Lecture 7

Bitcoin Apps and Security

To spend a Bitcoin, you need to know:

* some info from the public blockchain, and

* the owner’s secret signing key

So it’s all about key management.

Goals (for Bitcoin Key management)

availability: You can spend your coins.

security: Nobody else can spend your coins.

convenience

<u>Simplest approach</u>: store key in a file,
on your computer or phone

- Very convenient
- As available as your device.
  - device lost/wiped ⇒ key lost ⇒ coins lost
- As secure as your device
  - device compromised ⇒ key leaked ⇒ coins stolen

Simplest approach: store key in a file, on your computer or phone

- Very convenient
- As available as your device.
  - device lost/wiped ⇒ key lost ⇒ coins lost
- As secure as your device
  - device compromised ⇒ key leaked ⇒ coins stolen

Wallet software

Keeps track of your coins, provides nice user interface.

Nice trick: use a separate address/key for each coin.

benefits privacy (looks like separate owners)

wallet can do the bookkeeping, user needn't know

Encoding addresses

Encode as text string: base58 notation

123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz

or use QR code

Hot and Cold Storage

Hot storage

online

convenient but risky

Cold storage

offline

archival but safer

← separate keys →

Hot storage

online

hot secret key(s)

Cold storage

offline

cold secret key(s)

Hot storage

online

hot secret key(s)

cold address(es)

Cold storage

offline

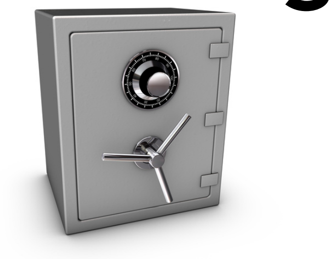
cold secret key(s)

hot address(es)

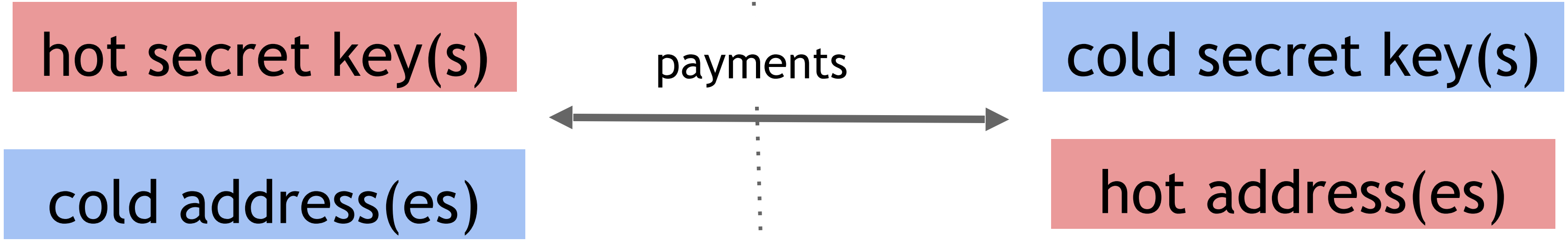
Hot storage

online

Cold storage



offline



hot secret key(s)

cold address(es)

payments

cold secret key(s)

hot address(es)

Hot storage

online

hot secret key(s)

Cold storage

offline

Hot storage

online

hot secret key(s)

cold address(es)

Cold storage

offline

Hot storage

online

hot secret key(s)

cold address(es)

payments

Cold storage

offline

Problem:

Want to use a new address (and key) for each coin sent to cold

But how can hot wallet learn new addresses if cold wallet is offline?

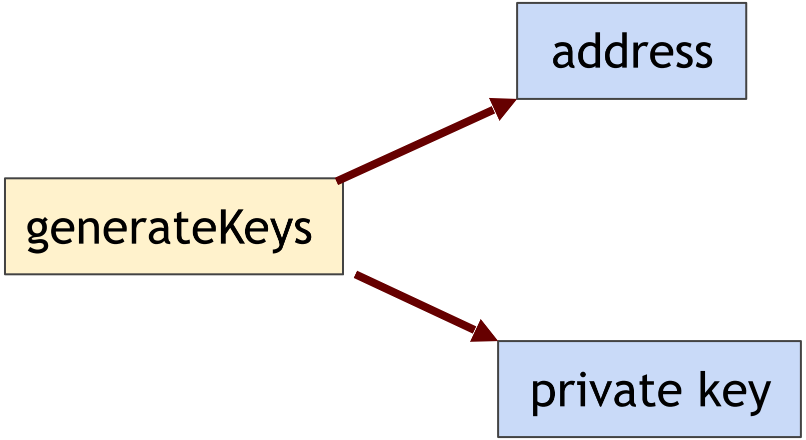
Strawman solution:

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

Better solution:

Hierarchical wallet

Regular key generation:

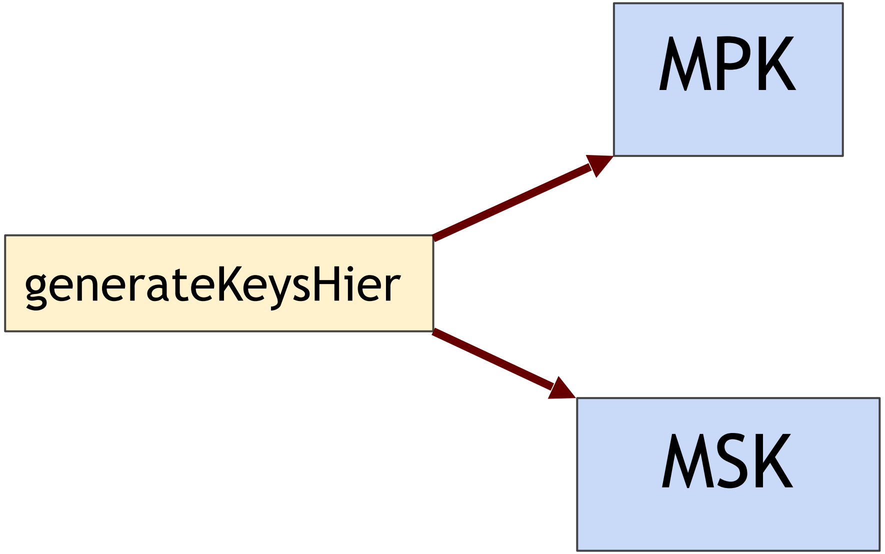


Hierarchical key generation:

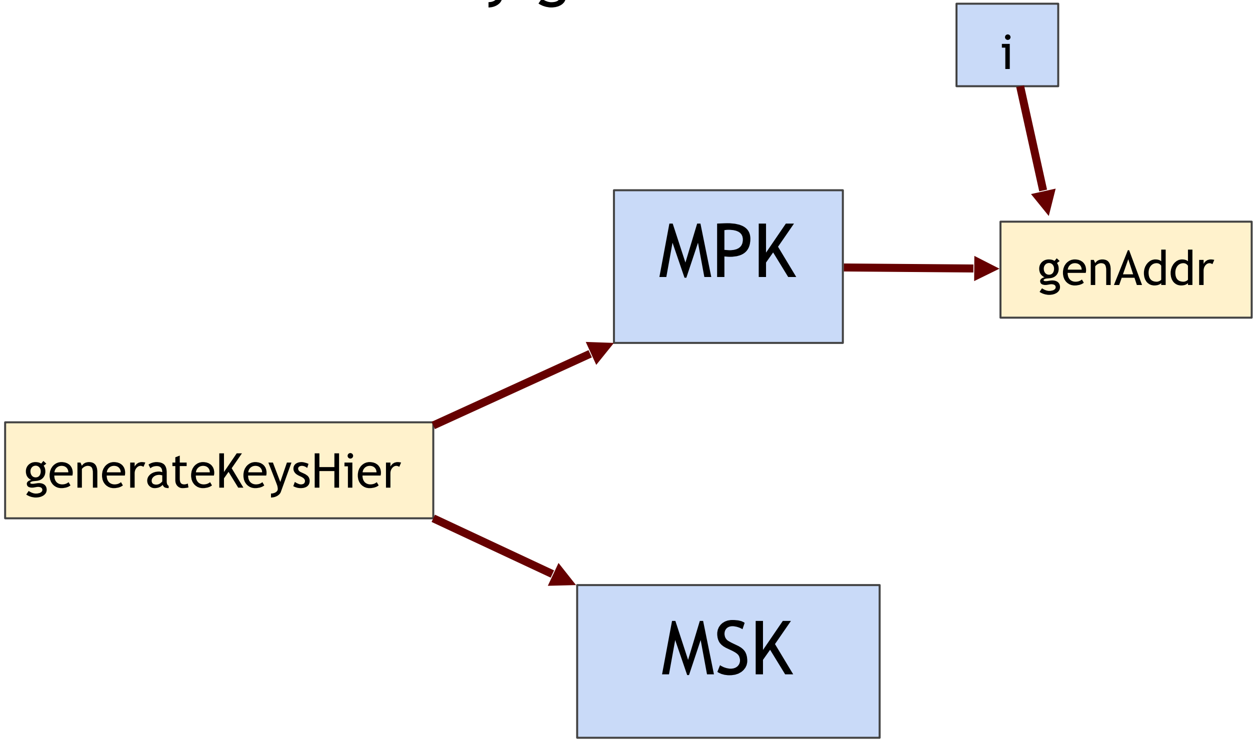
Hierarchical key generation:

generateKeysHier

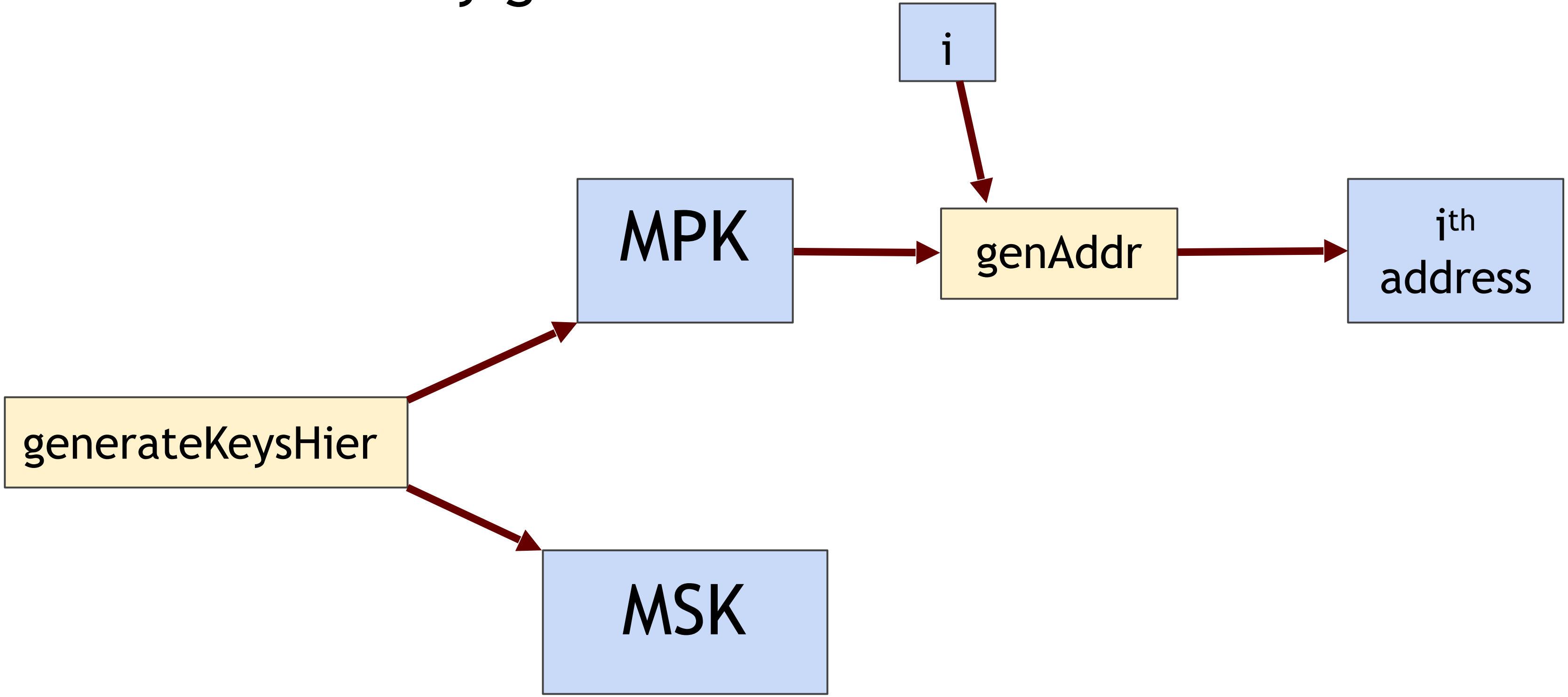
Hierarchical key generation:



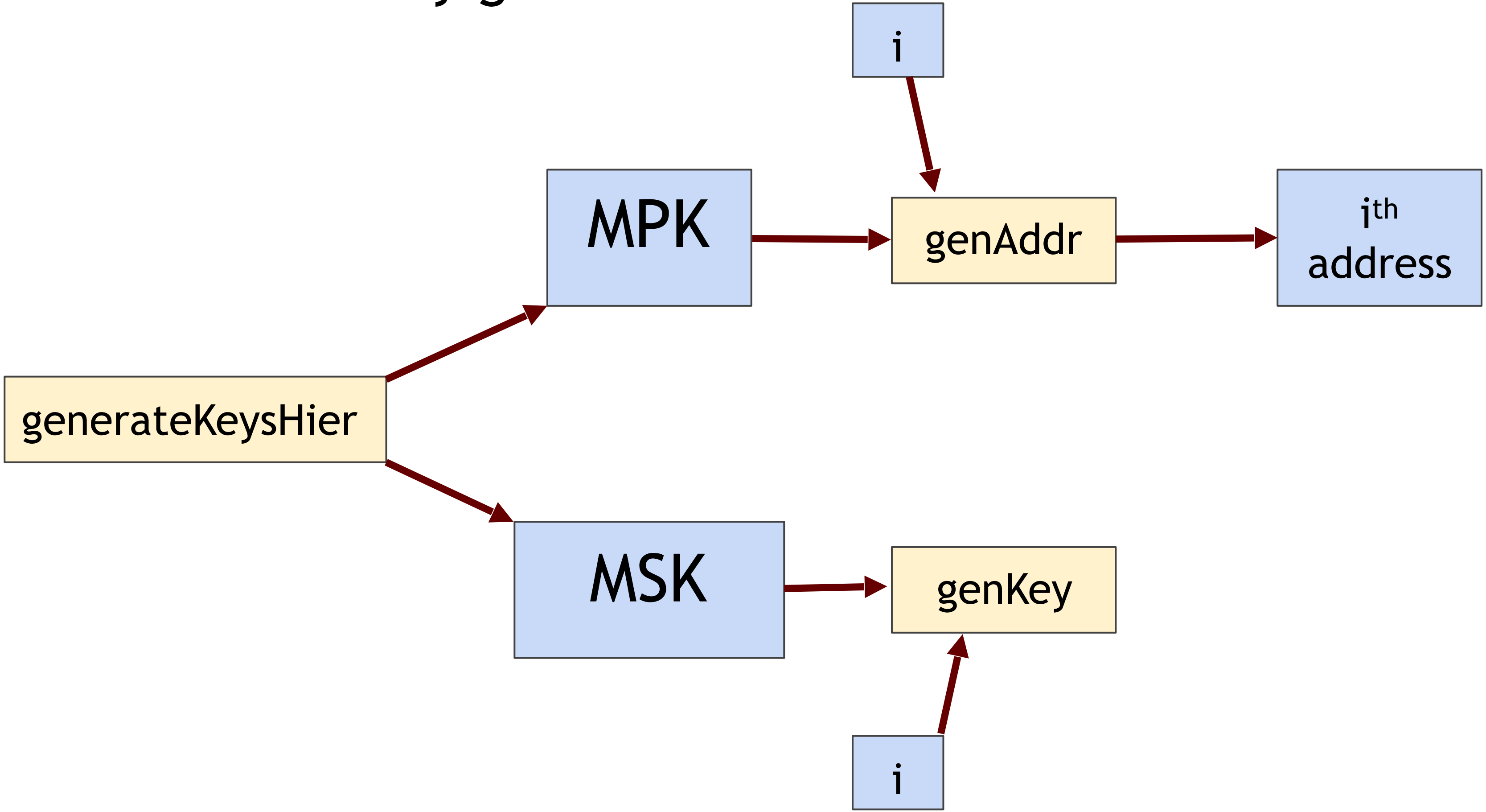
Hierarchical key generation:



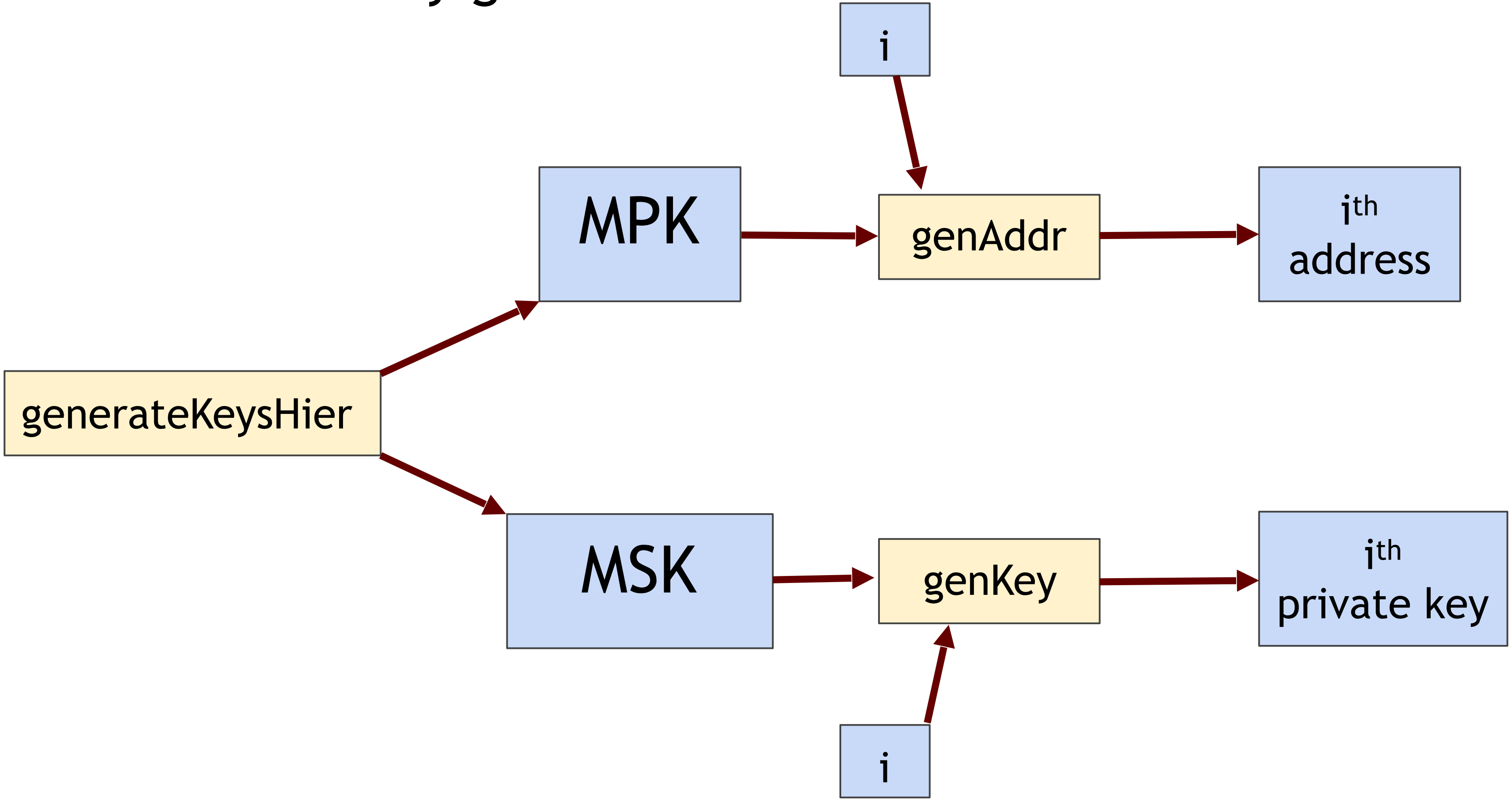
Hierarchical key generation:



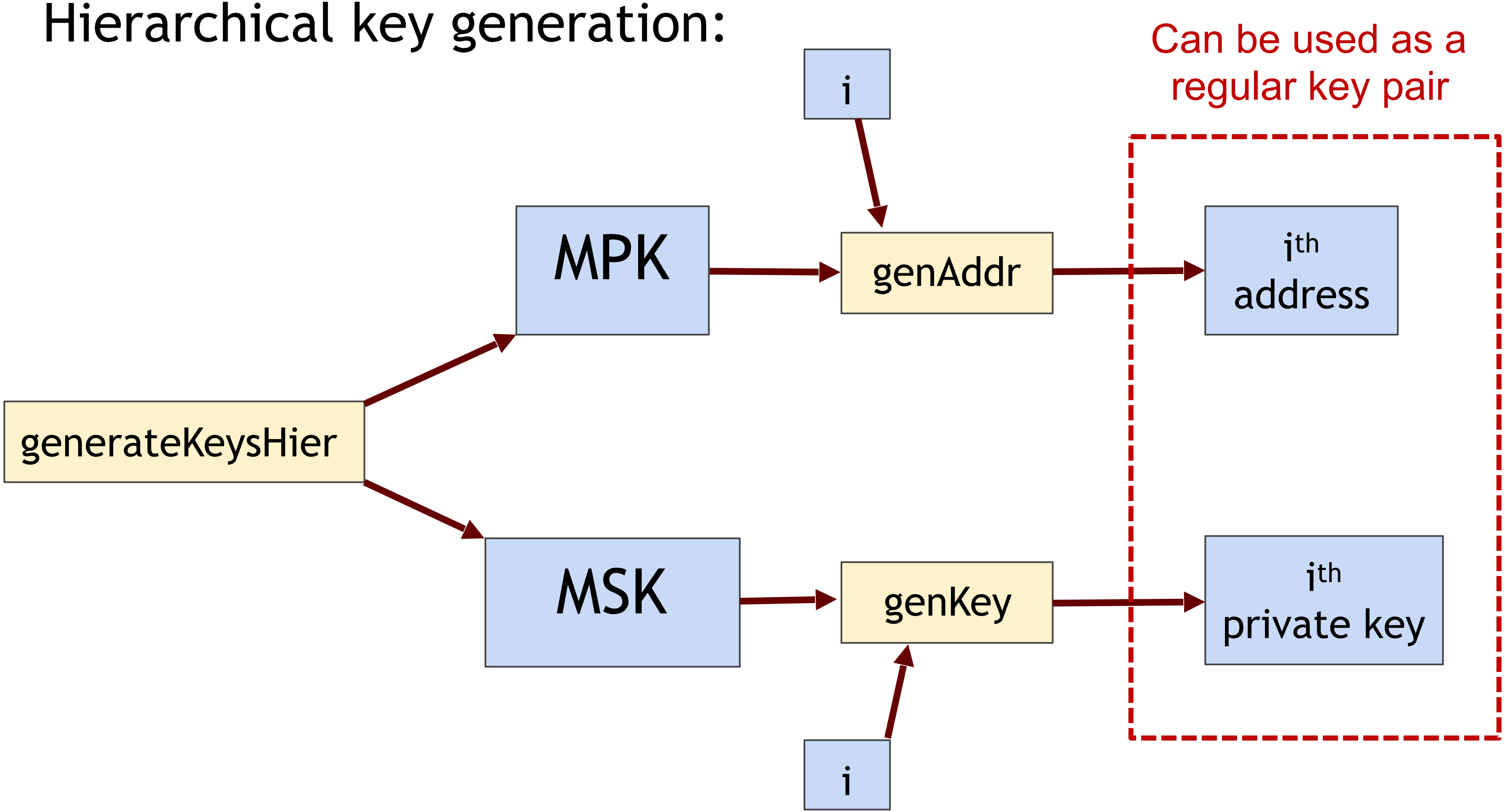
Hierarchical key generation:



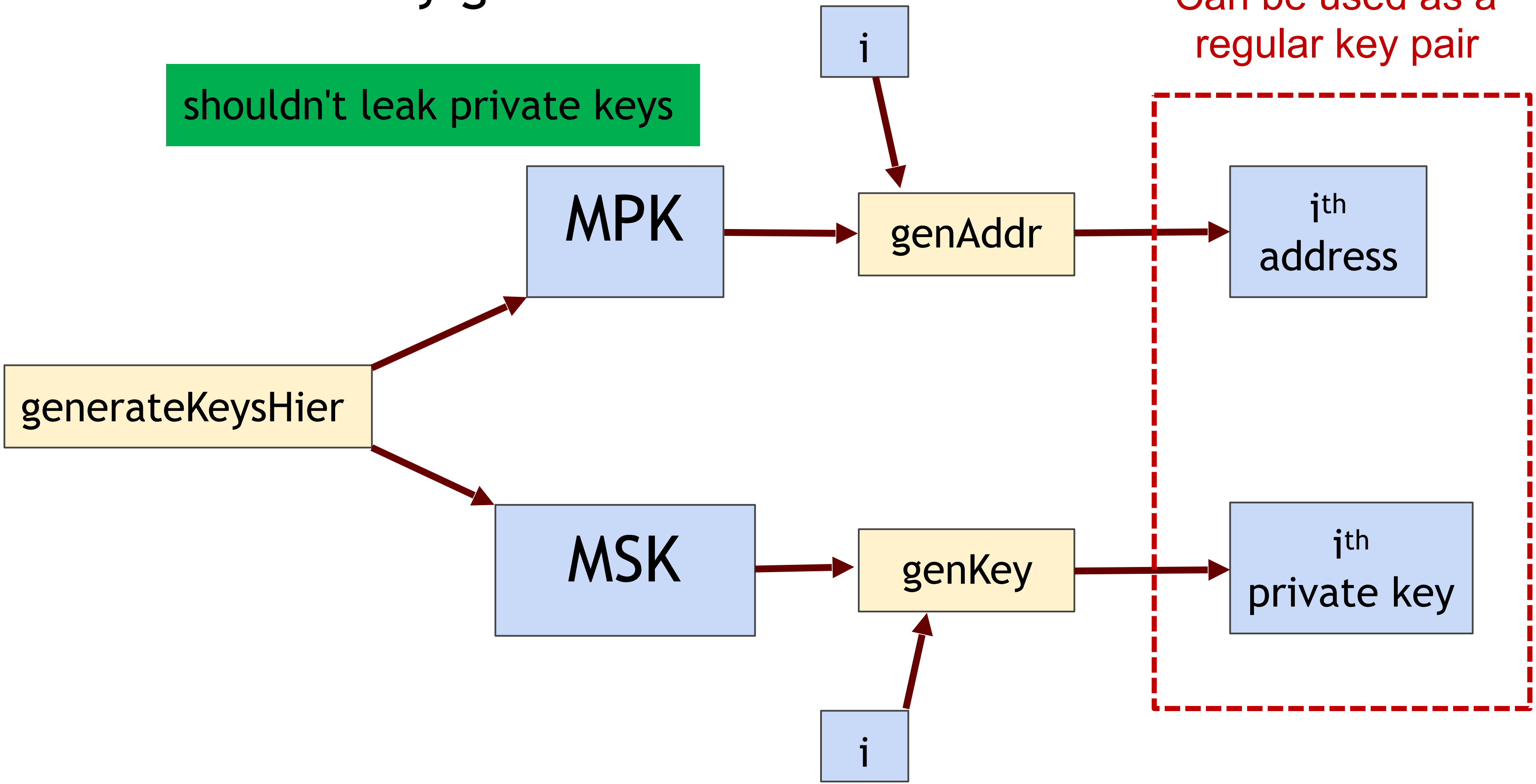
Hierarchical key generation:



Hierarchical key generation:



Hierarchical key generation:



Implementation for EC-DSA [BIP32]

- generateKeysHier(1^n): MSK = x, MPK = g^x , H (hash function)
- genAddr(MPK,i): r = H(i,MPK), $addr_i$ = MPK * g^r
- genKey(MSK,i): r = H(i,MPK), sk_i = MSK + r

Implementation for EC-DSA [BIP32]

- Not resilient to key leakage: given (i, sk_i, MPK), easy to determine MSK (therefore, no “forward” or “backward” privacy)

Implementation for EC-DSA [BIP32]

- Not resilient to key leakage: given (i, sk_i, MPK), easy to determine MSK (therefore, no “forward” or “backward” privacy)
    - MSK = sk_i - H(i,MPK)

Implementation for EC-DSA [BIP32]

- Not resilient to key leakage: given (i, sk_i, MPK), easy to determine MSK (therefore, no “forward” or “backward” privacy)
  - MSK = sk_i - H(i,MPK)
  - Source of vulnerability: the “re-randomization” randomness is publicly computable

Implementation for EC-DSA [BIP32]

- Not resilient to key leakage: given (i, sk_i, MPK), easy to determine MSK (therefore, no “forward” or “backward” privacy)
  - MSK = sk_i - H(i,MPK)
  - Source of vulnerability: the “re-randomization” randomness is publicly computable

- BIP32 gives an alternative construction where key leakage can be tolerated, but it does not support hierarchical addresses (i.e., only hierarchical private keys, but address remains fixed)

Implementation for EC-DSA [BIP32]

- Not resilient to key leakage: given (i, sk_i, MPK), easy to determine MSK (therefore, no "forward" or "backward" privacy)
  - MSK = sk_i - H(i,MPK)
  - Source of vulnerability: the "re-randomization" randomness is publicly computable

- BIP32 gives an alternative construction where key leakage can be tolerated, but it does not support hierarchical addresses (i.e., only hierarchical private keys, but address remains fixed)

- Question: Hierarchical construction that tolerates key leakage?

[Gutoski-Stebila, FC'15])

- Resilience to leakage of **m** keys: MSK remains secret even if adversary gets access to m private keys

[Gutoski-Stebila, FC'15])

- Resilience to leakage of **m** keys: MSK remains secret even if adversary gets access to m private keys

$$\mathbf{MPK} = (MPK_1, \ldots, MPK_m); \mathbf{MSK} = (MSK_1, \ldots, MSK_m)$$

$$sk_i = \textstyle\sum_{j=1}^{m} \alpha_j \cdot MSK_j \quad \text{where} \quad H(i, \mathbf{MPK}) = (\alpha_1, \ldots, \alpha_m)$$

[Gutoski-Stebila, FC'15])

- Resilience to leakage of **m** keys: MSK remains secret even if adversary gets access to m private keys

$$\mathbf{MPK} = (MPK_1, \ldots, MPK_m); \mathbf{MSK} = (MSK_1, \ldots, MSK_m)$$

$$sk_i = \textstyle\sum_{j=1}^{m} \alpha_j \cdot MSK_j \quad \text{where} \quad H(i, \mathbf{MPK}) = (\alpha_1, \ldots, \alpha_m)$$

- Security against "full break" can be based on "one-more Discrete Log" problem

[Gutoski-Stebila, FC’15])

- Resilience to leakage of **m** keys: MSK remains secret even if adversary gets access to m private keys

$$\mathbf{MPK} = (MPK_1, \ldots, MPK_m); \mathbf{MSK} = (MSK_1, \ldots, MSK_m)$$

$$sk_i = \sum_{j=1}^{m} \alpha_j \cdot MSK_j \quad \text{where} \quad H(i, \mathbf{MPK}) = (\alpha_1, \ldots, \alpha_m)$$

- Security against “full break” can be based on “one-more Discrete Log” problem
- Main Drawback: **m** must be fixed in advance, size of MPK grows with **m**

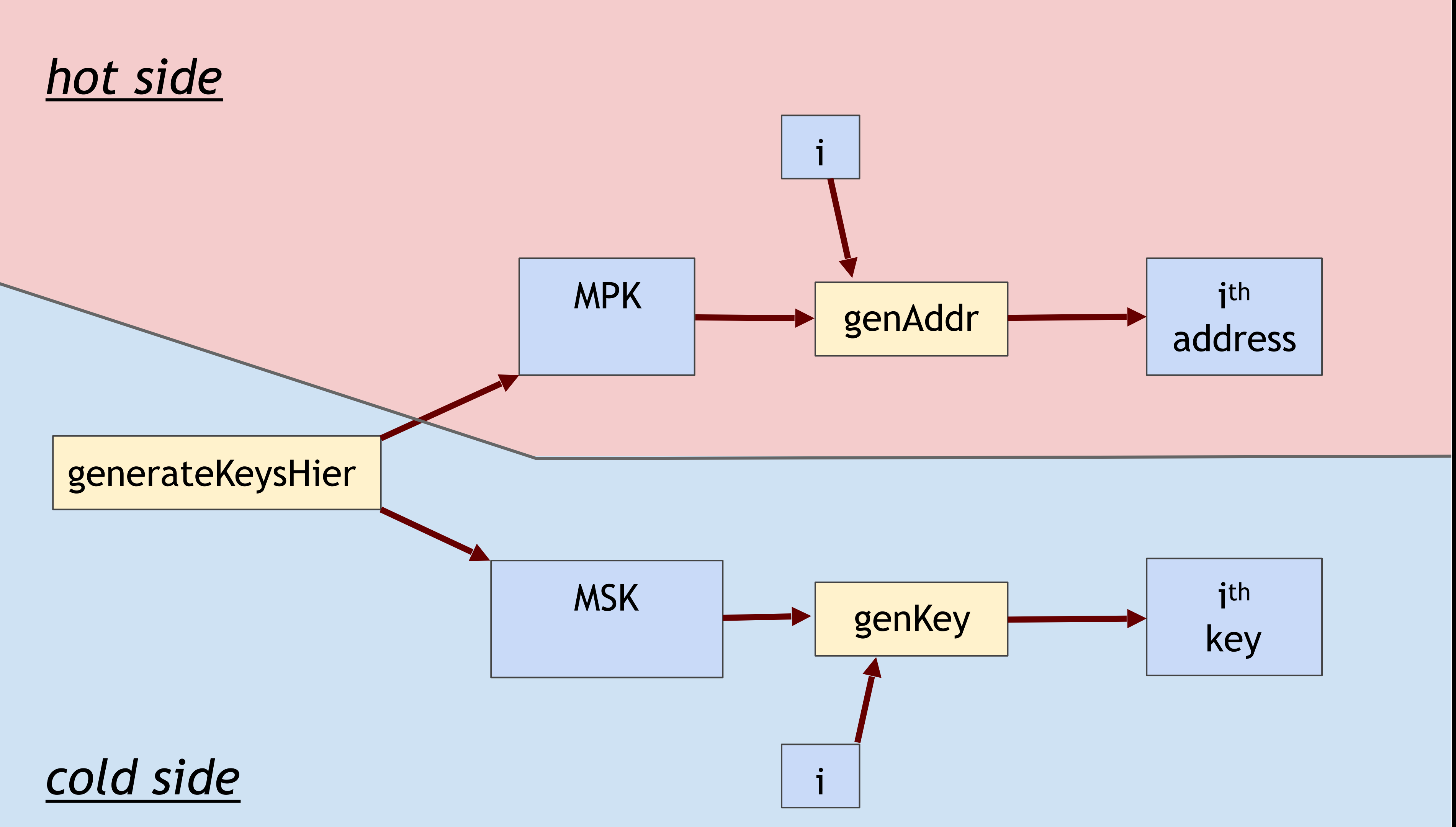
[Gutoski-Stebila, FC'15])

- Resilience to leakage of **m** keys: MSK remains secret even if adversary gets access to m private keys

$$\mathbf{MPK} = (MPK_1, \ldots, MPK_m); \mathbf{MSK} = (MSK_1, \ldots, MSK_m)$$

$$sk_i = \sum_{j=1}^{m} \alpha_j \cdot MSK_j \quad \text{where} \quad H(i, \mathbf{MPK}) = (\alpha_1, \ldots, \alpha_m)$$

- Security against "full break" can be based on "one-more Discrete Log" problem
- Main Drawback: **m** must be fixed in advance, size of MPK grows with **m**
- **Question**: Hierarchical construction that achieves forward/backward privacy with any leakage?



hot side

i

MPK

genAddr

i^{th} address

generateKeysHier

MSK

genKey

i^{th} key

i

cold side

How to store cold info

- Info stored in device, device locked in a safe
- “Brain wallet”: encrypt info under password phrase that user remembers
- Paper wallet: print info on paper, lock up the paper
- Tamperproof device: device will sign things for you, but won’t divulge keys

Splitting and Sharing Keys

Secret sharing

(k,n)-secret sharing: Divide a secret value S into n shares $S_1,\ldots,S_n$ such that:

- **Correctness**: *Any* k shares can be used to reconstruct S
- **Privacy**: S is hidden given at most k-1 shares

Secret sharing [Shamir]

- **Share(S):** Output a tuple S_1,…,S_n
- **Reconstruct(x_1,…,x_k):** Output a value S*

k-Privacy: For any (S,S'), and any subset X of < k indices, the following two distributions are statistically close:

$$\{(S_1, \ldots, S_n) \leftarrow Share(S) : (S_i | i \in X)\},$$

$$\{(S'_1, \ldots, S'_n) \leftarrow Share(S') : (S'_i | i \in X)\}.$$

Example: n=2, k=2

- p = a large prime
- S = secret in [0, P)
- R = random in [0, P)

Share(S):

x_1 = (S+R) mod p x_2 = (S+2R) mod p

Reconstruct(x_1,x_2):

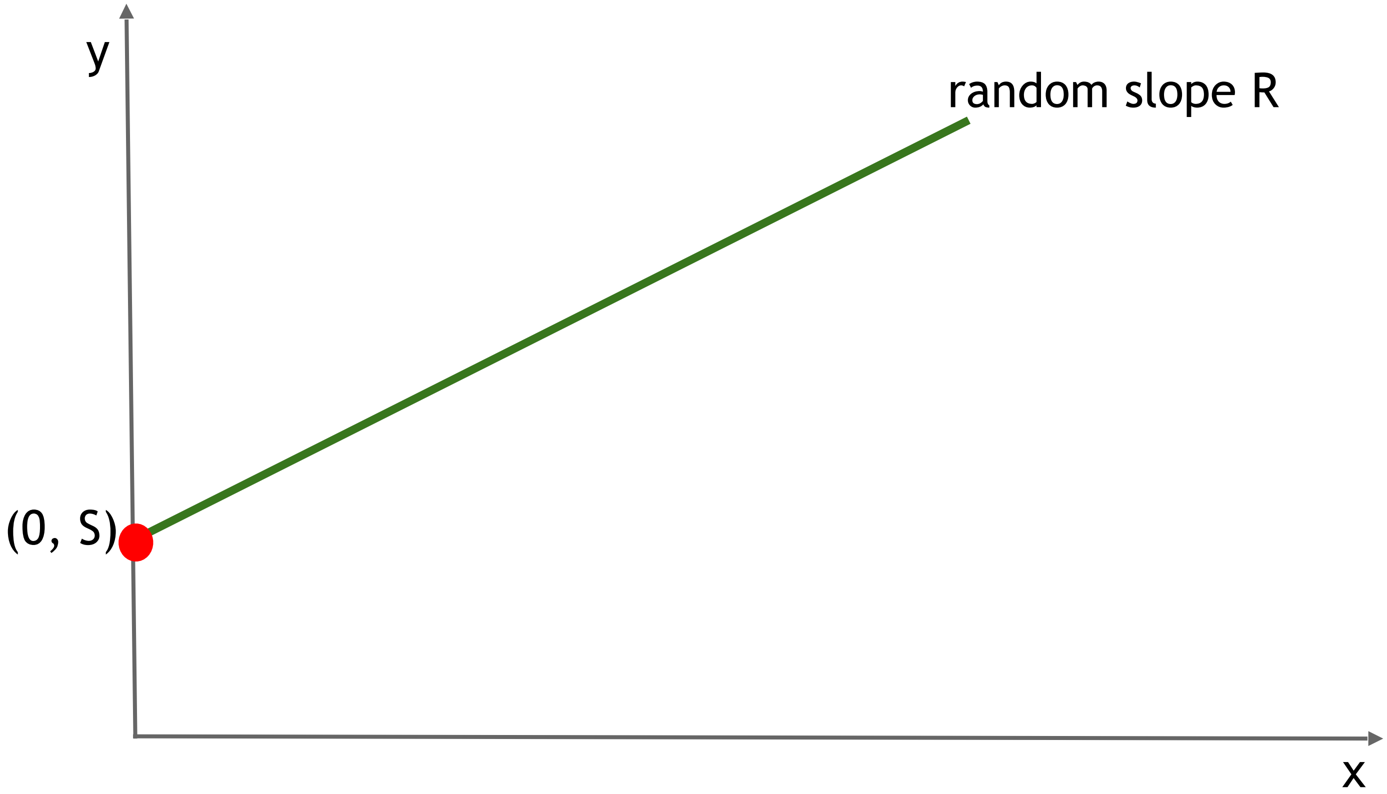
($2x_1$-x_2) mod p = S

<u>2-Privacy</u>: each x_i has uniform distribution over [0,P); independent of S

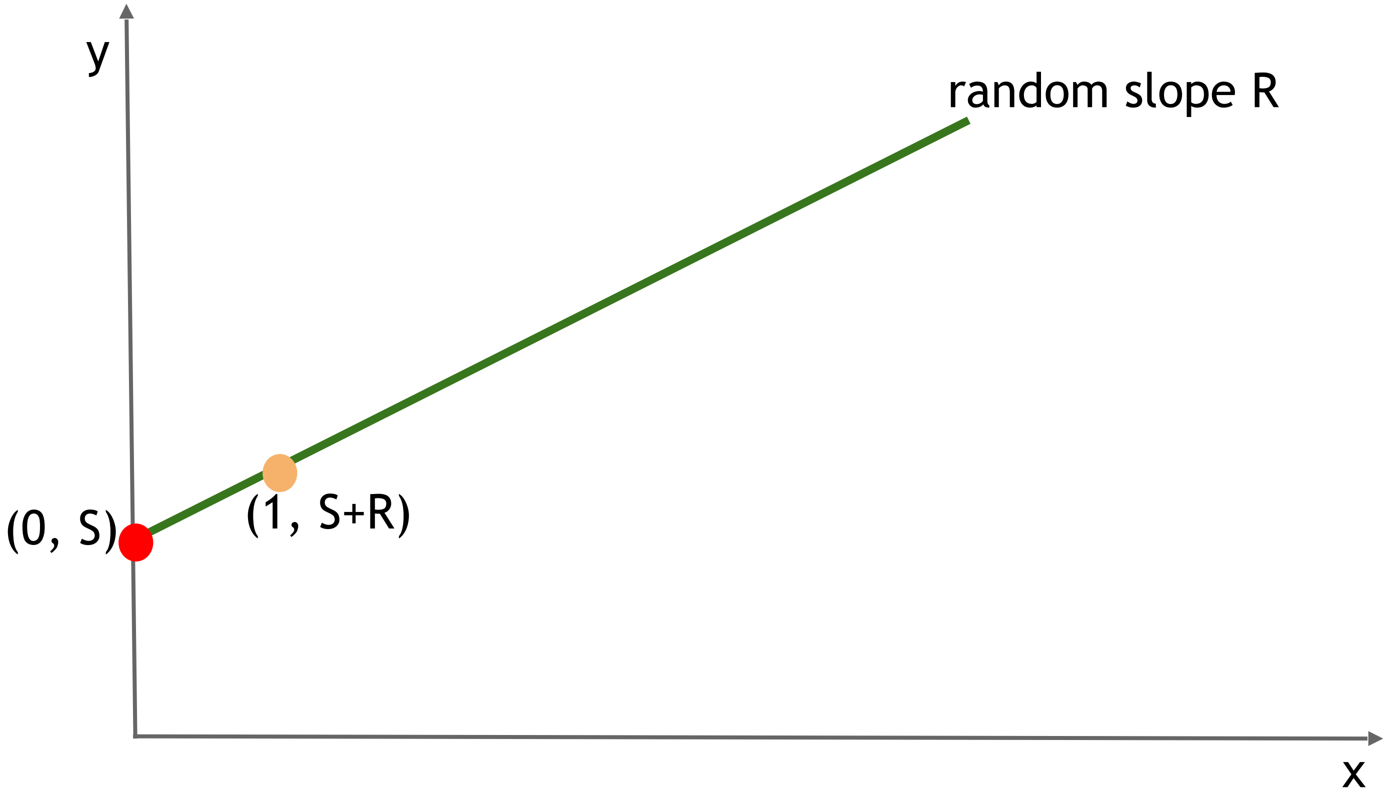
Example: k = 2, n > 2

Example: k = 2, n > 2

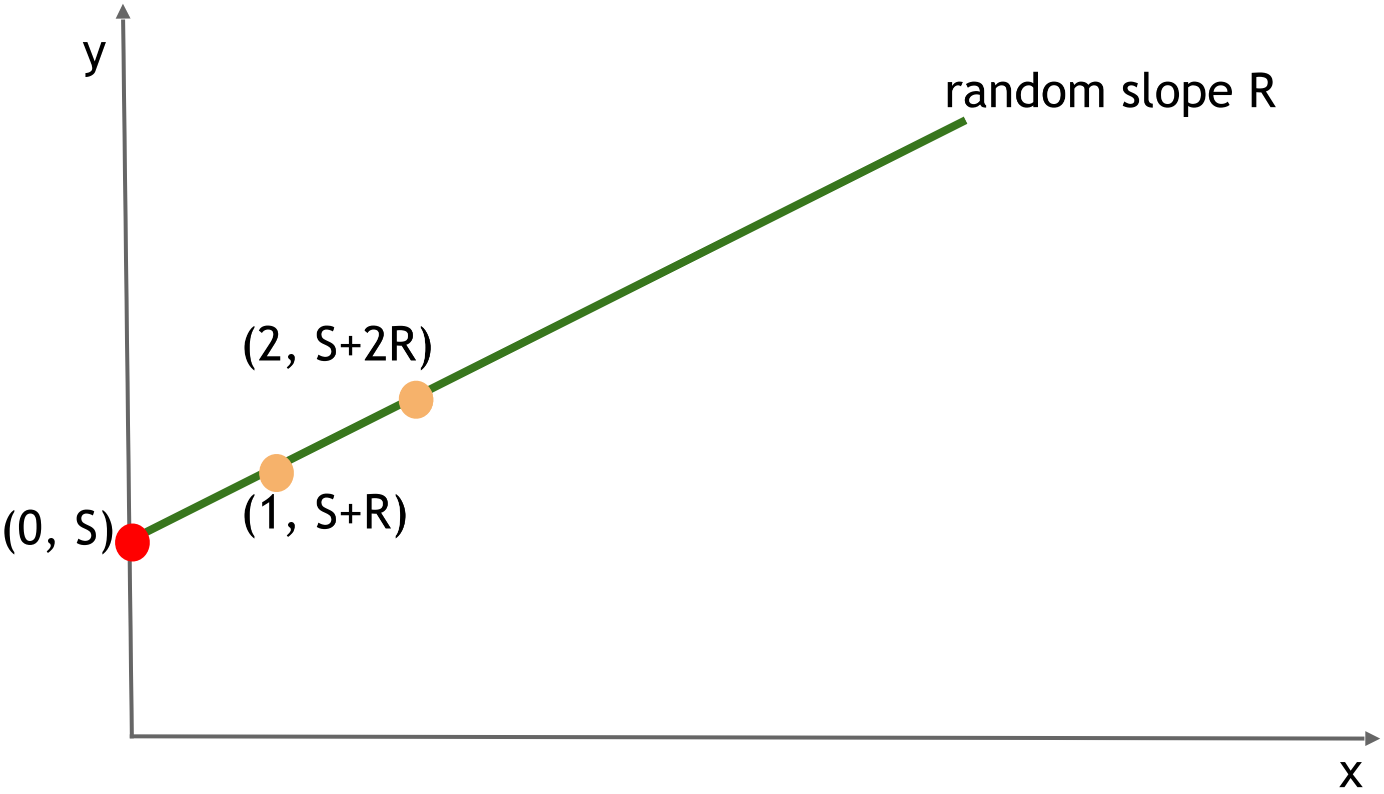
Example: k = 2, n > 2



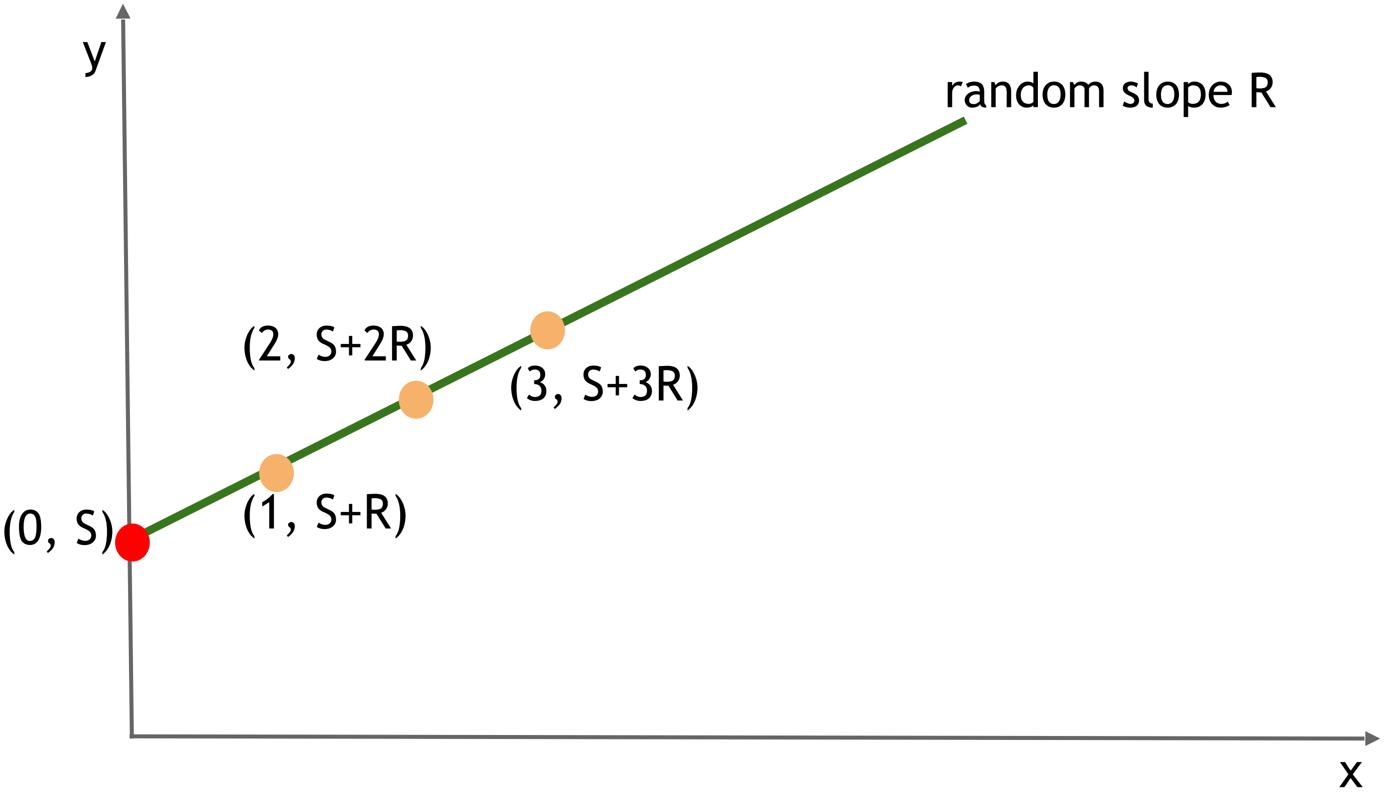
Example: k = 2, n > 2



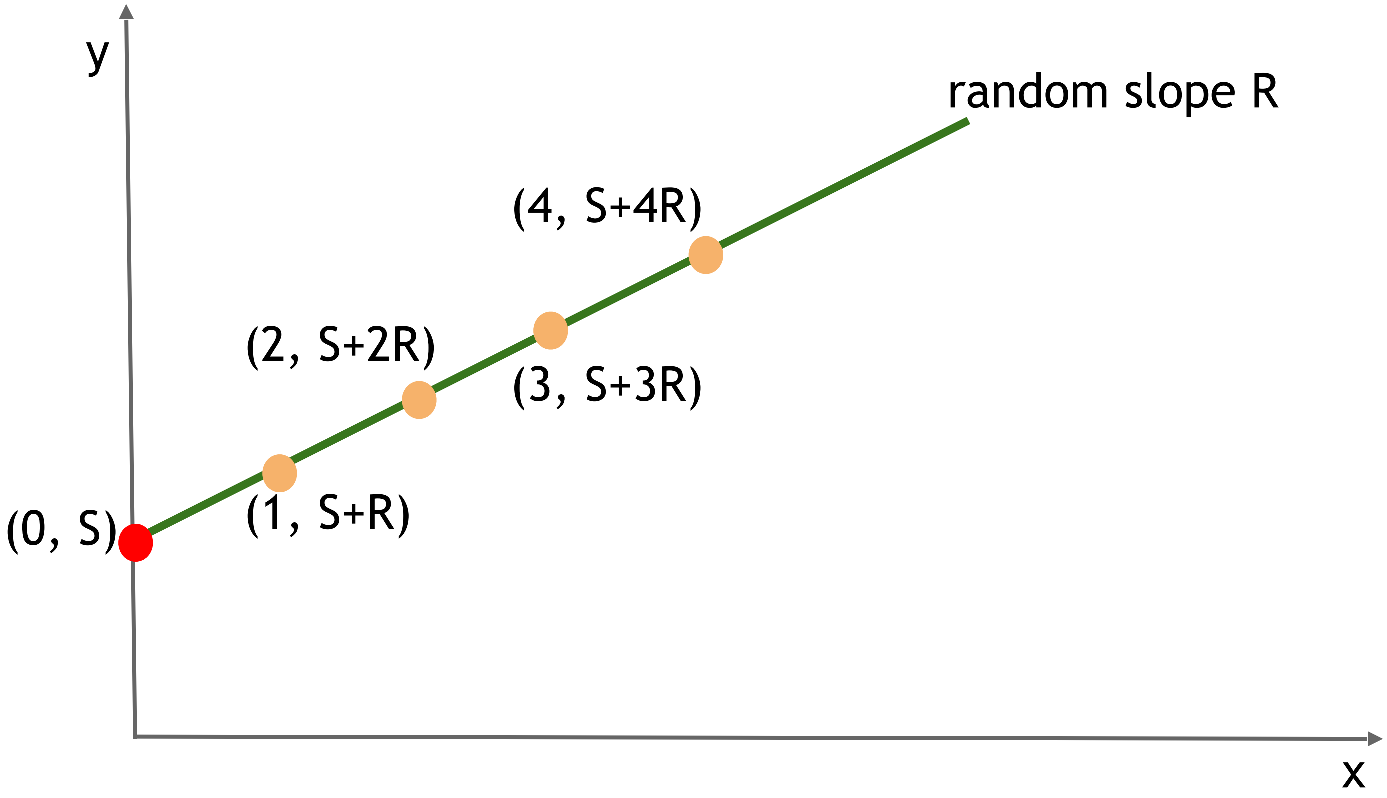
Example: k = 2, n > 2



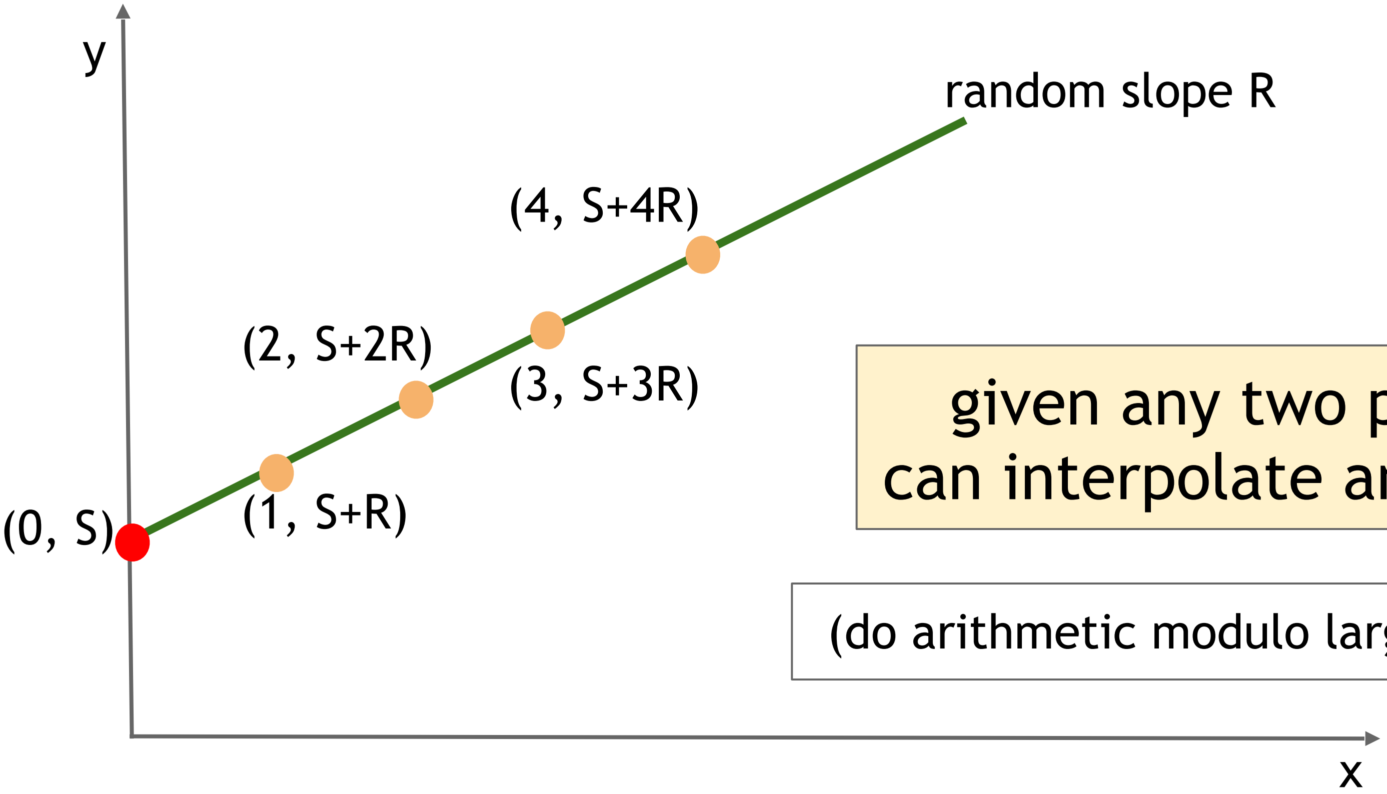
Example: k = 2, n > 2



Example: k = 2, n > 2



Example: k = 2, n > 2

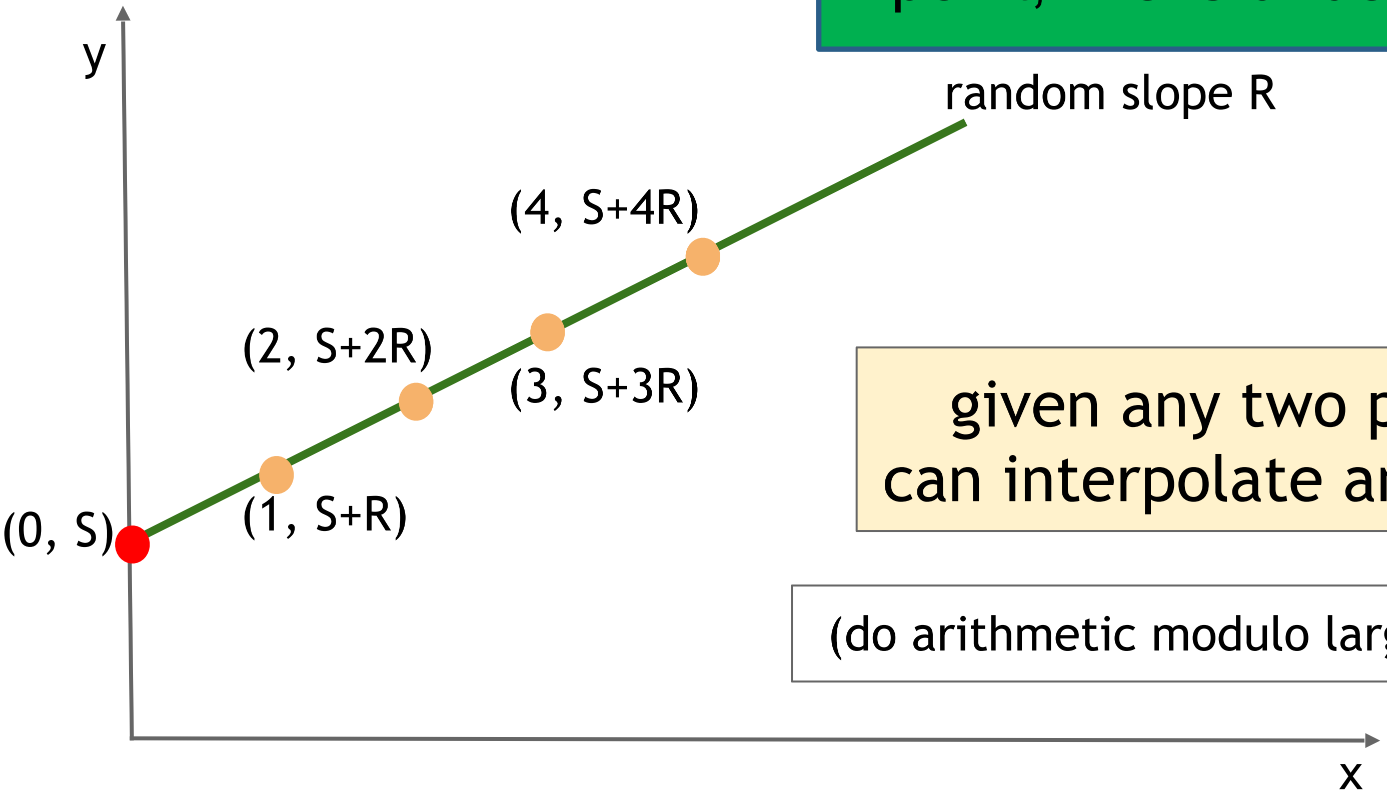


given any two points, can interpolate and find S

(do arithmetic modulo large prime p)

Example: k = 2, n > 2

k-Privacy: Given only one point, line is undertemined



given any two points, can interpolate and find S

(do arithmetic modulo large prime p)

Going Beyond k = 2

Going Beyond k = 2

Equation	Random parameters	Points needed to recover S
$(S + RX) \bmod p$	R	2
$(S + R_1X + R_2X^2) \bmod p$	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) \bmod p$	R_1, R_2, R_3	4

Going Beyond k = 2

Equation	Random parameters	Points needed to recover S
$(S + RX) \bmod p$	R	2
$(S + R_1X + R_2X^2) \bmod p$	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) \bmod p$	R_1, R_2, R_3	4

etc.

Going Beyond k = 2

Equation	Random parameters	Points needed to recover S
(S + RX) mod p	R	2
(S + R_1X + R_2X^2) mod p	R_1, R_2	3
(S + R_1X + R_2X^2 + R_3X^3) mod p	R_1, R_2, R_3	4

etc.

support K-out-of-N sharing, for any K, N

Secret sharing

Secret sharing

- **Good**: Store shares separately, adversary must compromise several shares to get the key.

- **Bad**: To sign, need to bring shares together, to first reconstruct the key. Point of vulnerability

Multi-sig

Recall multi-sig from previous lecture.

Lets you keep shares apart, approve transaction without reconstructing key at any point.

Example

- Alice, Bob, Charlie, and David are co-founders of a company.

- Each of the four generates a key-pair, puts secret key in a safe, private, offline place.

- The company’s cold-stored coins use multi-sig, so that three of the four keys must sign to release a coin.

Threshold Signatures

- **(k,n)-Threshold Signatures:** A signing key can be "divided" amongst n signers such that any subset of k signers can jointly produce a signature, but any subset of <k signers cannot
  - TSetup(1^n): Each party learns PK. Party i additionally learns Sk_i
  - TSign(m): Parties run a protocol to compute a signature **sig** on m
  - TVerify(PK,m,**sig**): Output 0/1

Threshold Signatures

- Advantages over Multi-Sig:
  - Threshold policy enforced in signature scheme as opposed to script
  - Threshold signature size can potentially be same as a single signature (as opposed to increasing linearly with k)
  - Threshold policy can be hidden in Threshold signatures
  - ----

Threshold Signatures

- (k,n)-Threshold Signatures for EC-DSA: Many solutions known, see e.g., [Gennaro-Goldefeder-Narayanan’16]

- However, all known solutions require interactive signing protocol

- Question: Threshold Signatures with 1-round signing process?

Online Wallets and Exchanges

Online wallet

like a local wallet

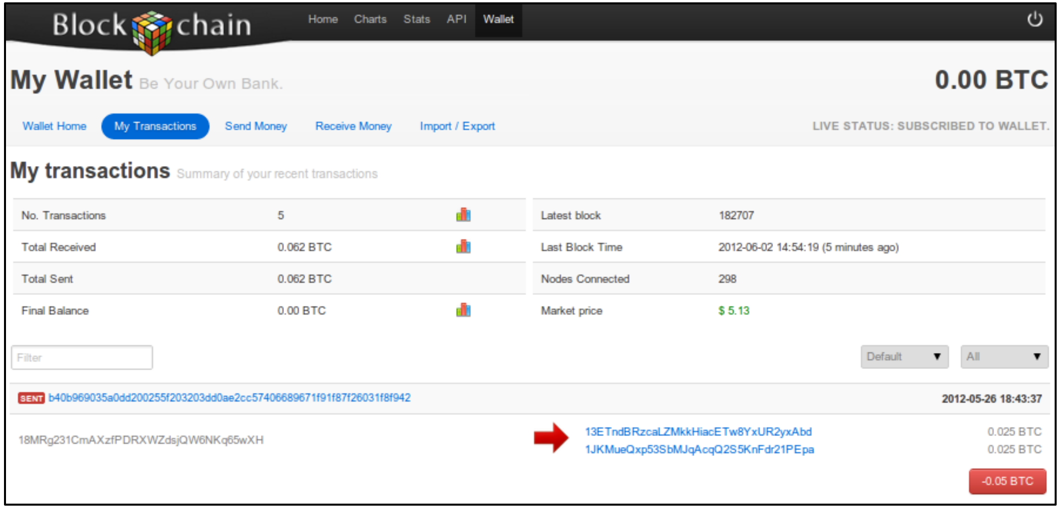
but “in the cloud”

runs in your browser

site sends code

site stores keys

you log in to access wallet



My Wallet Be Your Own Bank. 0.00 BTC

LIVE STATUS: SUBSCRIBED TO WALLET.

My transactions Summary of your recent transactions

No. Transactions	5	Latest block	182707
Total Received	0.062 BTC	Last Block Time	2012-06-02 14:54:19 (5 minutes ago)
Total Sent	0.062 BTC	Nodes Connected	298
Final Balance	0.00 BTC	Market price	$ 5.13

SENT b40b969035a0dd200255f203203dd0ae2cc5740668967 1f91f87f26031f8f942 2012-05-26 18:43:37

18MRg231CmAXzfPDRXWZdsjQW6NKq65wXH → 13ETndBRzcaLZMkkHiacETw8YxUR2yxAbd 0.025 BTC
1JKMueQxp53SbMJqAcqQ2S5KnFdr21PEpa 0.025 BTC

-0.05 BTC

Online wallet tradeoffs

- convenient: nothing to install, works on multiple devices
- but security worries: vulnerable if site is malicious or compromised
- ideally, site is run by security professionals

Bank-like services

- you give the bank money (a “deposit”)
- bank promises to pay you back later, on demand

- bank doesn’t actually keep your money in the back room
    - typically, bank invests the money
    - keeps some around to meet withdrawals (“fractional reserve”)

Bitcoin Exchanges

accept deposits of Bitcoins and fiat currency ($, €, …)

promise to pay back on demand

lets customers:

make and receive Bitcoin payments

buy/sell Bitcoins for fiat currency

typically, match up BTC buyer with BTC seller

What happens when you buy BTC

suppose my account at Exchange holds $40000 + 3 BTC
I use Exchange to buy 2 BTC for $9000 each

result: my account holds $22000 + 5 BTC

note: no BTC transaction appears on the blockchain
only effect: Exchange is making a different promise now

Exchanges: Pros and Cons

pro: connects BTC economy to fiat currency economy

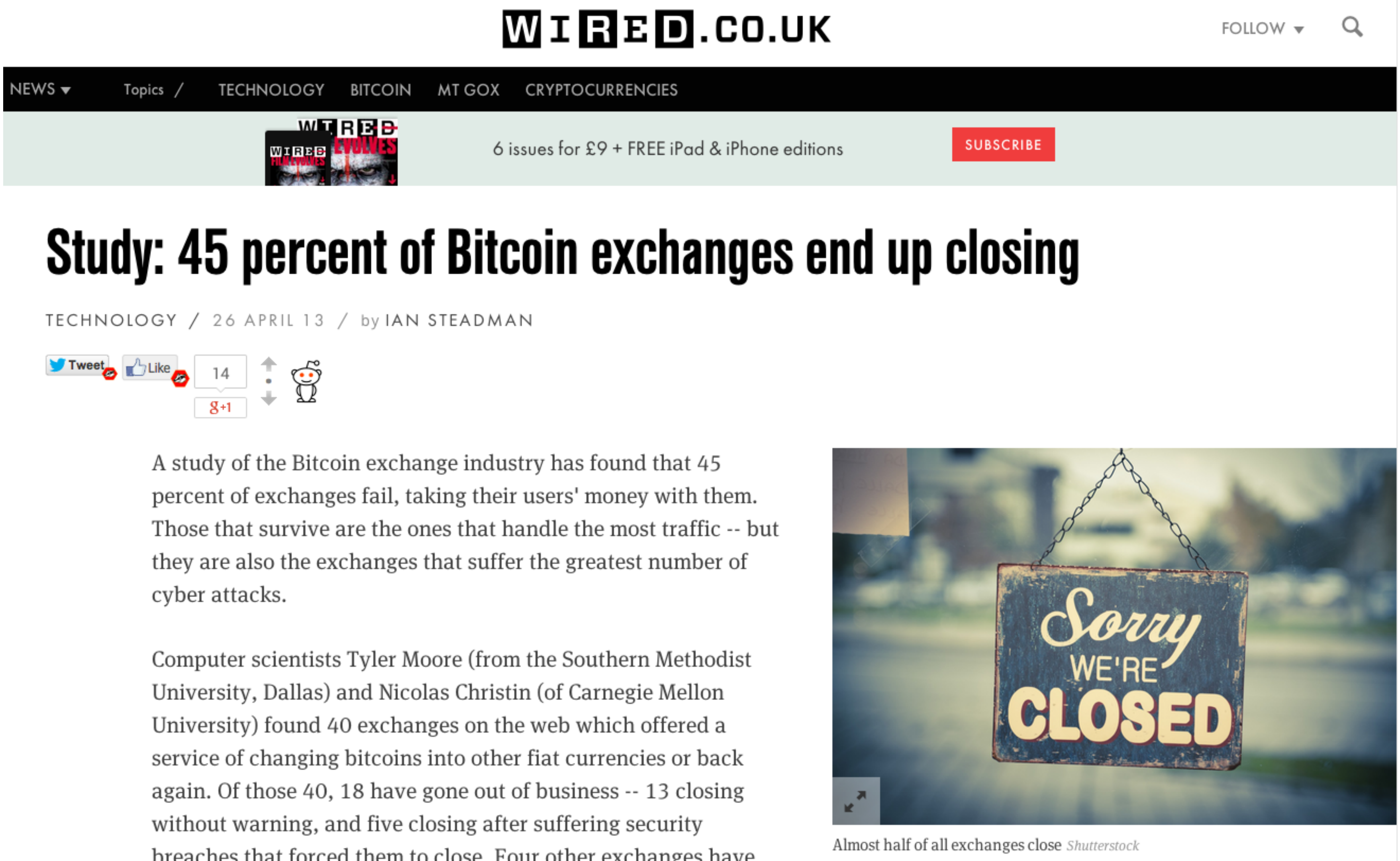
easy to transfer value back and forth

con: risk

same kinds of risks as banks

BROS

Charles Ponzi



Study: 45 percent of Bitcoin exchanges end up closing

TECHNOLOGY / 26 APRIL 13 / by IAN STEADMAN

A study of the Bitcoin exchange industry has found that 45 percent of exchanges fail, taking their users' money with them. Those that survive are the ones that handle the most traffic -- but they are also the exchanges that suffer the greatest number of cyber attacks.

Almost half of all exchanges close *Shutterstock*

Computer scientists Tyler Moore (from the Southern Methodist University, Dallas) and Nicolas Christin (of Carnegie Mellon University) found 40 exchanges on the web which offered a service of changing bitcoins into other fiat currencies or back again. Of those 40, 18 have gone out of business -- 13 closing without warning, and five closing after suffering security breaches that forced them to close. Four other exchanges have

東京でMT.GOXのデモ
へ参加してください。
東京都渋谷区渋谷
2丁目１１−５
MTGOX
WHERE IS
OUR MONEY

Bank Regulation

for traditional banks, government typically:

- imposes minimum reserve requirements
  - must hold some fraction of deposits in reserve
- regulates behavior, investments
- insures depositors against losses
- acts as lender of last resort

Proof of Reserve

Bitcoin exchange can prove it has fractional reserve.

fraction can be 100%

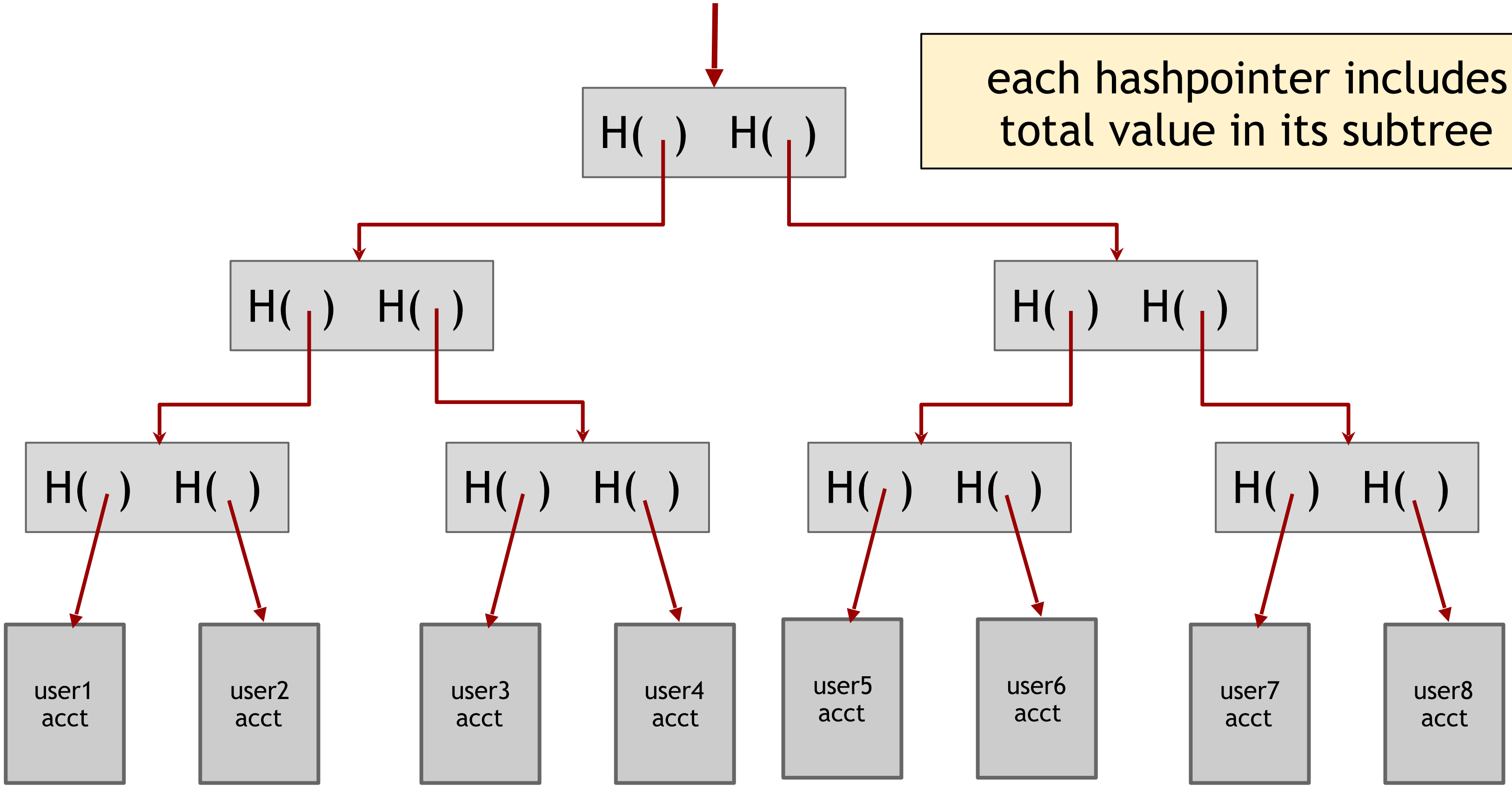
Prove how much reserve you're holding:

publish valid payment-to-self of that amount

sign a challenge string with the same private key

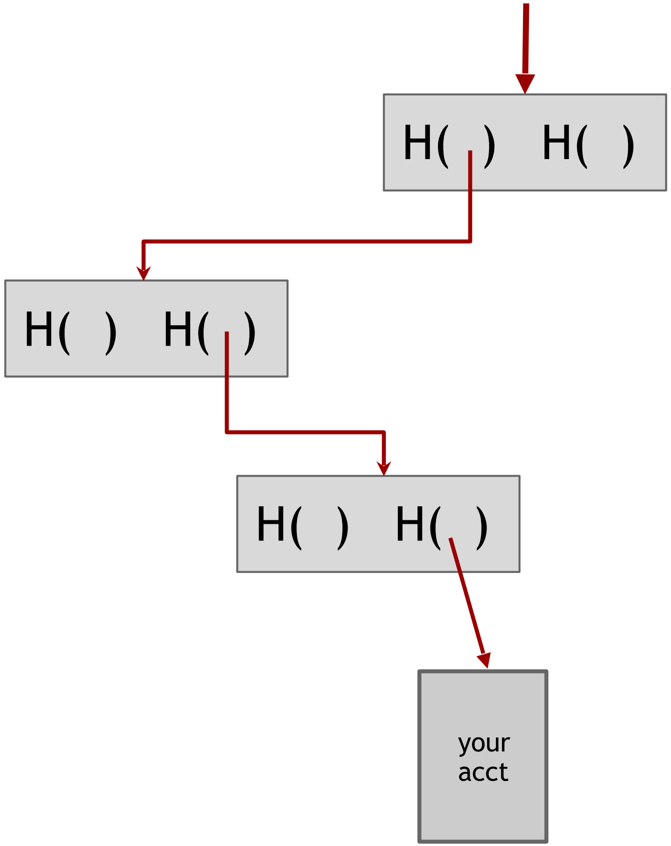
Prove how many demand deposits you hold: ...

Merkle tree with subtree totals



each hashpointer includes total value in its subtree

Checking that you're represented in the tree



show O(log n) items

Proof of Reserve

Prove that you have at least X amount of reserve currency

Prove that customers have at most Y amount deposited

So reserve fraction ≥ X / Y