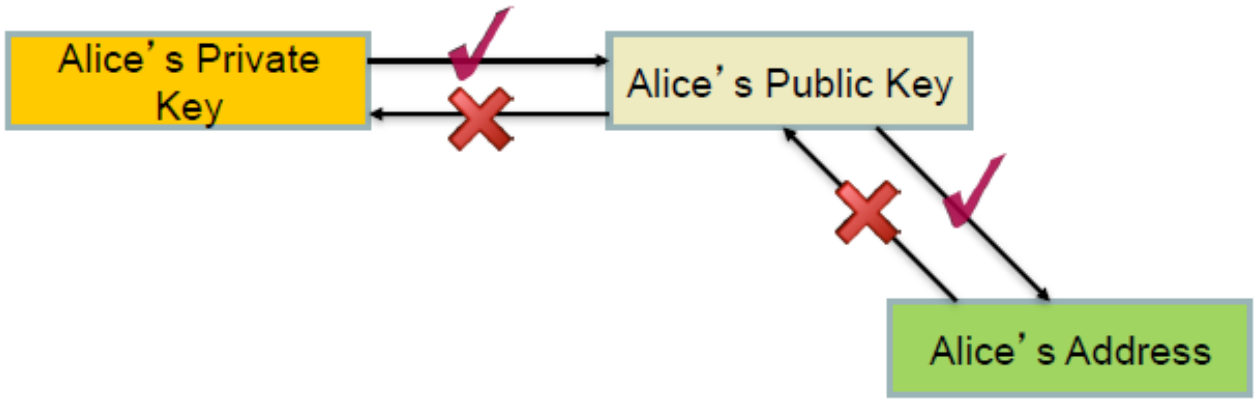
Bitcoin Public/Private Keys

- Bitcoin uses an instantiation of Digital Signature Algorithm (DSA) on elliptic curves
- A private key is 32 random bytes
- A public key is computed from a private key
- There is no encryption in Bitcoin, only signing

Bitcoin Addresses

- A Bitcoin addresses is a bit like a bank account
  `1Kk18SN6WRPTEXbXBm3dZSzEw7NdbChyc9`
- Calculated from a public key
  `RIPEMD-160( Sha256( public key ) )`
- Nobody knows who owns which addresses
- Value is moved between addresses using transactions.

Properties



Alice generates the private key
Only Alice can generate the public key and address

Spending bitcoins

- Suppose I want to pay you 1 BTC
- I generate a "transaction" record and sign it
  - Contains the amount, some metainfo, and your address
    - Also has hash of previous transaction that granted me the money I'm using
  - Signed by my secret key
    - If I lose the secret key associated to the transaction that granted me the bitcoins I'm sending you, I lose that money!

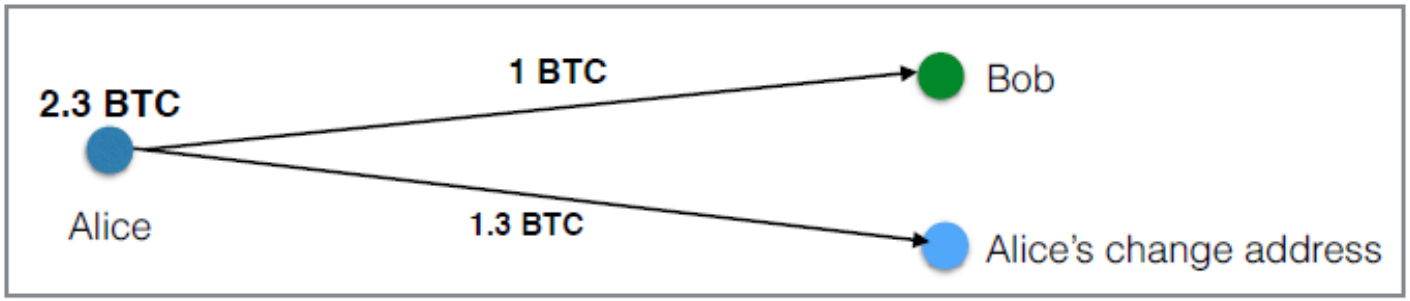
Transactions

- A normal transaction consists of one or more inputs and one or more outputs
- Each input has a value (number of bitcoins) and each output has a value

$$\Sigma \text{inputs} \geq \Sigma \text{outputs}$$

  Difference is the transaction fee
- Each input must spend all the value from some specified unspent previous input, known as an unspent transaction output, or UTXO

Change addresses



- Each transaction must reference a previous transaction, so all bitcoins received must be spent all at once
- Change address: used to collect excess bitcoins

Anonymity basics

- Literally: anonymous = "without a name"
- Recall: Bitcoin addresses are public key hashes rather than real identities
- Computer scientists call this pseudonymity

Anonymity basics

- Anonymity = Pseudonymity + Unlinkability
- Unlinkability
  - Different interactions of the same user with the system should not be linkable to each other (and to the user's real identity)

Anonymity basics

- Anonymity = Pseudonymity + Unlinkability
- Unlinkability
  - Different interactions of the same user with the system should not be linkable to each other (and to the user's real identity)
- Is Bitcoin anonymous?

Wallet

- Wallet = file which stores your “coins”
- A Bitcoin client app is also called a wallet
- Main Functionality:
  - Private Key Generation
  - Public key Generation
  - ECDSA Sign

Major wallet types

- Desktop client: You participate as node in the network
  - Private key on your machine (lose it, lose your coins)
- Online Service: You give your private key to a company/site
  - Log in to site to view "balance", make transactions; easy to use
  - They have your key
- Offline (cold) storage
  - Address and private key stored in offline media, such as in USB keys or printed out on paper

Paper wallets

Print bitcoin address / private key on paper

Bitcoin Address

SHARE

1Atuv5zFi5P5dzgfHNGWWR8EWjRSzDbCEL

Private Key

SECRET

L13HRyX7Lj3TLve4jAx53ink49sR6eLrJP2q5kvijPQDzGBzVARG

Evaluation of key management techniques[1]

- Malware Resistant
- Key Stored Offline
- No Trusted Third Party
- Resistant to Physical Theft
- Resistant to Physical Observation
- Resilient to Password Loss
- Resilient to Key Churn
- Immediate Access
- No New User Software
- Cross-Device Portability

Scoring: Full (•) - Half (◦) and empty for none

[1] Eskandari et al., USEC'15

Key management tradeoffs[2]

Category	*Example*	Malware Resistant	Key(s) Kept Offline	No Trusted Third Party	Resistant to Physical Theft	Resistant to Physical Observation	Resilient to Password Loss	Resilient to Key Churn	Immediate Access to Funds	No New User Software	Cross-device Portability
Keys in Local Storage	Bitcoin Core			●		●	●	●	●		
Password-protected Wallets	MultiBit		○	●	○	●		●	●		
Offline Storage	Bitaddress	○	●	●			●				●
Air-gapped Storage	Armory	○	●	●		●	●	●			
Password-derived Keys	Brainwallet		●	●	○			●	●	●	●
Hosted Wallet (Hot)	`Coinbase.com`						●	●	●	●	●
Hosted Wallet (Cold)		○	●				●	●		●	●
Hosted Wallet (Hybrid)	Blockchain.info		○	○			●	●	●	●	●
Cash		●	●	●		●	●	●	●	●	●
Online Banking							●	●	●	●	●

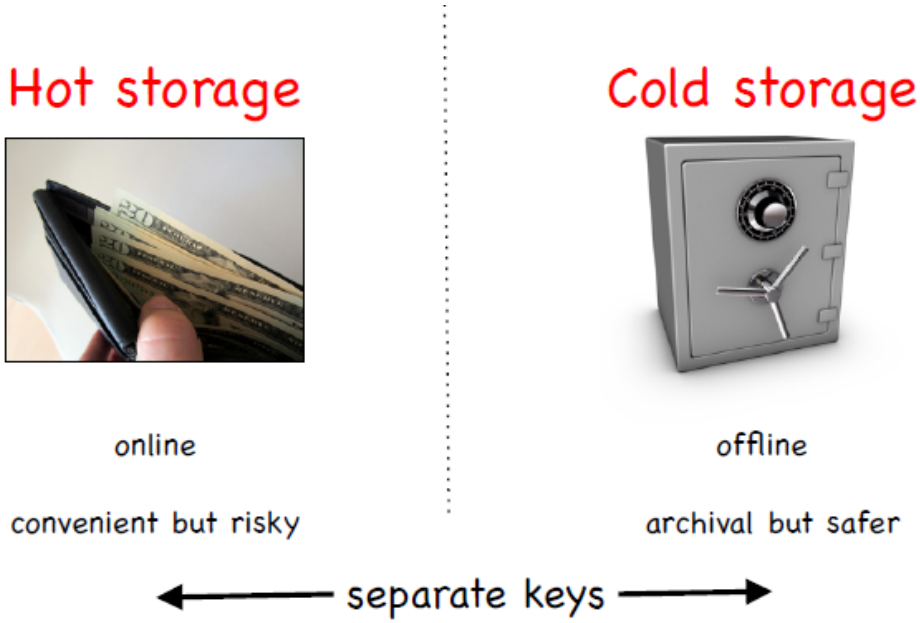
- No single superior approach
- Hosted wallets are the most similar to online banking
- All techniques have potential usability pitfalls

[2] Eskandari et al., USEC'15

Usability pitfall

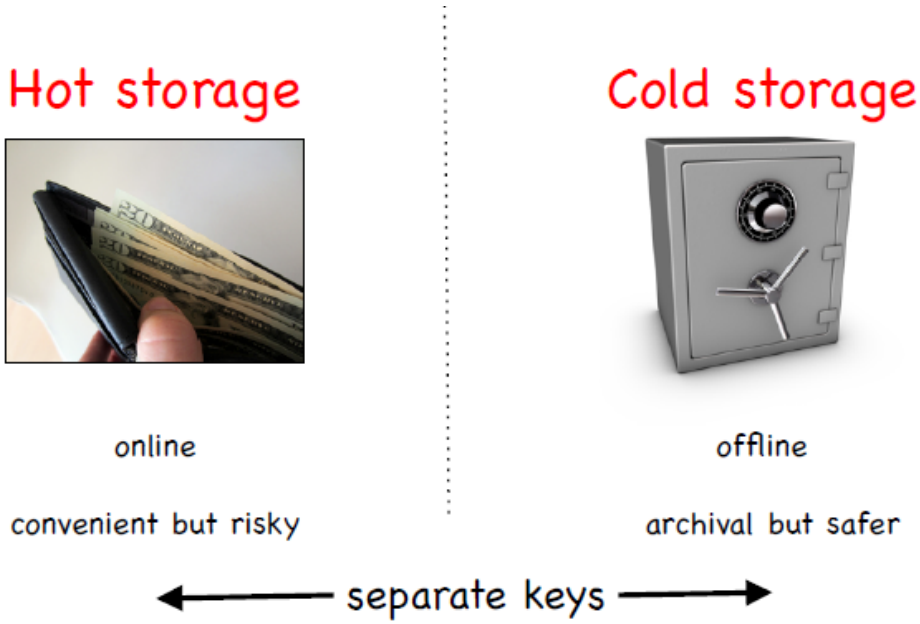
- User spends funds from address stored on a paper wallet by importing the private key into a wallet client.
- Client returns change to a newly generated address, as the user does not spend the full amount on the paper wallet.
- Subsequently, importing the paper wallet into a different client will display no funds.

Hot and Cold Storage



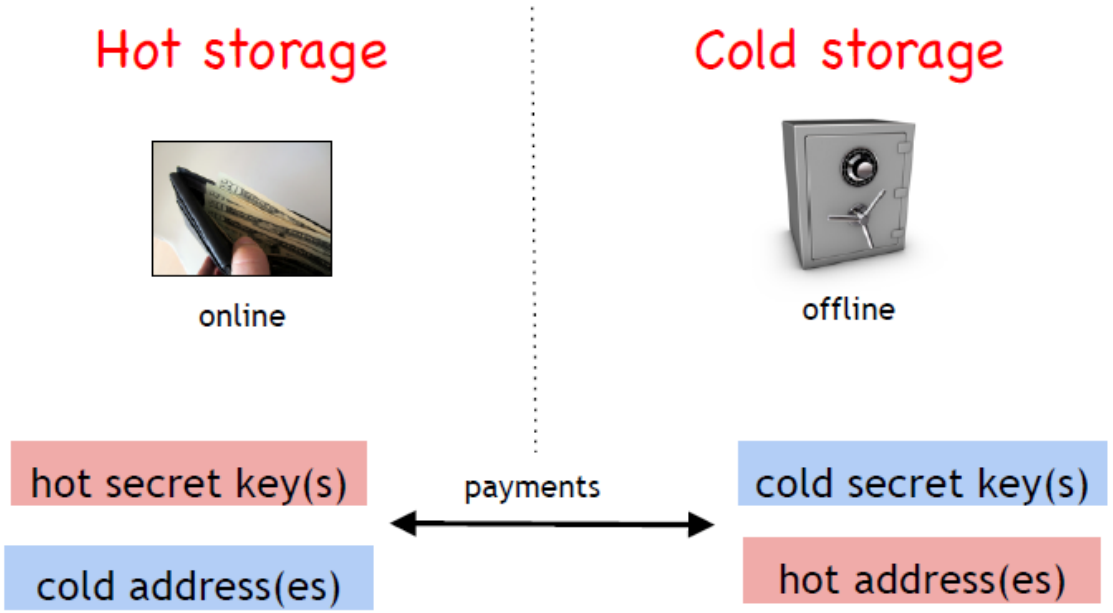
- Why separate secret keys for hot and cold sides?

Hot and Cold Storage



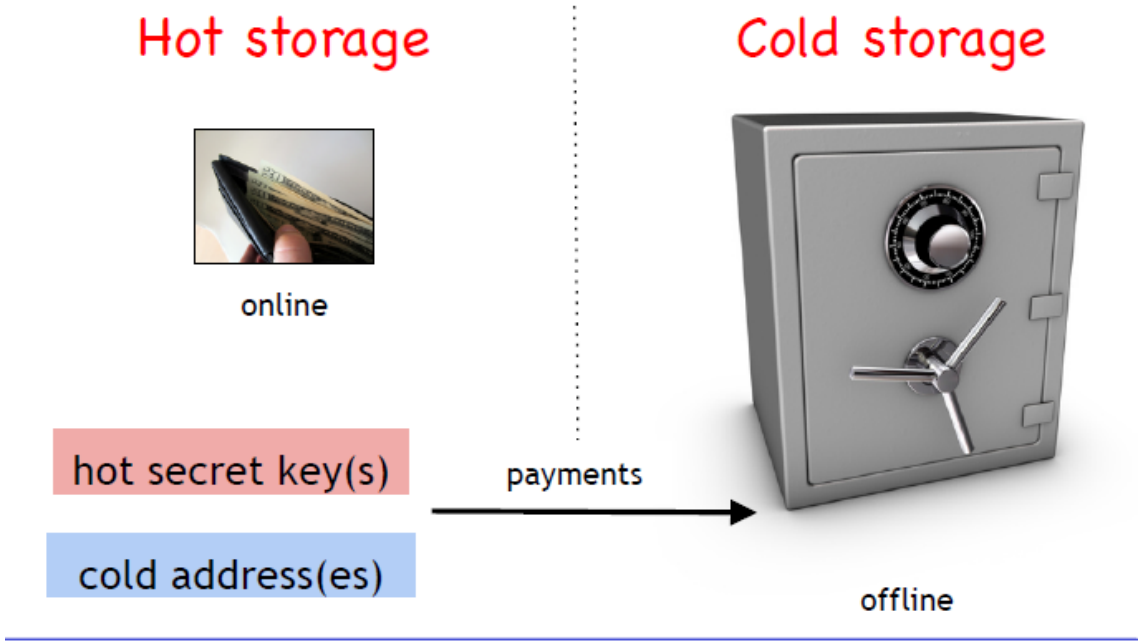
- Why separate secret keys for hot and cold sides?
  - otherwise the coins in cold storage would be vulnerable if the hot storage is compromised

Hot and Cold Storage



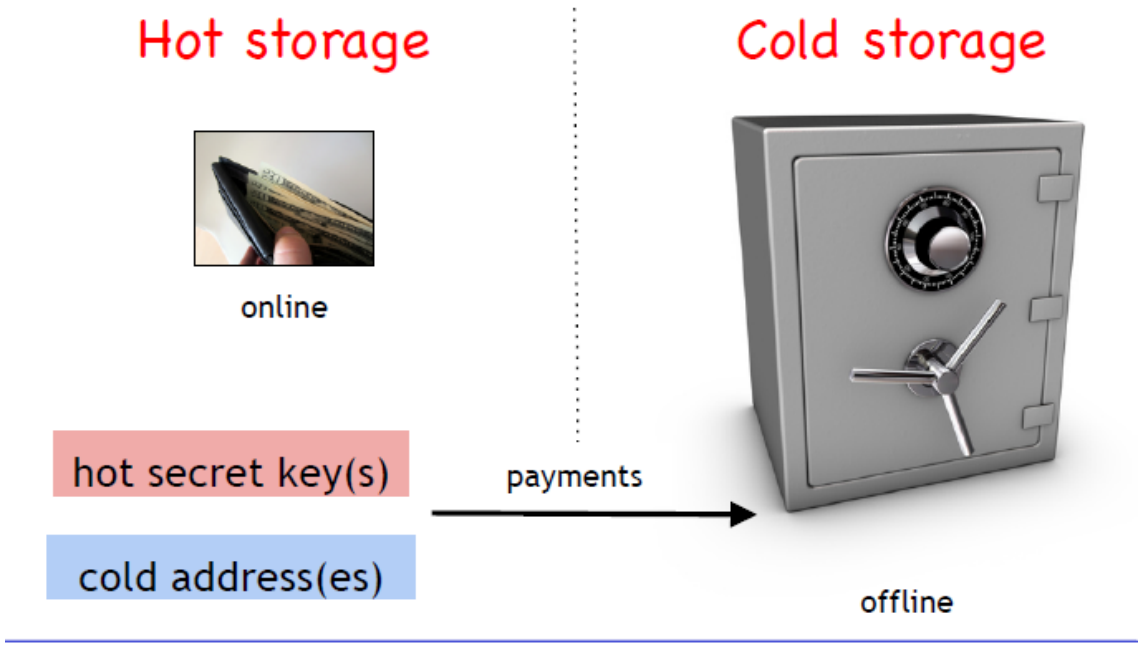
- To move coins back and forth between the two sides, each side will need to know the other's addresses

Hot and Cold Storage



- Problem
  - Want to use a new address (and key) for each coin sent to cold storage. But how can hot wallet learn new addresses if cold wallet is offline?

Hot and Cold Storage

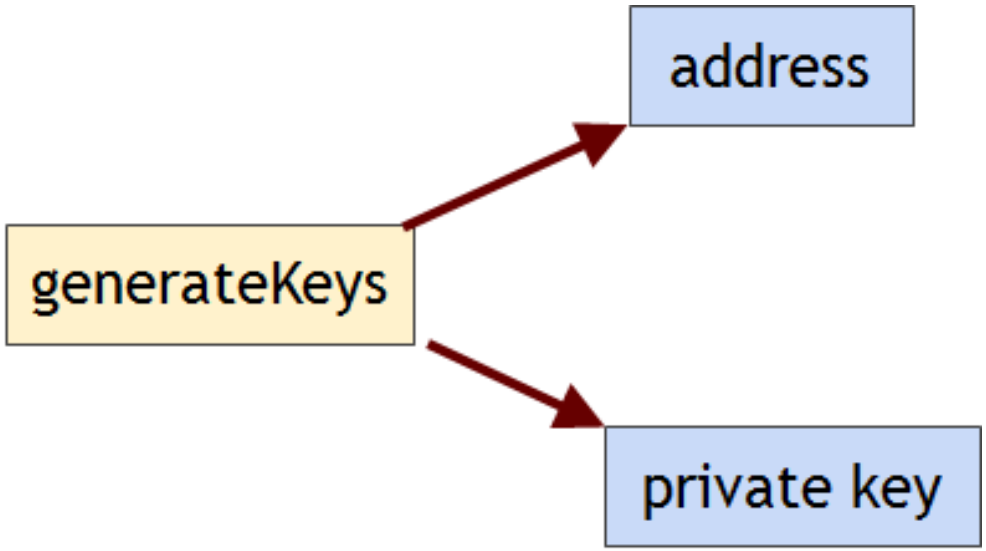


- Awkward solution
  - Generate a big batch of addresses/keys, transfer to hot beforehand

Hierarchical wallet

- It allows the cold side to use an essentially unbounded number of addresses and the hot side to know about these addresses,
- but with only a short, one-time communication between the two sides. But it requires a little bit of cryptographic trick

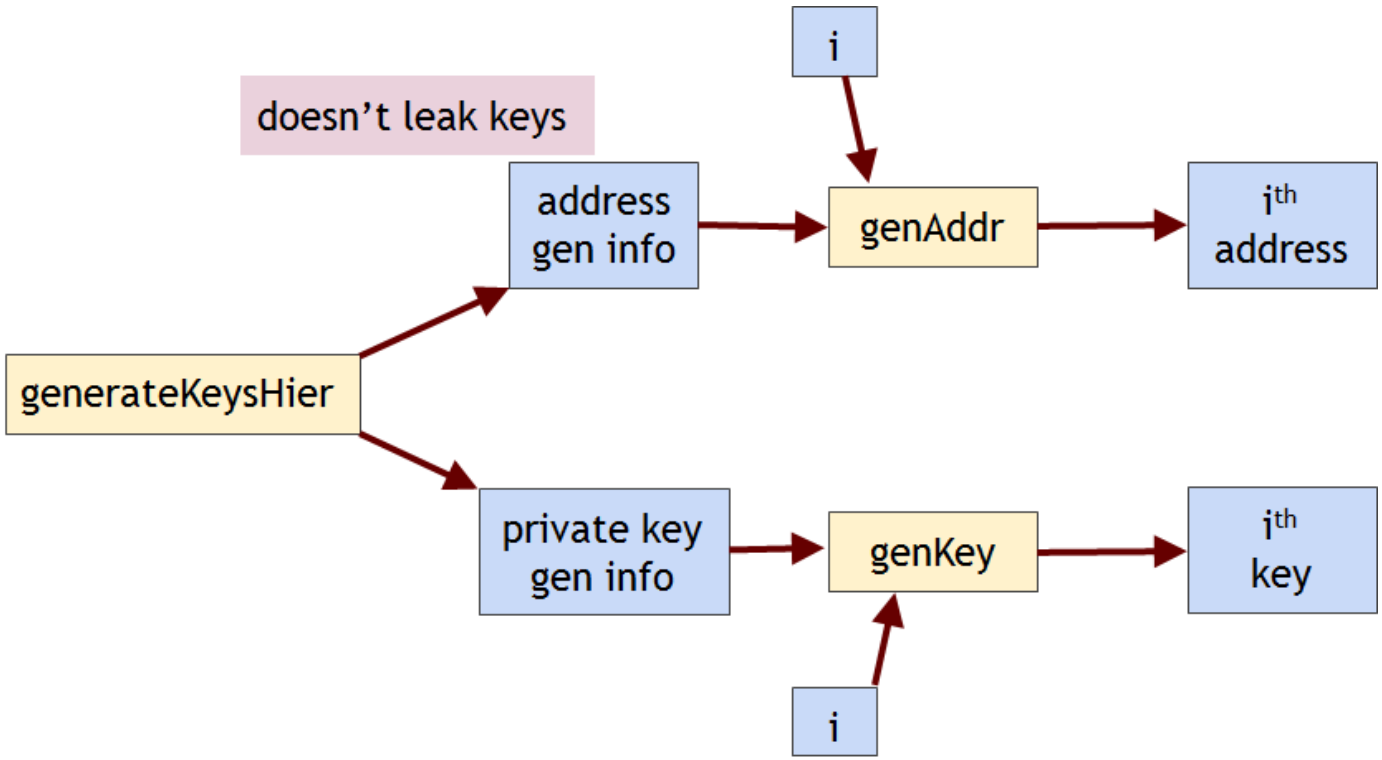
Regular key generation



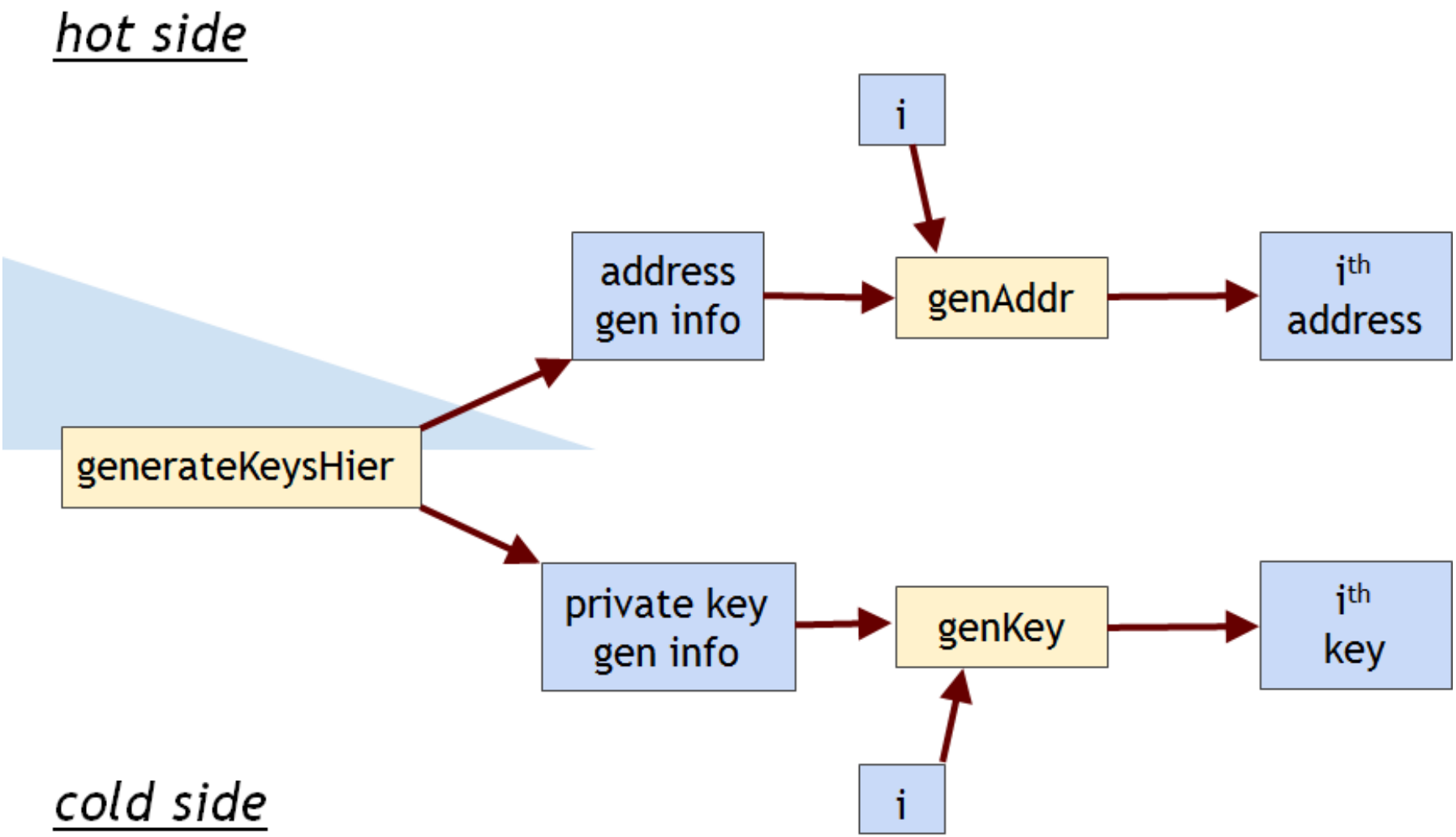
Hierarchical key generation

Given an initial address generation info, there is a function that generates a sequence of public and private keys

- For any integer i the function generates the i-th address
- and the i-secret key in the sequence
- Knowing the list of public keys does not reveal any secret key

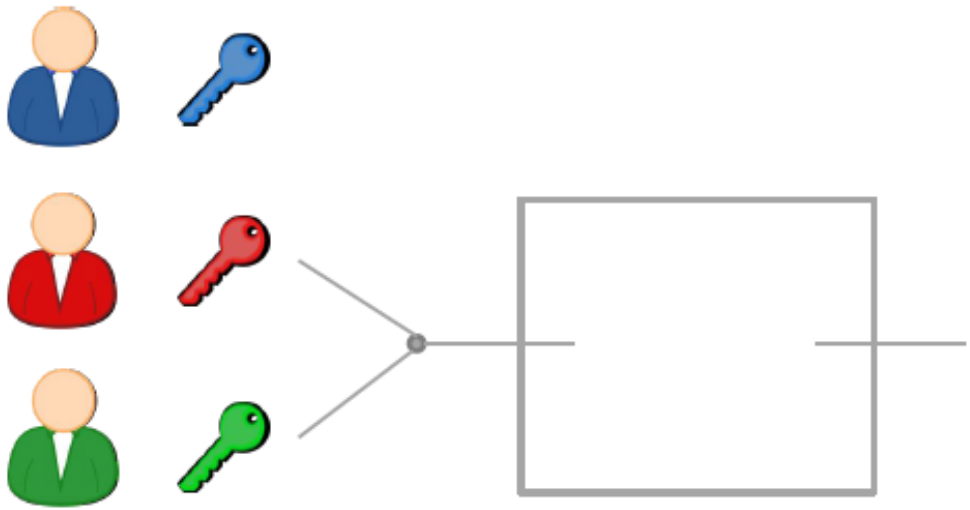


Hierarchical key generation



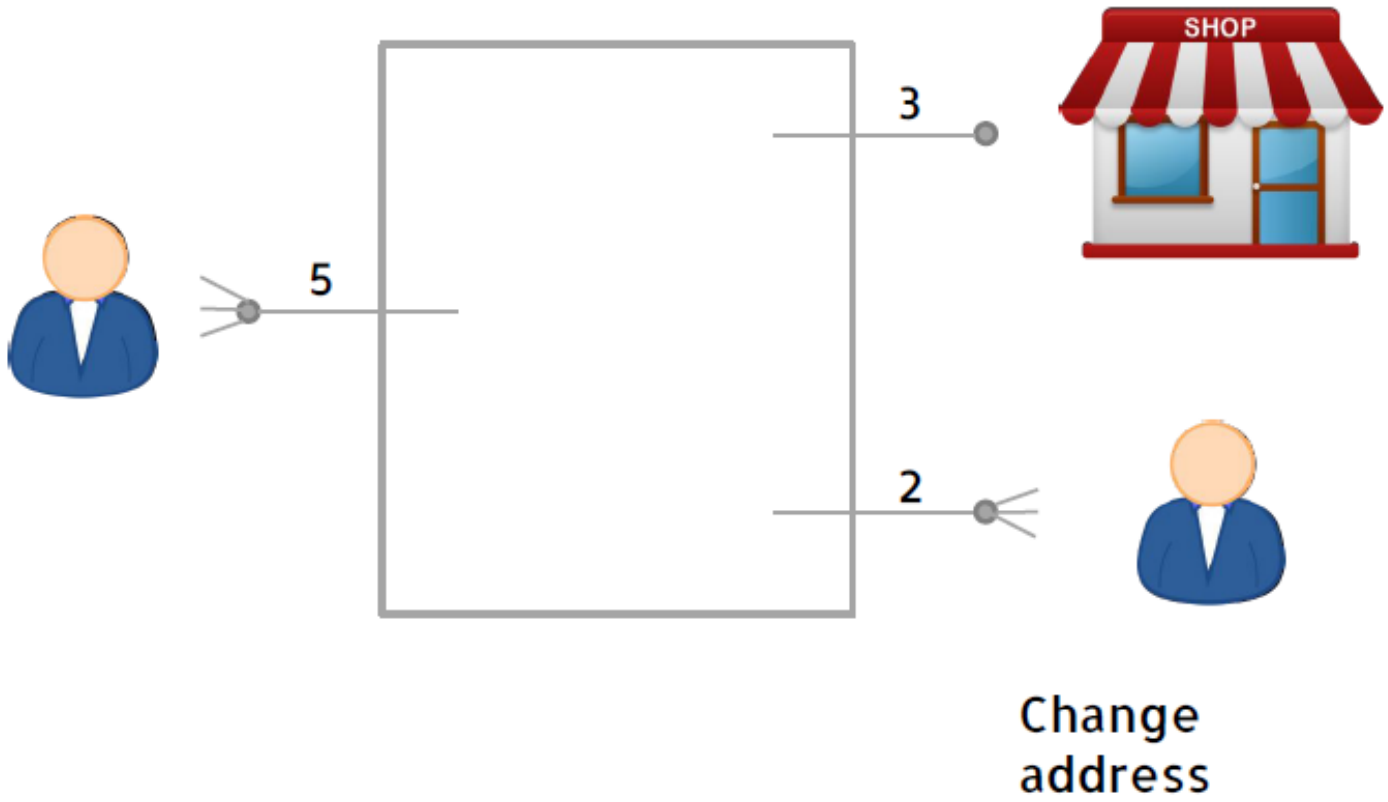
Multisig at a company

- Associate n keys with an address
- Specify a threshold t of keys that must sign to spend from that address
- Probably should be called multiple signatures!
- Example: Joint control between 3 employees



Multisig drawback

- Multisig address and change address have the same access control structure
- Causes loss of anonymity!



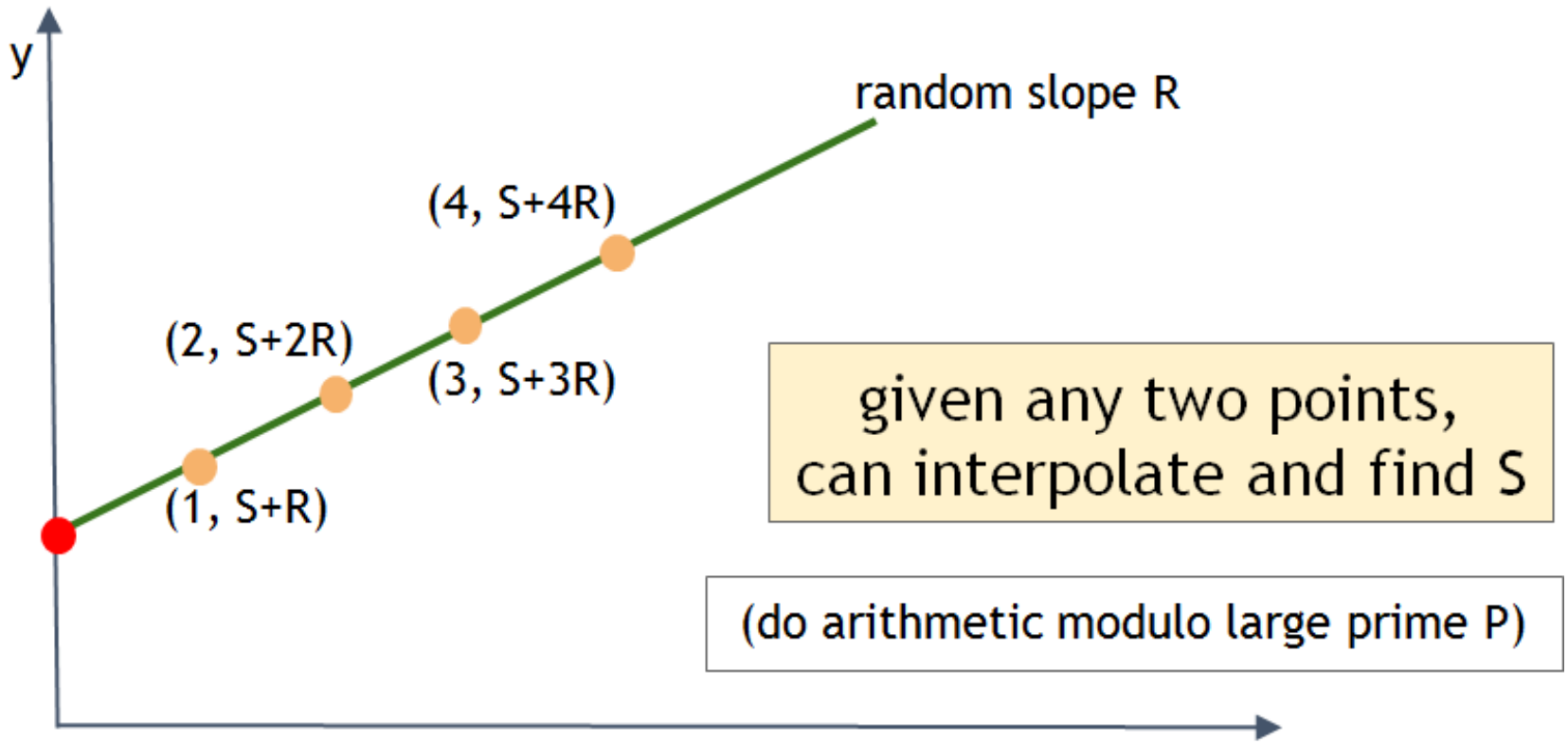
Threshold signatures

- Multisignatures don't split your keys
- Threshold signatures do
- Don't broadcast your security policy to the world
- Sign transactions without reconstructing key in a single place![3]

[3] Goldfeder et al.

Secret sharing

- Goal: split secret into N pieces, such that
  - given any K pieces, can reconstruct the secret
  - given fewer than K pieces, don't learn anything
- Example: $N = 4$, $K = 2$



Digital Signature Algorithm (DSA)

Given:

- a group G of order N
- a generator g
- a private key x

To sign a message m:

- pick a nonce k s.t. $1 \leq k \leq N-1$
- $r = g^k$
- $s = k^{-1}(m + xr) \bmod N$
- Signature is (r, s).

Verification of signature (r, s) on m:

- Check $r^s = g^m y^r$

Randomness in DSA

- What are the unknowns in the signature element s?

$$s = k^{-1}(m + xr) \bmod N$$

- What happens if the same k is used twice?
- Nobody told Sony about this in 2010

Bad randoms in Bitcoin blockchain

- First publicized by Nils Schneider
- There was a bug in Android bitcoin code in 2013 . . .
  - The actual bug was in the random number generator library where it would occasionally return the same random number twice!?!?
  - So if you did two back-to-back transactions . . .
- Somebody noticed this
  - And set up a bot to look for this automatically:
    When this happens the money is stolen!

Deterministic DSA (and ECDSA)

- Avoids attacks with bad RNG
- Derives k deterministically from the private key and the message hash, as described by RFC 6979 [Pornin]

BIP32 key generation

Given

- a group G of order N
- a generator g
- a master private key x
- a master public key $y = g^x$

To generate child private keys:

$$x_i = x + hash(i, y)$$

To generate child public keys:

$$y_i = y \cdot g^{hash(i,y)}$$

Key leakage[4]

Show that if the master public key y and any child private key x_i are known, then the private key x can be easily recovered.

$$x = x_i - hash(i, y)$$

[4] Gutoski and Stebila, FC'15

Property of hashed public keys

- Many cryptographic systems in use today are vulnerable to quantum computing attacks
  - Given the public key it is trivial to find the private key (Shor's factoring algorithm)
- If you never use a private key more than once . . .
  - By instead transferring all unspent money to a new random private key
  - A quantum computer can't steal your money!
- This is how Satoshi "dodged" some future attacks!

Bitcoin and quantum computing

Louis Tessler[2,6] and Tim Byrnes[1,2,3,4,5]

1 State Key Laboratory of Precision Spectroscopy, School of Physical and Material Sciences,

East China Normal University, Shanghai 200062, China

2 New York University Shanghai, 1555 Century Ave, Pudong, Shanghai 200122, China

3 NYU-ECNU Institute of Physics at NYU Shanghai,

3663 Zhongshan Road North, Shanghai 200062, China

4 National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

5 Department of Physics, New York University, New York, NY 10003, USA

6 CEMS, RIKEN, Wako-shi, Saitama 351-0198, Japan

Abstract

Bitcoin is a digital currency and payment system based on classical cryptographic technologies which works without a central administrator such as in traditional currencies. It has long been questioned what the impact of quantum computing would be on Bitcoin, and cryptocurrencies in general. Here, we analyse three primary directions that quantum computers might have an impact in: mining, security, and forks. We find that in the near-term the impact of quantum computers appear to be rather small for all three directions. The impact of quantum computers would require considerably larger number of qubits and breakthroughs in quantum algorithms to reverse existing hash functions.

More information

- Narayanan, Bonneau, Felten, Miller, Goldfeder, Clark, Bitcoin and Cryptocurrency Technologies http://bitcoinbook.cs.princeton.edu/
- Goldfeder, Gennaro, Kalodner, Bonneau, Kroll, Felten, Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme.
- Pornin, Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), 2013.
- Gutoski and Stebila, Hierarchical Deterministic Bitcoin Wallets that Tolerate Key Leakage, Financial Cryptography 2015.
- Eskandari, Barrera, Stobert, Clark, A First Look at the Usability of Bitcoin Key Management, Workshop on Usable Security 2015.
- Tessler, Byrnes, Bitcoin and quantum computing, published 2017 in ArXiv.