CS1699: Blockchain Technology and Cryptocurrency

10. Using Bitcoin

Bill Laboon

Using Bitcoin

- *Owning bitcoin* really means:
  - "I have the ability to control any UTXOs sent to some set of addresses (loosely, public keys) and generate new transactions from them…
  - … which I can prove by signing said transactions with the corresponding secret keys for each address."

Public Keys != Addresses

- Although a public key can be a kind of public address (as in StringCoin), in Bitcoin it is slightly more complex
- Your address is *A(pk)*, where *A()* is a rather convoluted multi-hashing process (see next slide)
- Thus, P2PKH ("pay-to-public-key-hash") as opposed to P2PK ("pay-to-public-key")
- Addresses are generally represented in *Base58Check* format

Base58Check Encoding

- Just like binary (base 2) has 2 possible values (01), octal (base 8) has 8 (01234567), hexadecimal (base 16) has 16 (0123456789ABCDEF), Base58 has 58 (123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz)
- Note that 0 (capital O), 0 (zero) , l (lowercase L), and I (capital i) to avoid confusion
    - Thus, the “1” at the beginning of standard Bitcoin addresses actually means the value “0”!
- Base58Check also includes a four-byte checksum at the end to ensure accurate transcription

A Number Base58Check is still a NUMBER

- Can represent as a QR code (2-d barcode)

SHARE

1MkDdcojPfYcWkQLaZhhuKo24sBRdxPRhJ

- Or decimal:
  5579715324429201893029744493272740905093118947118065324001,
- Or binary:
  1110001110001110111100100011011000100100100010010100000000011111110101000000000
  0011100111011001011101111010101110110001110011100110110111101010110001011101000
  110000011001001011000011111100001

Bitcoin Address Generation From a Public Key

1. Generate an ECDSA keypair, and take its public key
2. Take the SHA-256 hash of the public key.
3. Take the RIPEMD-160 hash of the SHA-256 hash generated in the step above.
4. Prepend a “0” (i.e. “1” :)) to indicate this is a Bitcoin Mainnet address.
5. *Take the SHA-256 hash of the prepended RIPEMD-160 hash.*
6. *Take the SHA-256 hash of the previous SHA-256 hash.*
7. *Take the first four bytes of that last SHA-256 hash and store it as a checksum.*
8. Append the four-byte checksum to the end of the prepended address found in Step 4.
9. Convert the result from Step 8 to Base58Check encoding (see next slide).

Steps in italics represent checksum calculations.

What If I Want A Specific Address?

- 1CS1699xxxxxxxxxxxxxxxxxxxxxxxxxxxxx is a valid address, I want it!
- Recall that generating an address means taking a randomly generated public key, prepending it with 1, then hashing it (twice, with two different hash algorithms) then adding a checksum
- Ergo, generating an address equal to a value or matching some pattern can be seen as a kind of Bernoulli trial

Vanity Address Generation

- Just like mining, we can just keep trying with different inputs
- For each character we want, there is a 1/58 chance we will get the correct one, ergo chances are $1 / 58^k$ for a k-character pattern
- Luckily, programs can do this for us… do this locally! Keys have been stolen from "online vanity generators!"
- Also note that there are some possible efficiency optimizations over sheer random guesses, see book for details
- https://github.com/samr7/vanitygen

Vanity Address Generation

```
$ time ./vanitygen 1C
Address: 1CXakjLjyXCb54nY78C9dzZtTbuLsSv2sT
Privkey: 5Kk7QVaupb8YDLHDJmrfCnZ582kPsCCQLWKZFVXGuxVbY61cXwz
real 0m0.016s

$ time ./vanitygen 1CS
Address: 1CSsTLkrKDziit54WDemuagM5jbqALhims
Privkey: 5KLvhtRKHkRLP2rf5dMoTBrWmJRVoM78p773cbr6of3ZsHR8Ljp
real 0m0.016s

$ time ./vanitygen 1CS1
Address: 1CS1Koo5r9jqMBsMaNjazVpSzXuUvdqdG7
Privkey: 5JMP4B5LrkeKe2WaJpPt1xn7vqRg1r1WBKt9TxHSK2KkLc2Jubc
real 0m0.526s

$ time ./vanitygen 1CS16
Address: 1CS16y6VQSTzoK8eGGeBwQGW9rCTgecV28
Privkey: 5KFeGex4kwq1hJ5psh3fBW5Yn3kbSqsi4aRq3iehBoy9G97S8qk
real 0m23.411s

$ time ./vanitygen 1CS169
Address: 1CS169cXfqKyTfJR1jLY6mGwQMBtWDfbx8
Privkey: 5KLyxp1qZvpTjV9mZsQF4117wPcuFUx3UGcY7xn7aTEUwtYcZLj
real 10m40.580s
```

How Do I Store My Bitcoin?

- Storing bitcoin is ALL about managining your secret (and to a lesser extent, your public) keys
- But there are a variety of ways to do so, with various trade-offs, mainly:
  - Availability
  - Security
  - Convenience

Hot vs. Cold Storage

- **Hot storage**: Bitcoin can directly be spent on the Bitcoin network (i.e. node or node-proxy is online)
- **Cold storage**: Bitcoin cannot directly be spent on the Bitcoin network (i.e. secret keys are offline)
  - Note: Cold storage retains the possibility of sending Bitcoin to it even if offline, however, as long as a valid public key is known

Wallet Software (Hot)

- Software on your computer or phone which keeps track of your keys, coins, makes transactions, etc.
- Usually a SPV node or connected to a SPV node
- Think of it as the wallet in your pocket - simple to use and understand, but pickpockets are rife!
- Availability: HIGH, Convenience: HIGH, Security: LOW

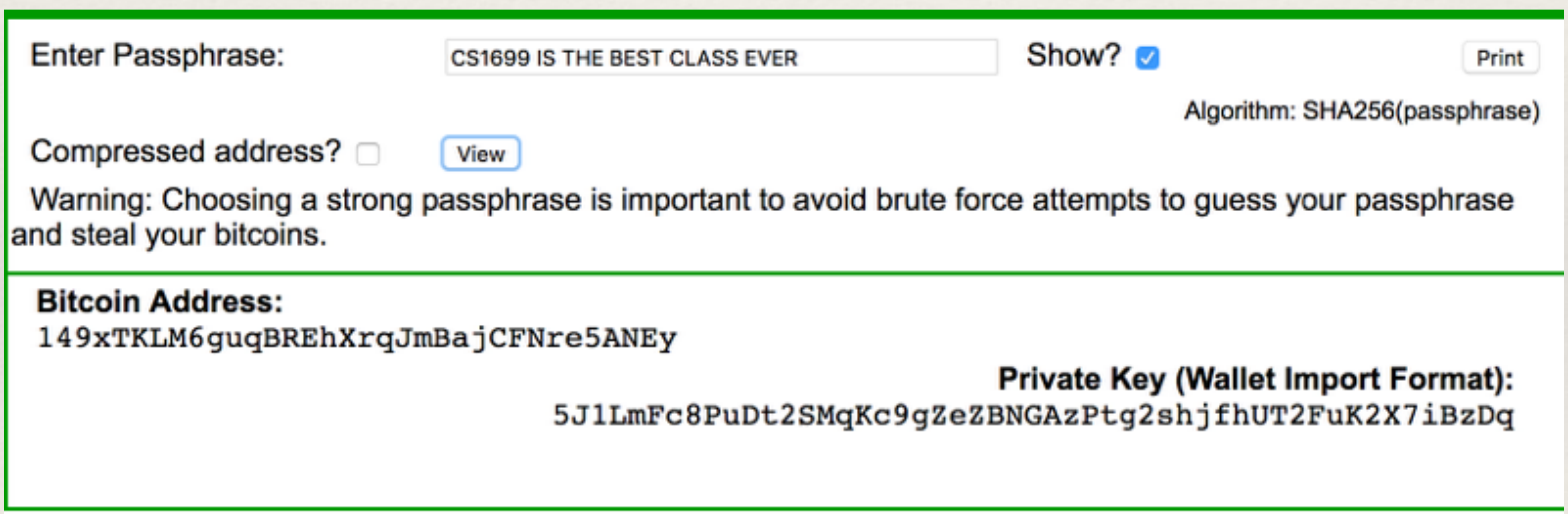
Paper Wallet (Cold)

- A generated address / secret key which is printed out and kept on piece of paper

- Availability: LOW, Convenience: MEDIUM, Security: MEDIUM / HIGH

Brain Wallet (Cold)

- Remember a pass phrase and use the hash of it as a seed to pseudorandomly generate a keypair



- Availability: LOW, Convenience: LOW, Security: HIGH

Hardware Wallet (hybrid hot/cold)

- Simple devices (not full computers) which generate keys but secret keys never leave device
- Transactions are fed in, signed, and output
- Examples: Trezor or Nano
- Availability: MEDIUM, Convenience: MEDIUM, Security: HIGH

Why Address Re-Use Is A Bad Idea

- Leaks information about your identity
- Assume you have been sending transactions all of your life to a single address. If anyone links that address to you, they know you have several hundred thousand dollars in Bitcoin.
- Assume you pay your landlord in Bitcoin from the same address every month and get paid from your employer to the same address. Now your landlord knows you can afford a raise in rent.
- A Bitcoin wallet can generate a large (in practical terms, infinite) number of addresses, so let's use them

Hierarchical Deterministic Wallets

- Instead of a wallet specifically generating an address, it generates a way to create a series of unlinked addresses, but to which it has all of the secret keys
- Now you can have as many addresses as you like, but "see" them all in your wallet as a single Bitcoin "address"
  - Recall that tx inputs can come from multiple addresses!
- See BIP 32 https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

Splitting and Sharing Keys

- Assume my wife and I want veto power over spending from our joint Bitcoin account - i.e., we both need to agree that Bill buying a Bitcoin sweater is a good use of our funds, and so we want to only be able to reconstruct our key if both of us agree
- How can I do this?

Naive Splitting

- Public address is:149xTKLM6guqBREhXrqJmBajCFNre5ANEy
- Secret key is: 5J1LmFc8PuDt2SMqKc9gZeZBNGAzPtg2shjfhUT2FuK2X7iBzDq
- I know the red part of the secret key, she knows the blue part - we have to put them together in order to actually sign anything (and thus spend our bitcoin)
- Several problems with this scenario!

Secret Sharing - K of N

- Idea: Given some values K and N and $K <= N$, can we split the key into N pieces such that if we have K shares, we can determine the key…
- … but having any number of shares n (where $n > 0$ and $n < K$), we will not gain any knowledge of the key?

Secret Sharing

- Suppose $N = 2$, $K = 2$, and our secret key S is 128 bits.
- Generate a random 128-bit value R.
- Two shares: R and $(S \text{ XOR } R)$
- Together, can calculate S; otherwise, basically random numbers
- Can extrapolate out to any $N >= 2$ and $K >= 2$ BUT N must equal K.

What If We Want K != N?

- Example: I want to bequeath my bitcoin to my son upon my death. I want to split my key into three, so that 2/3 of my lawyer, me, and my wife can determine my key.
- Or extrapolate this out further: 2 out of 6 board members must agree to spend money from a corporation's bitcoin account, or 5 out of 9 justices on a court must agree to pay bitcoin held in escrow

2-of-n Secret Sharing

Secret key = 5

$Share_1$ = 3, 8

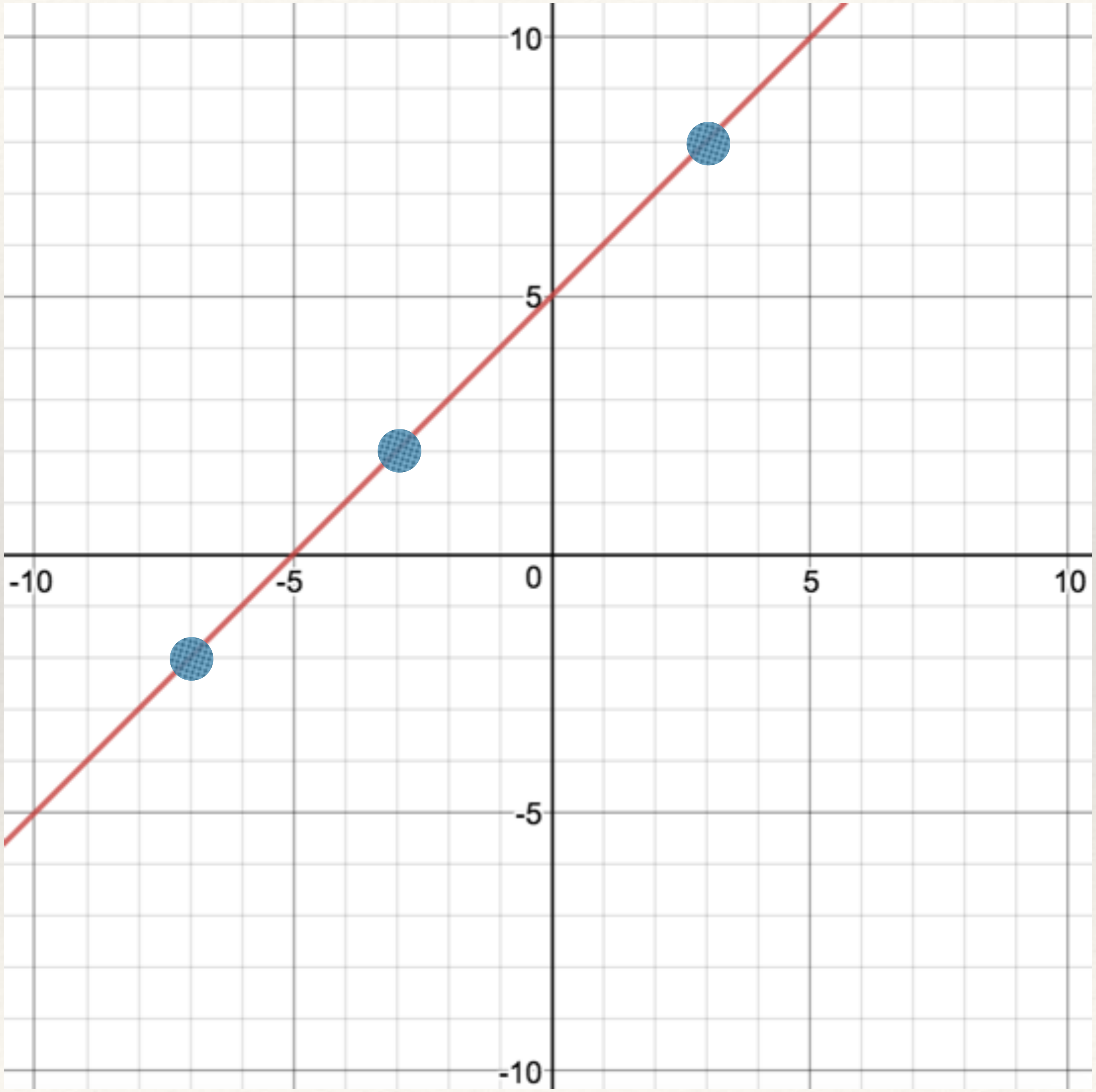
$Share_2$ = -3, 2

$Share_3$ = -7, -2

$Share_n = \dots$

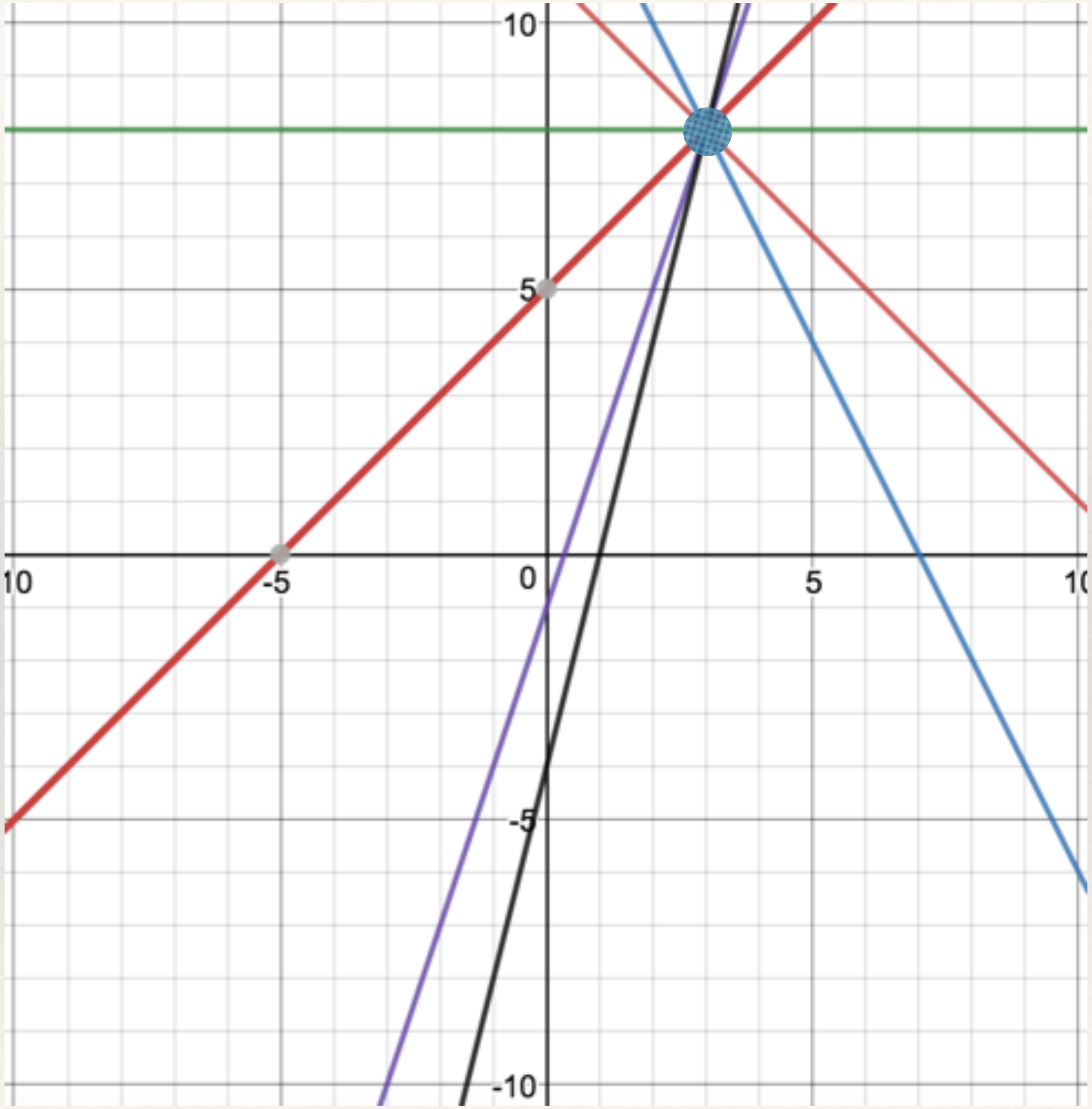
Slope: y = x + 5

Can calculate slope - and thus intercept - iff I have 2 or more points on the line



2-of-n Secret Sharing, One Share

With one share (point) - e.g. 3, 8 -
I cannot determine the slope…
thus cannot determine the intercept…
thus cannot determine the key.



3-of-n Secret Sharing

Secret key = 4

$Share_1$ = -4, 20

$Share_2$ = 1, 5

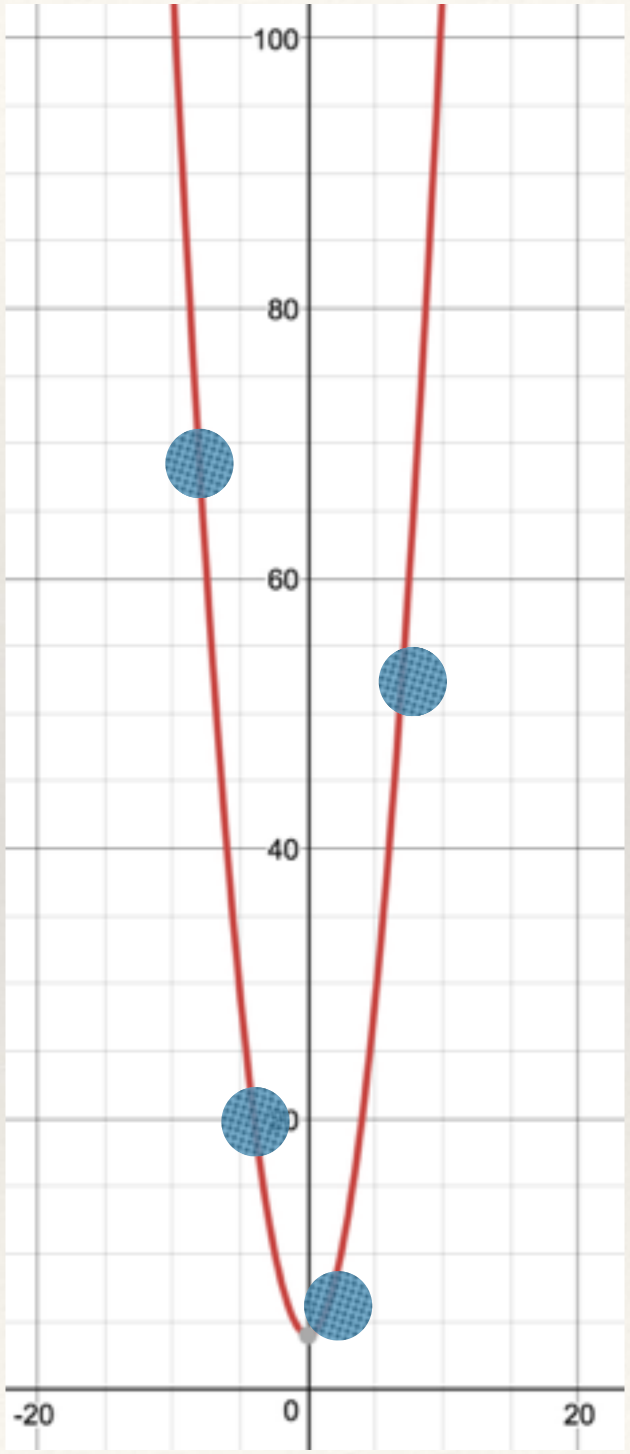
$Share_3$ = 7, 53

$Share_4$ = -8, 68

$Share_n = \dots$

Source Function: y = x + 5

Can recreate function -
and thus determine intercept -
iff I have 3 or more
points on the parabola.



Arbitrary k-of-n Secret Sharing

- Determine *key* (randomly)
- Determine *k*, and randomly generate a *k*-order polynomial which intersects origin at *0, key*
- Determine *n* random points on the function and distribute
- Can expand indefinitely via Lagrange interpolation

Threshold Signatures

- Problem with previous schemes - they all generate the actual secret key, which is a point of weakness (key can be stolen after reconstruction and used on its own)
- There is a (mathematically complex) way to make partial signatures and if enough actors generate partial signatures, they can sign a transaction with a key without ever revealing the key or even their key shares
- Avoids the single point of failure problem above

Online Wallet

- Software runs online where you have an account
- You will either need to enter key or they will store it in an encrypted format
- Very convenient! No installation on your part.
- If you don't have the keys, though, or they are grabbing keys that you pass in, big security worry. NOT YOUR KEYS, NOT YOUR COINS!
- https://login.blockchain.com/#/signup

Online Exchange

- Online wallet plus place to buy/sell bitcoin (like a stock exchange… I'll give you $9.25 for one share of Ford, or I'll give you $6,595.76 for one bitcoin)
- *Fractional reserve* - Just like real banks, exchanges promise to give you your bitcoin when you ask for it, but may not actually have it!
- Very convenient, offloads responsibility, but there are risks with trusting any bank

Risk 1: Bank/Exchange Runs

- Many people ask for their money back at a bank; bank does not have enough cash on hand
- Rumors spread that bank is insolvent
- More people try to cash out, exacerbating problem
- In US, bank accounts are backed by FDIC so this *shouldn't* happen (since the FDIC can always print more money… can't print more bitcoins!)

Risk 2: Counterparty Risk

- Bank/exchange is not really planning on giving people the bitcoin that is rightfully theirs
- "Exit scam"
- Ponzi scheme: paying people who ask for bitcoin with newcomer's bitcoin

Risk 3: Security Breach

- A hacker could break into the exchange’s software and move all of their bitcoin
- There is no “arbiter” to determine who “rightly” owns some bitcoin
- “Why do you rob banks? ‘Cause that’s where the money is.” -supposedly said by Willie Sutton

Some Possible Systemic Bitcoin Risks

- Sybil attack via peers (i.e., flood network with "bad" peers, new connections may see only bad peers)
- Adding illegal content to blockchain (regulatory risk)
- Major weakness in SHA-256, RIPEMD-160, ECDSA, or other cryptographic algorithm
- Quantum attacks - quantum computers can factor large numbers very quickly
- Security vulnerabilities / defects
- Timejacking
- Denial of service attacks
- Mining death spiral - no longer profitable to mine, so fewer miners, so less interest in bitcoin, so lower price, so even less profitable to mine, *ad infinitum*