Distributed Systems

Spring Semester 2020

Lecture 28: Bitcoin

John Liagouris
liagos@bu.edu

Bitcoin

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

Credit Cards

Pro

- Works Online
- *Hard* to Steel
- Can Cancel Transactions
- Tied to currency controlled by government

Con

- Can Cancel Transactions
- Tied to currency controlled by government
- 3% Fee
- Long Settling Time
- Hard to become a merchant

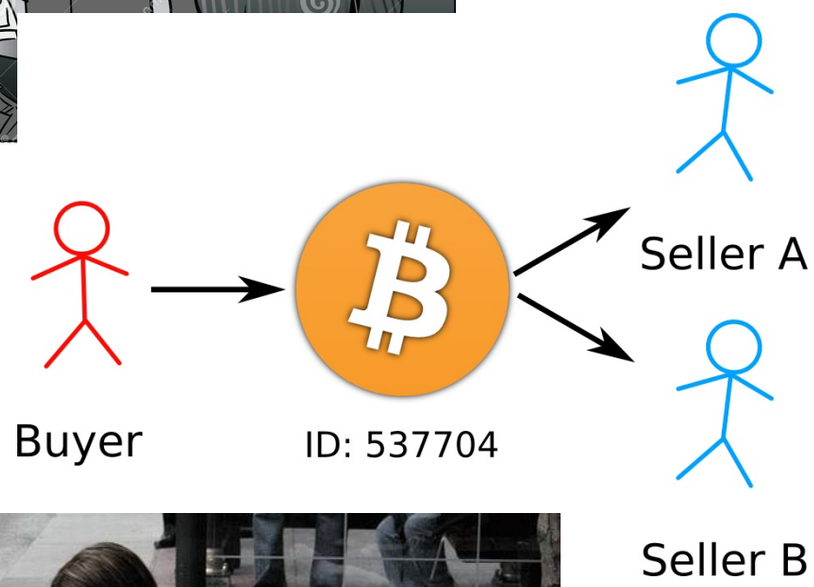
Bitcoin: e-money without a central trusted party

a **public ledger**: anyone can verify transactions

Technical Challenges

- Forgery — production and detection

- Double Spending — 2 for one



- Theft

Social/Economic Challenges

- Why does it have value?

- How to pay for infrastructure?

- Monetary Policy?

- Laws?

OneBit: Simple e$ System

GOAL: Illustrate a public, verifiable ledger using transaction chain

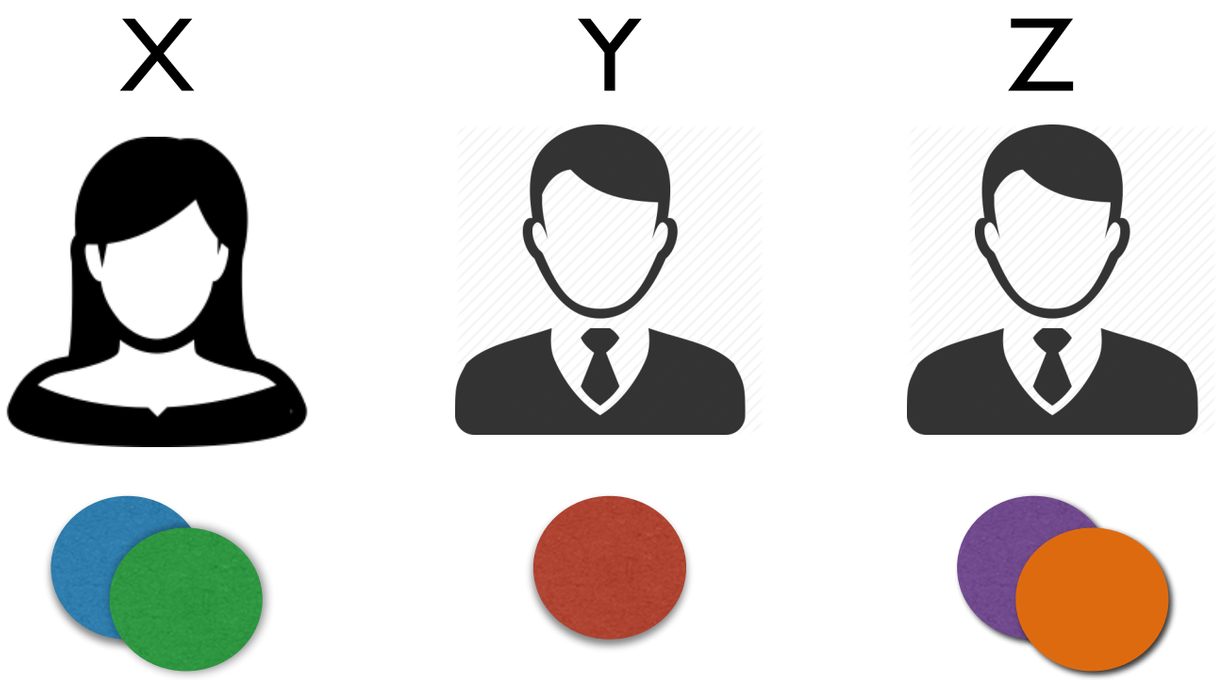
X

Y

Z

OneBit: Simple e$ System

GOAL: Illustrate a public, verifiable ledger using transaction chain



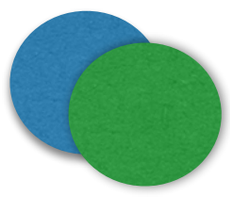
- Each user owns some coins

OneBit: Simple e$ System

GOAL: Illustrate a public, verifiable ledger using transaction chain

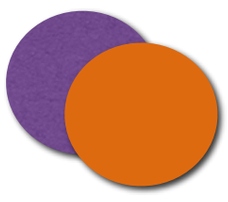
OneBank

X



Y

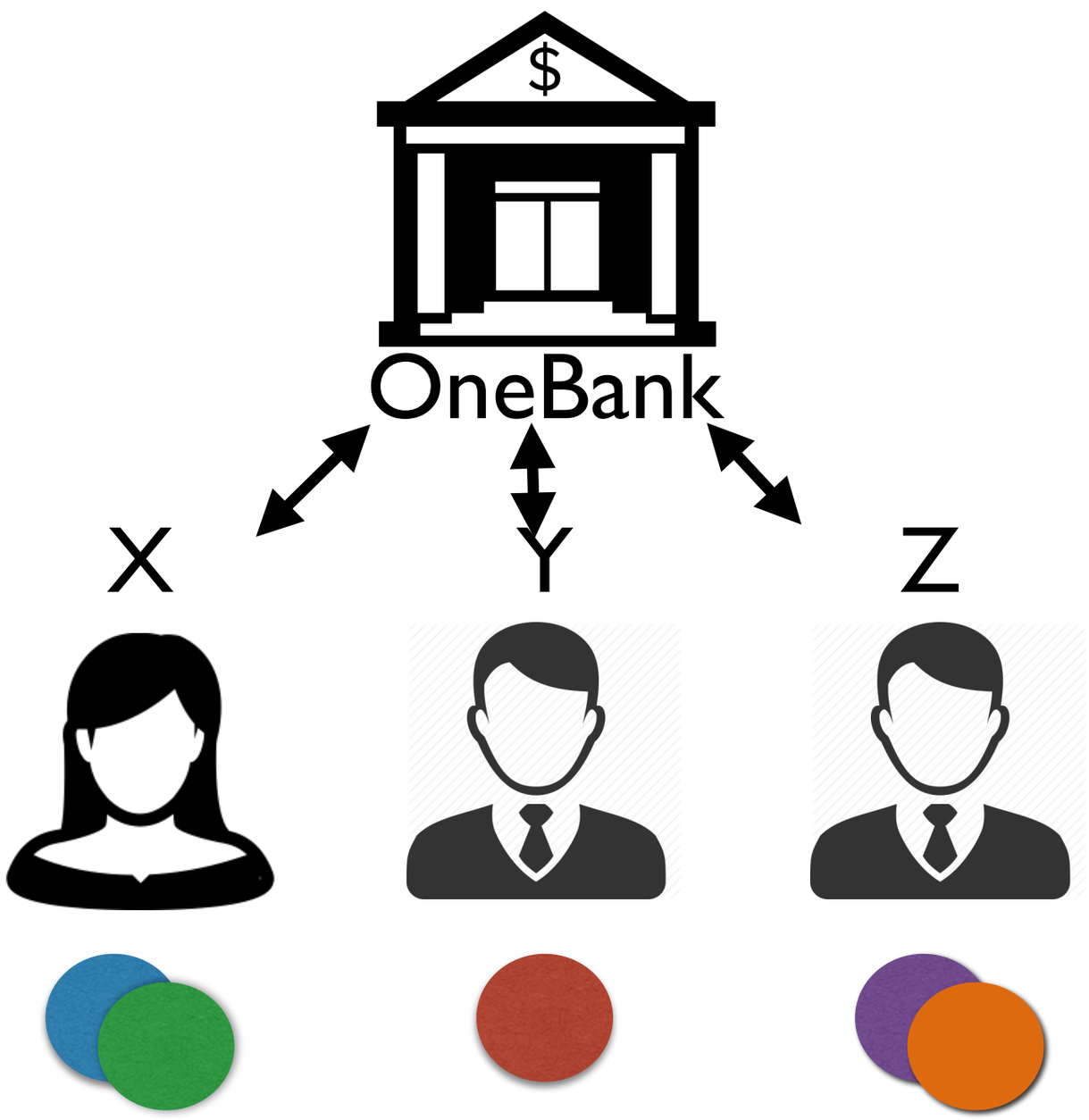
Z



- Each user owns some coins
- Single Server — OneBank
- Everyone talk to it
- OneBank records all transactions

OneBit: Simple e$ System

GOAL: Illustrate a public, verifiable ledger using transaction chain

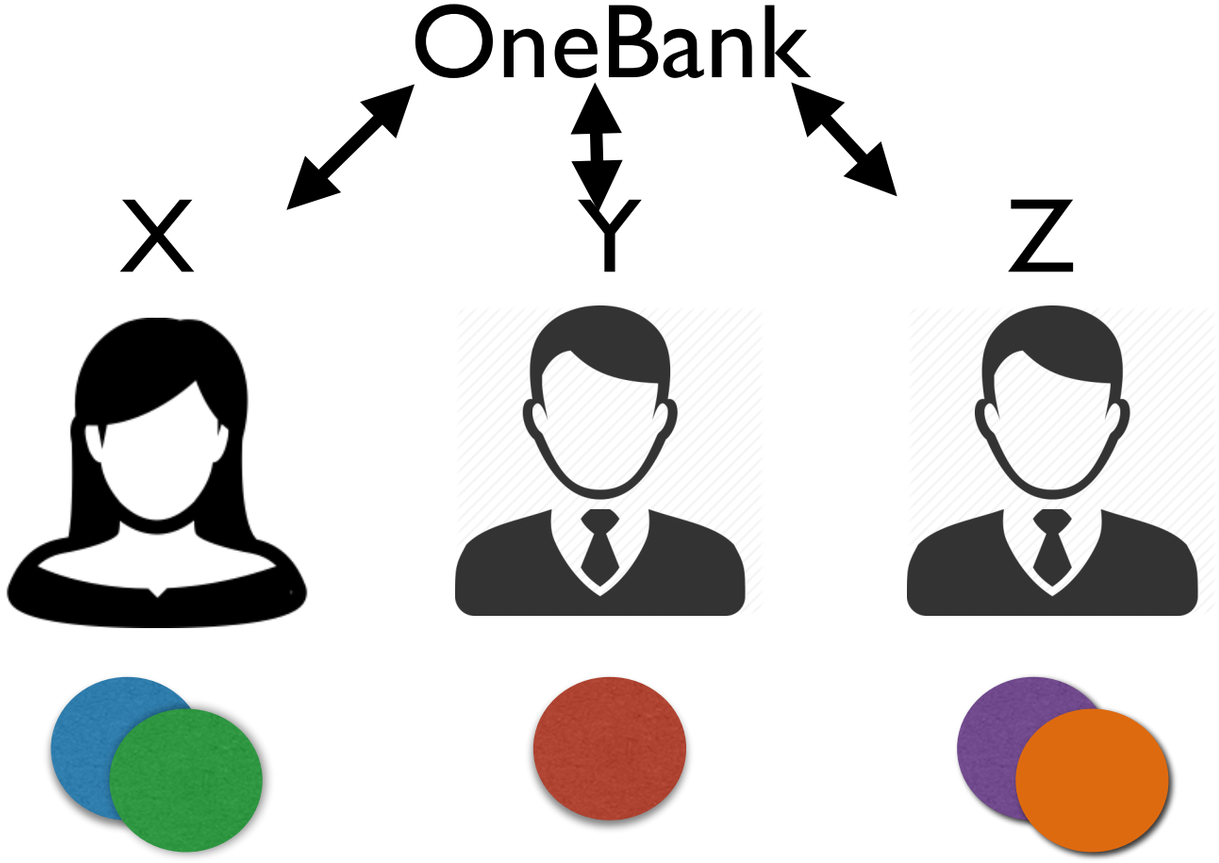


- Each user owns some coins
- Single Server — OneBank
- Everyone talk to it
- OneBank records all transactions

OneBit: Simple e$ System

The Ledger

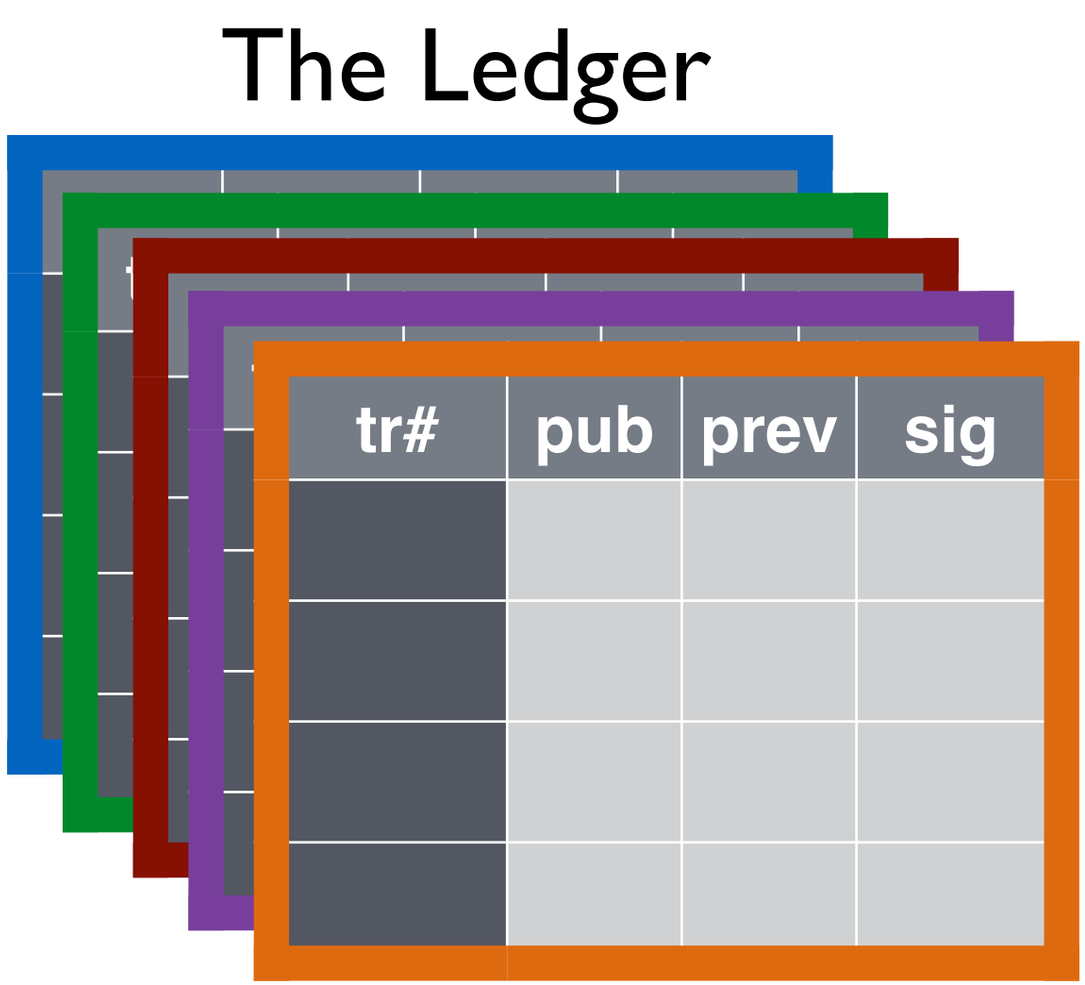
tr#	pub	prev	sig

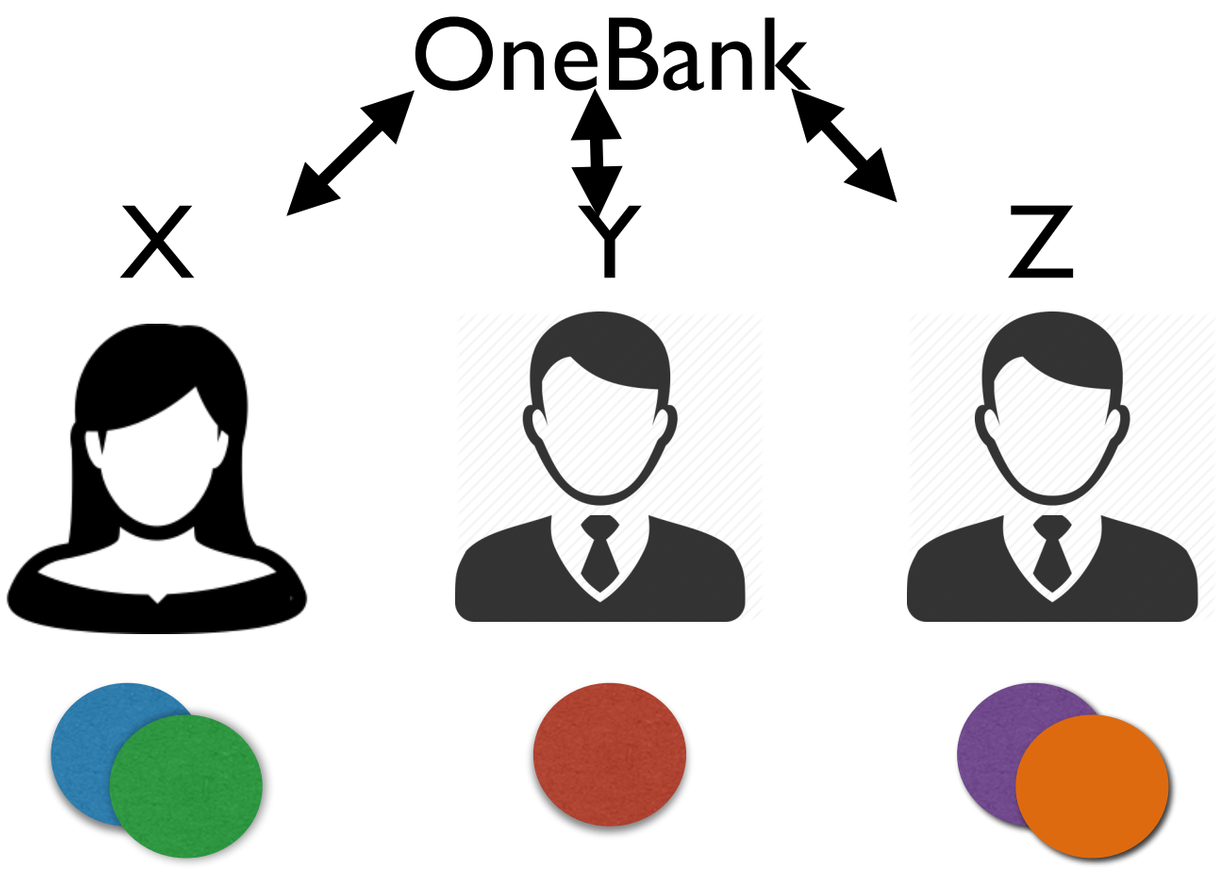


GOAL: Illustrate a public, verifiable ledger using transaction chain

- Each user owns some coins
- Single Server — OneBank
- Everyone talk to it
- OneBank records all transactions

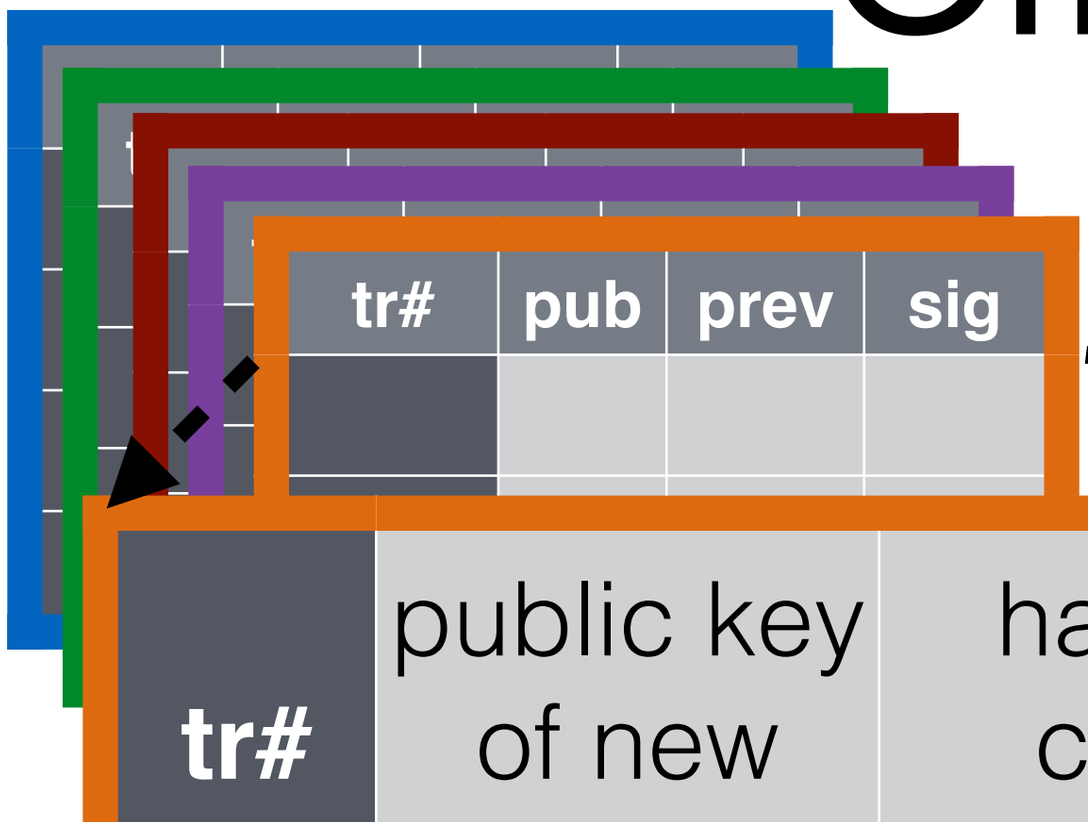
OneBit: Transactions



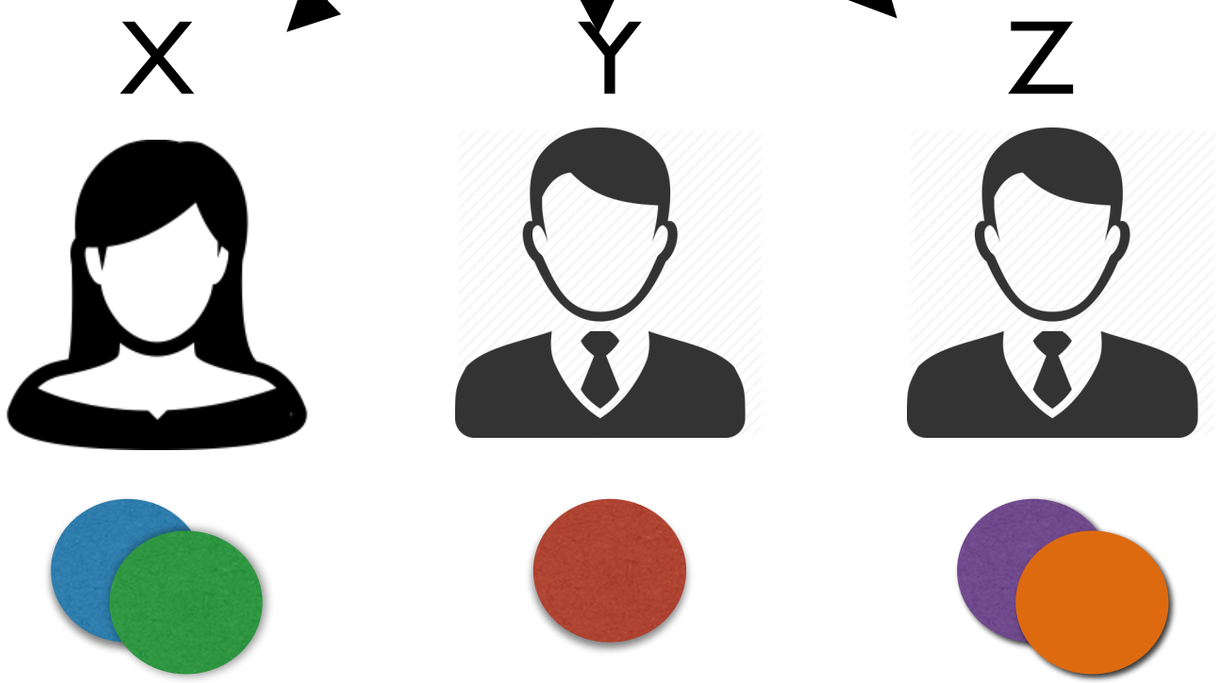


- Every Coin has a chain of transaction records
  - One record for each transfer of **the** coin
- OneBank maintains complete chain for each coin
- Chain ensures that only the current owner can spend the coins

OneBit: Transaction Record (TR)



tr#	public key of new owner	hash of coin's previous TR	Signed by Private Key of previous owner



Records link transfers into a single serial chain

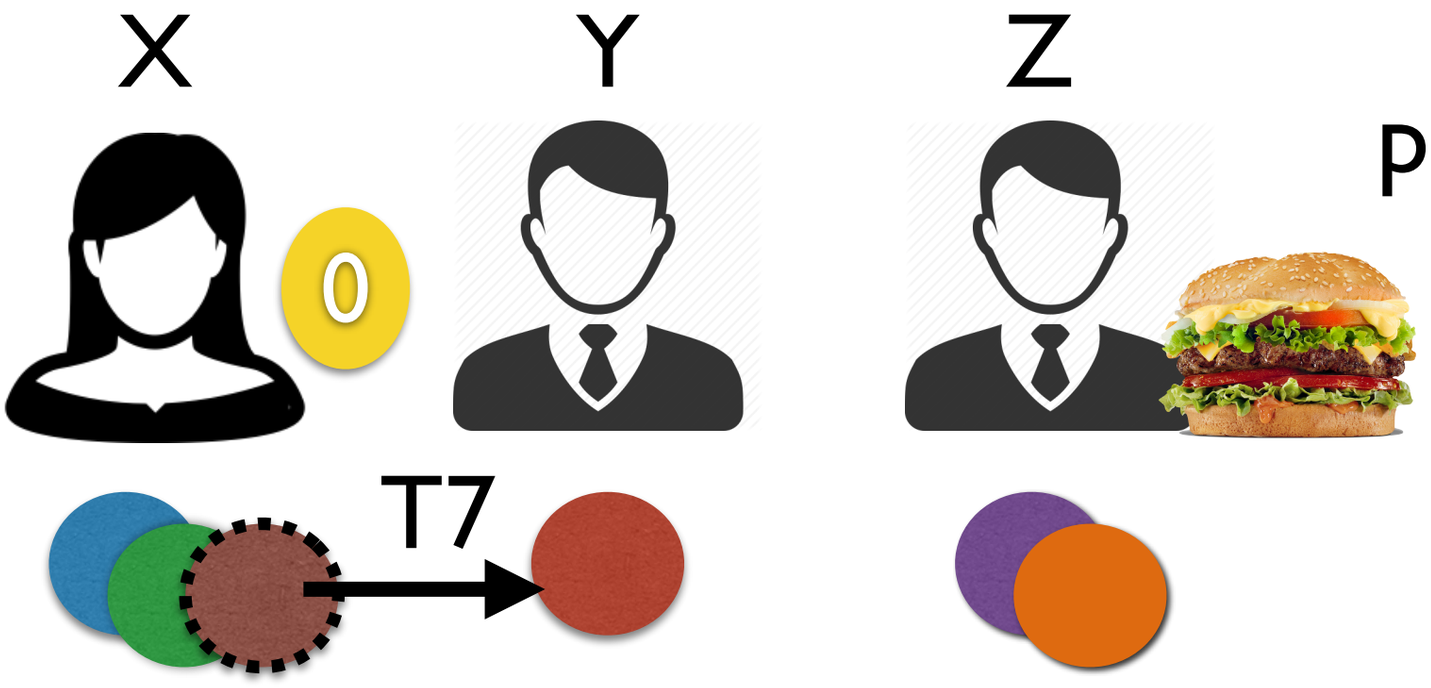
Crypto makes it secure (assuming private keys stay private)

OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)



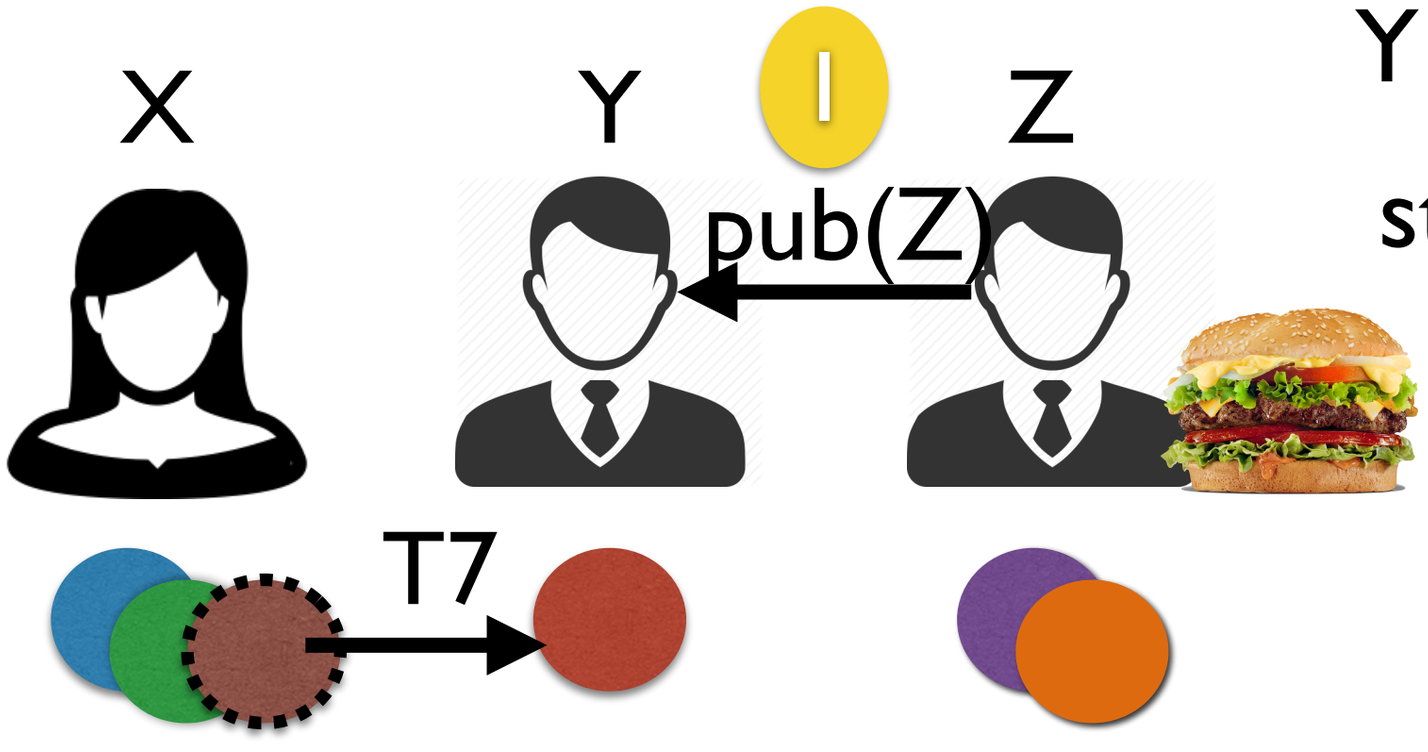
step 0: Y owns a coin, previously given to it by X

OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)



Y buys hamburger from Z

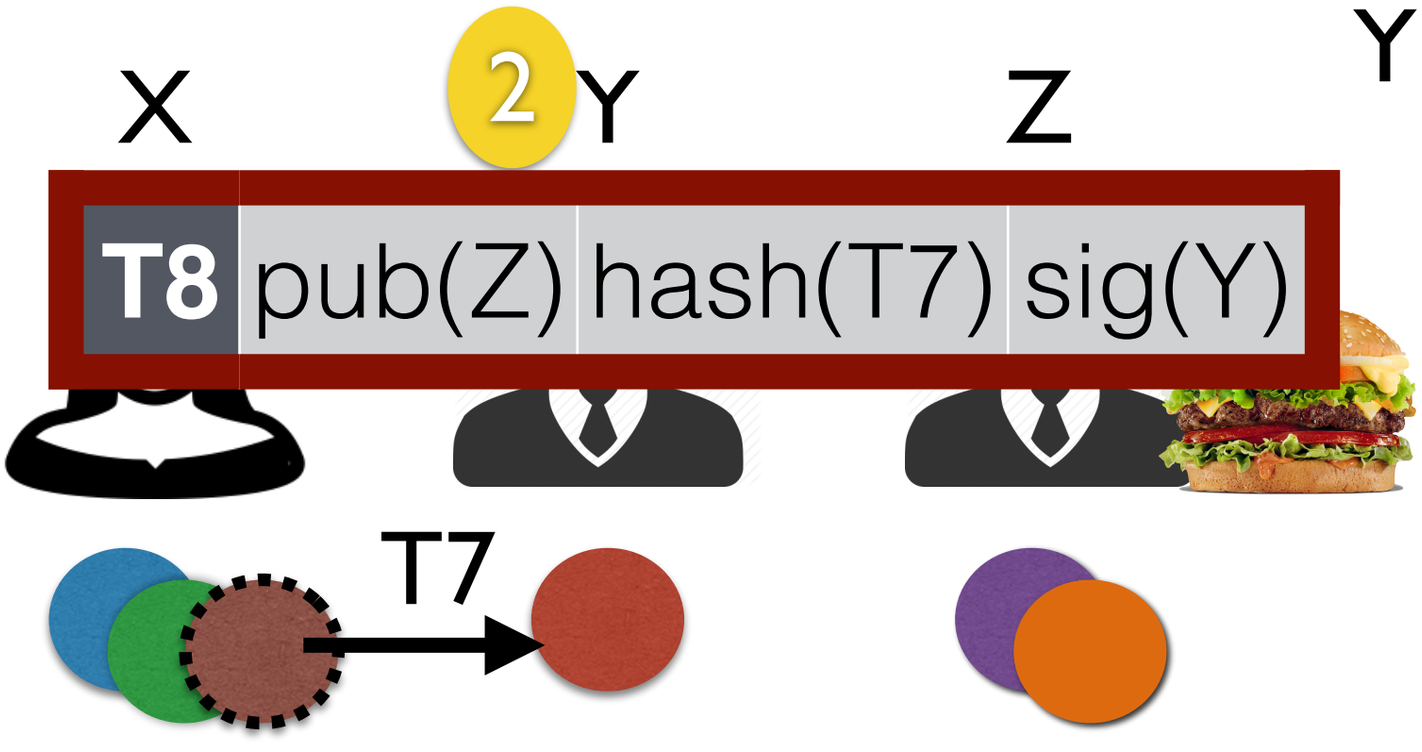
step 1: Z tells Y Z's public key ("address")

OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)



Y buys hamburger from Z

step 2: Y creates a new transaction and signs it

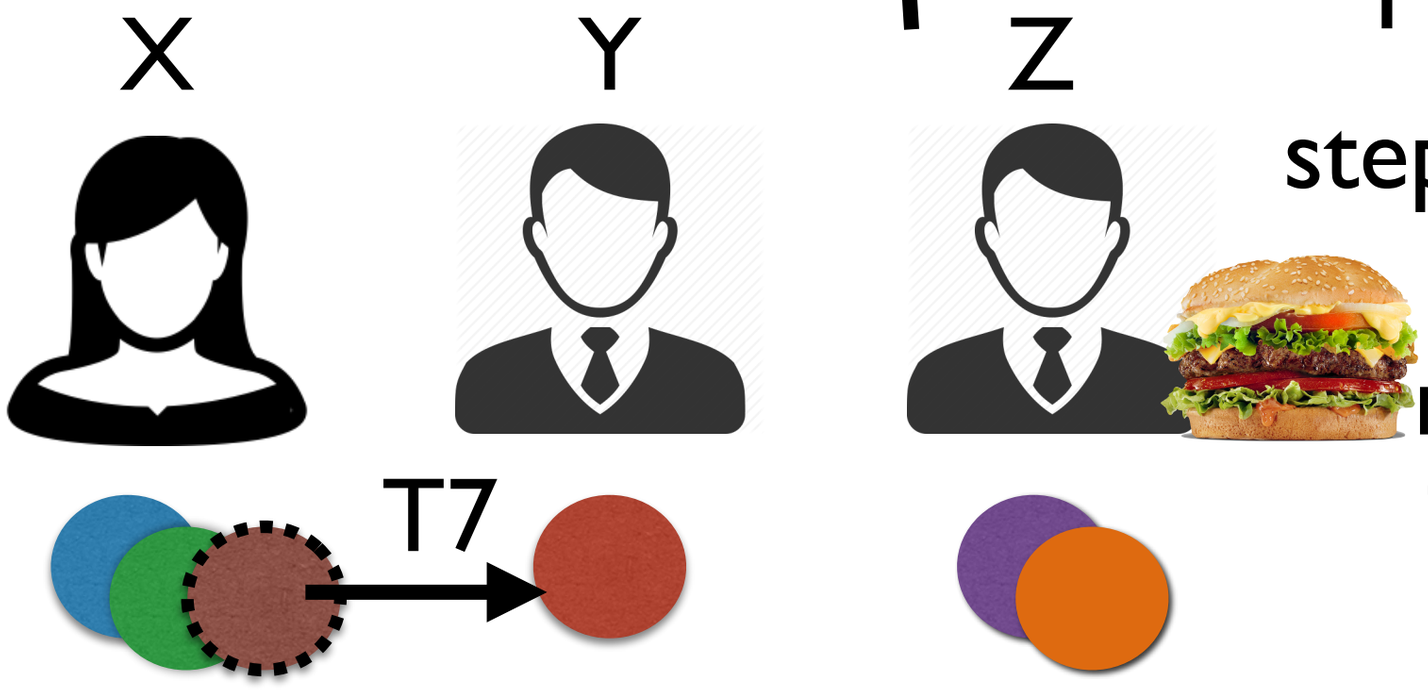
OneBit: Example

3

T8	pub(Z)	hash(T7)	sig(Y)

hamburger from Z

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)



Y buys hamburger from Z

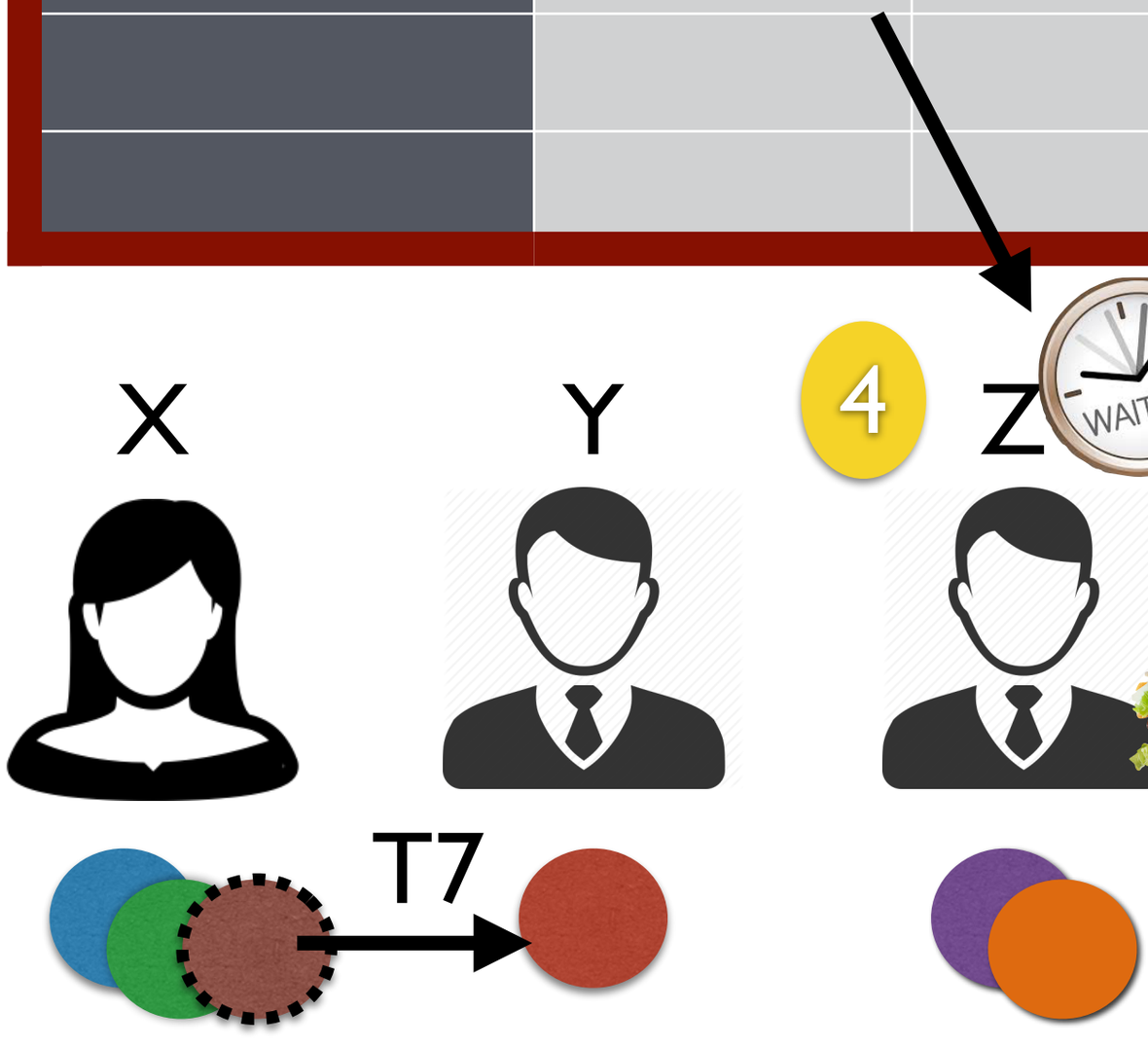
step 3: OneBank verifies that: no other transaction mentions hash(T7), and T8's sig() corresponds to T7's pub()

OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)



Y buys hamburger from Z

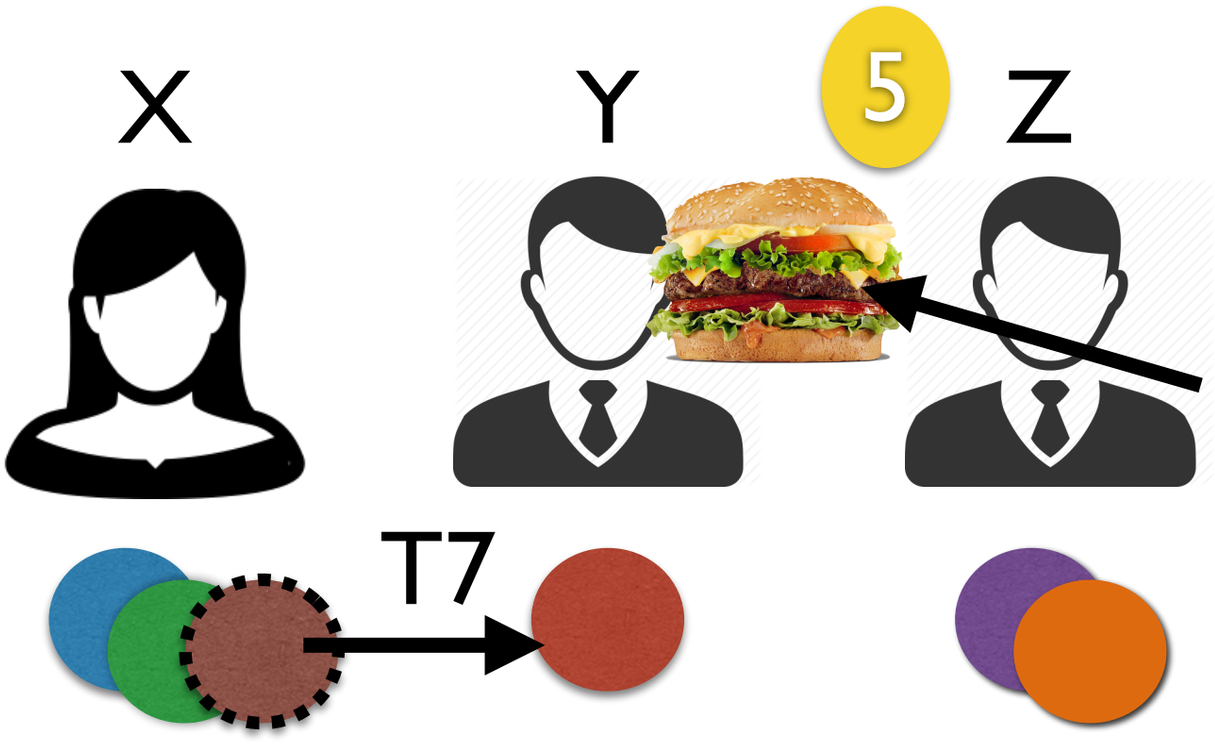
step 4: Z waits until OneBank has seen/verified T8, verifies that T8's pub() is Z's public key

OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)



Y buys hamburger from Z

step 4: Z gives Y the goods!

OneBit: Example

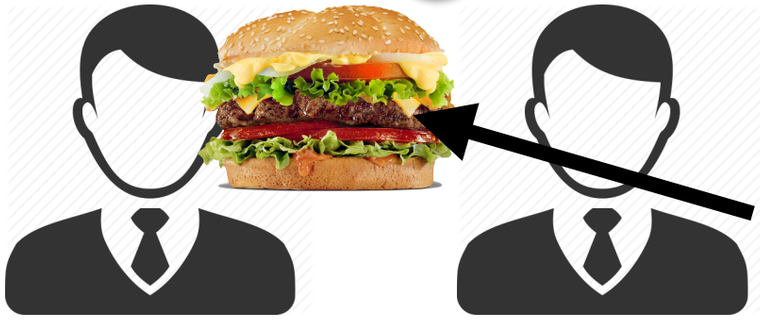
Y buys hamburger from Z

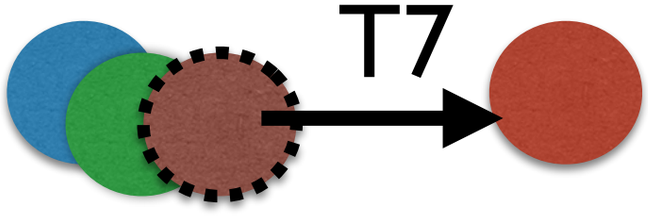
OneBank
The Ledger

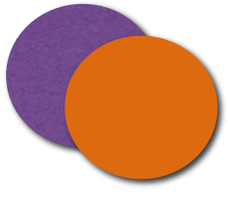
tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)

Why include pub(Z)?
Why sign with sig(Y)?

X







ger from Z

step 4: Z gives Y the goods!

OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

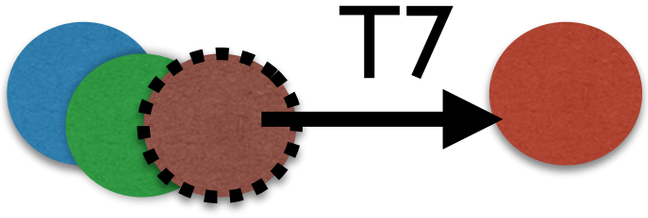
tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)

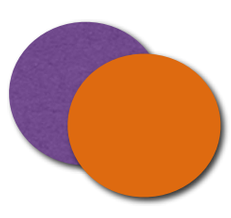
Why include hash(T7)?

X

om Z

step 4: Z gives Y the goods!





OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

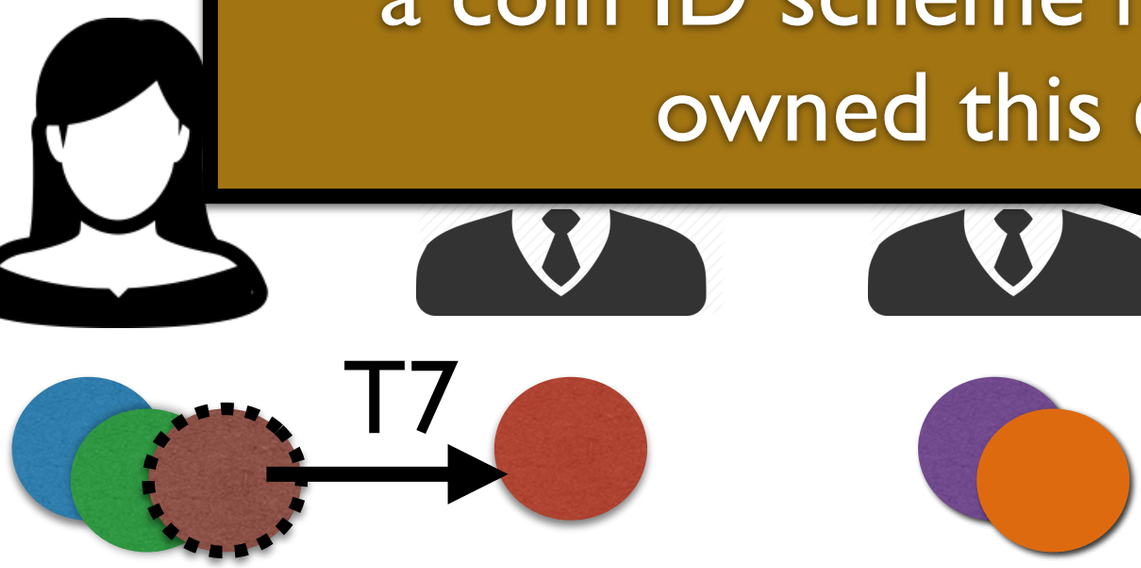
tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)

Why include hash(T7)?
hash(T7) identifies the coin and exact previous transaction to be spent —
a coin ID scheme might be ambiguous if Y owned this coin previously

X

om Z

step 4: Z gives Y the goods!



OneBit: Example

Y buys hamburger from Z

OneBank
The Ledger

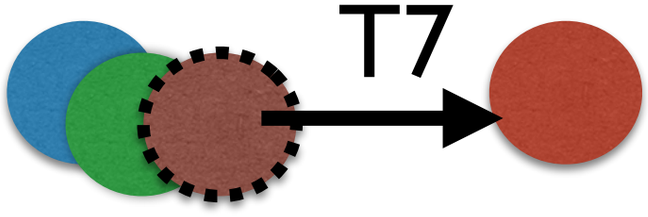
tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)

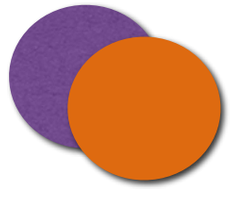
Coin's "identity" is **most recent** TR (hash of it)
Z "owns" the coin : has private key that allows Z to make the next TR

om Z

oods!

X





OneBit: Analysis

OneBank
The Ledger

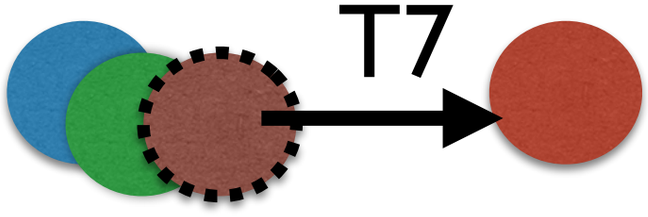
tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)

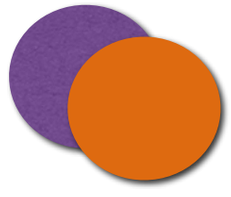
Does OneBit's transaction chain prevent stealing?

X

om Z

oods!





OneBit: Analysis

OneBank
The Ledger

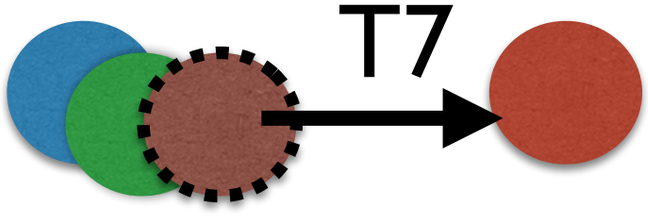
tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)

What if OneBank is corrupt?

X

om Z

oods!



OneBit: Analysis

OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)
8	Z	hash(T7)	sig(Y)

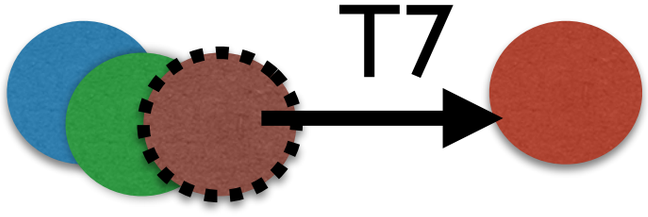
What if OneBank is corrupt?

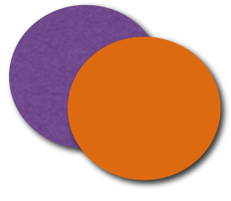
It can't forge transactions (doesn't know private keys) but it can help people double-spend!

X

om Z

oods!





OneBit: Double Spending

OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)

suppose OneBank is cooperating with Y to cheat Z or Q

Q

Y

Z

OneBit: Double Spending

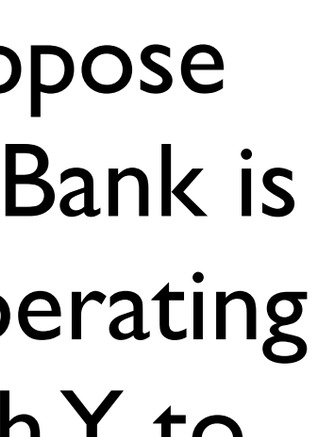
OneBank
The Ledger

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)

suppose OneBank is cooperating with Y to cheat Z or Q

Q

Y



Z

T9	pub(Q)	hash(T7)	sig(Y)

T8	pub(Z)	hash(T7)	sig(Y)

OneBit: Double Spending

T9			
T8	pub(Z)	hash(T7)	sig(Y)

tr#	pub	prev	sig
7	Y	hash(T6)	sig(X)

suppose OneBank is cooperating with Y to cheat Z or Q

OneBit: Double Spending

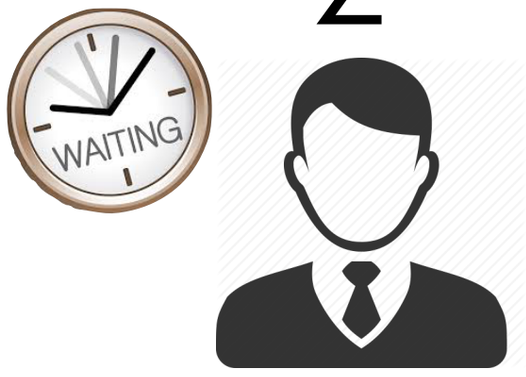
T9			
T8	pub(Z)	hash(T7)	sig(Y)

tr#	pub	prev	sig
7	**Y**	**hash(T6)**	**sig(X)**

suppose OneBank is cooperating with Y to cheat Z or Q

ok:Y to Q

ok:Y to Z

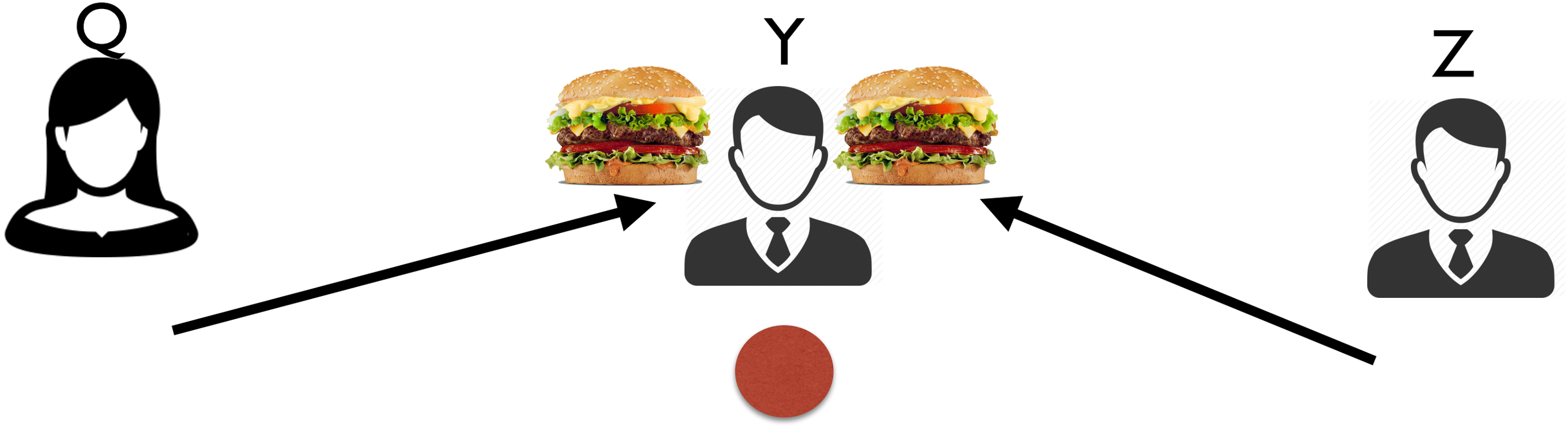


OneBit: Double Spending

T9			
T8	pub(Z)	hash(T7)	sig(Y)

tr#	pub	prev	sig
7	**Y**	**hash(T6)**	**sig(X)**

suppose OneBank is cooperating with Y to cheat Z or Q



OneBit: Double Spending

T9			
T8	pub(Z)	hash(T7)	sig(Y)

tr#	pub	prev	sig
7	**Y**	**hash(T6)**	**sig(X)**

suppose OneBank is cooperating with Y to [illegible] or Q

Why was double-spending possible?
OneBank can *hide* some information/ selectively reveal it — Lie in verification

OneBit: Double Spending

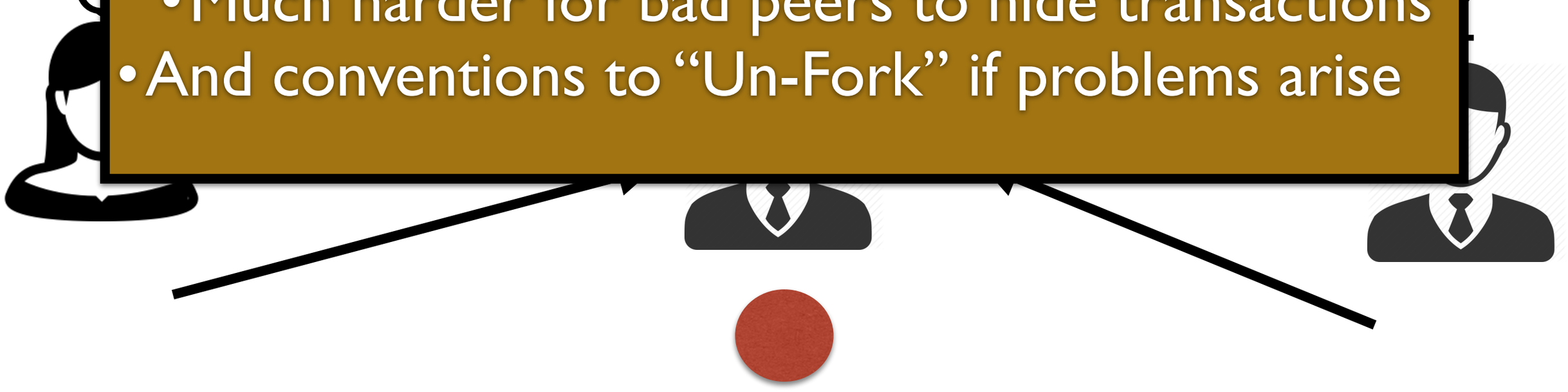
T9			
T8	pub(Z)	hash(T7)	sig(Y)

tr#	pub	prev	sig
Z	X	hash(T6)	sig(X)

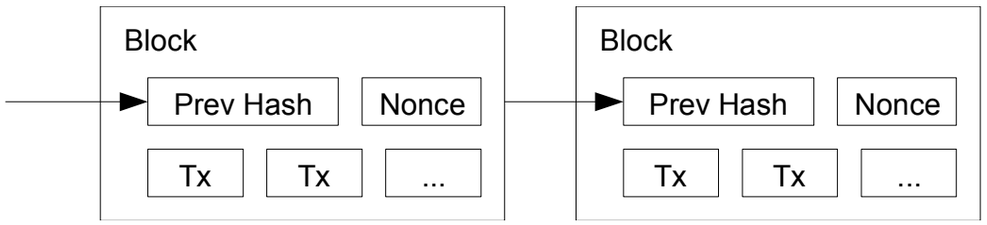
suppose OneBank is cooperating to [illegible]r Q

What is Needed?

- Many Servers ("peers")
  - Send all transactions to all peers
  - Much harder for bad peers to hide transactions
- And conventions to "Un-Fork" if problems arise

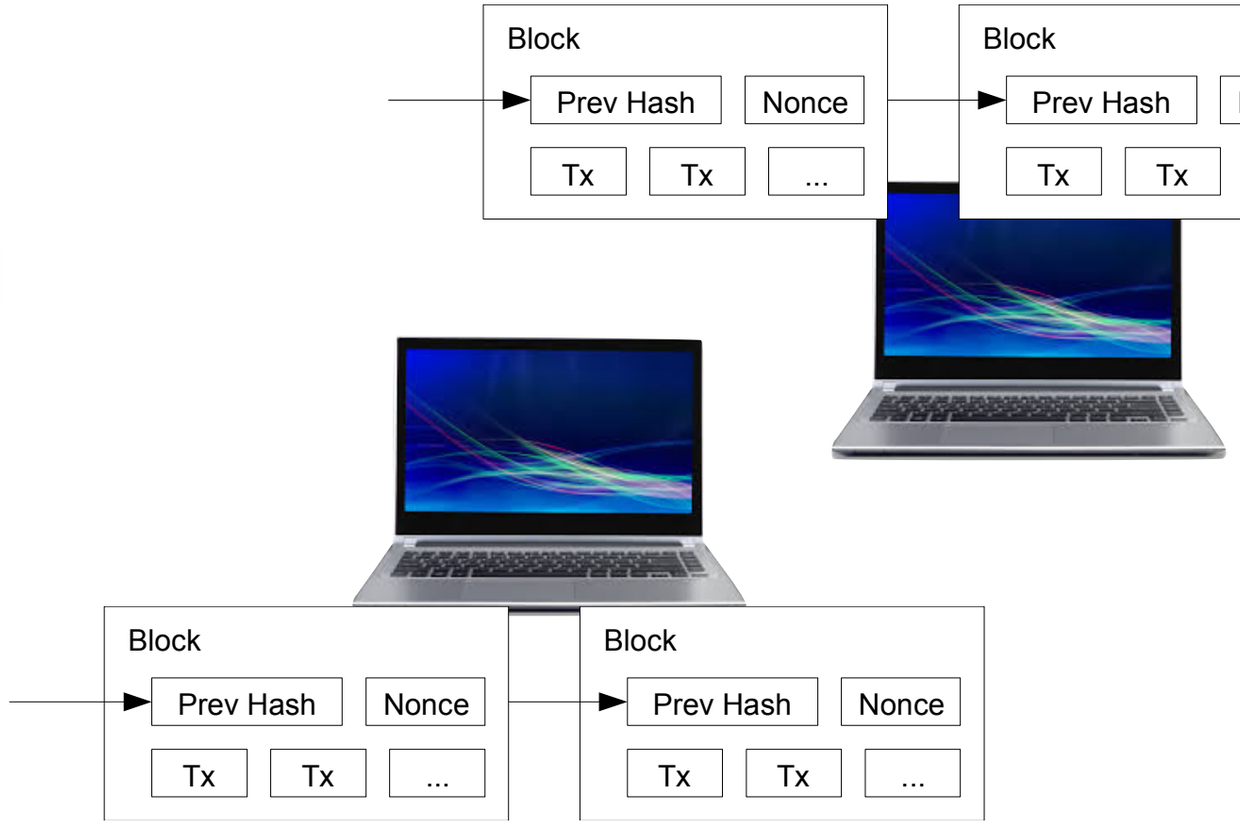


The BitCoin Block Chain



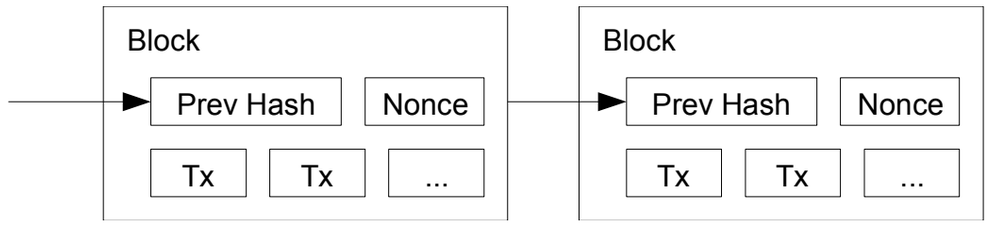
- Single Block chain contains transactions on all coins

- Copy stored at each peer

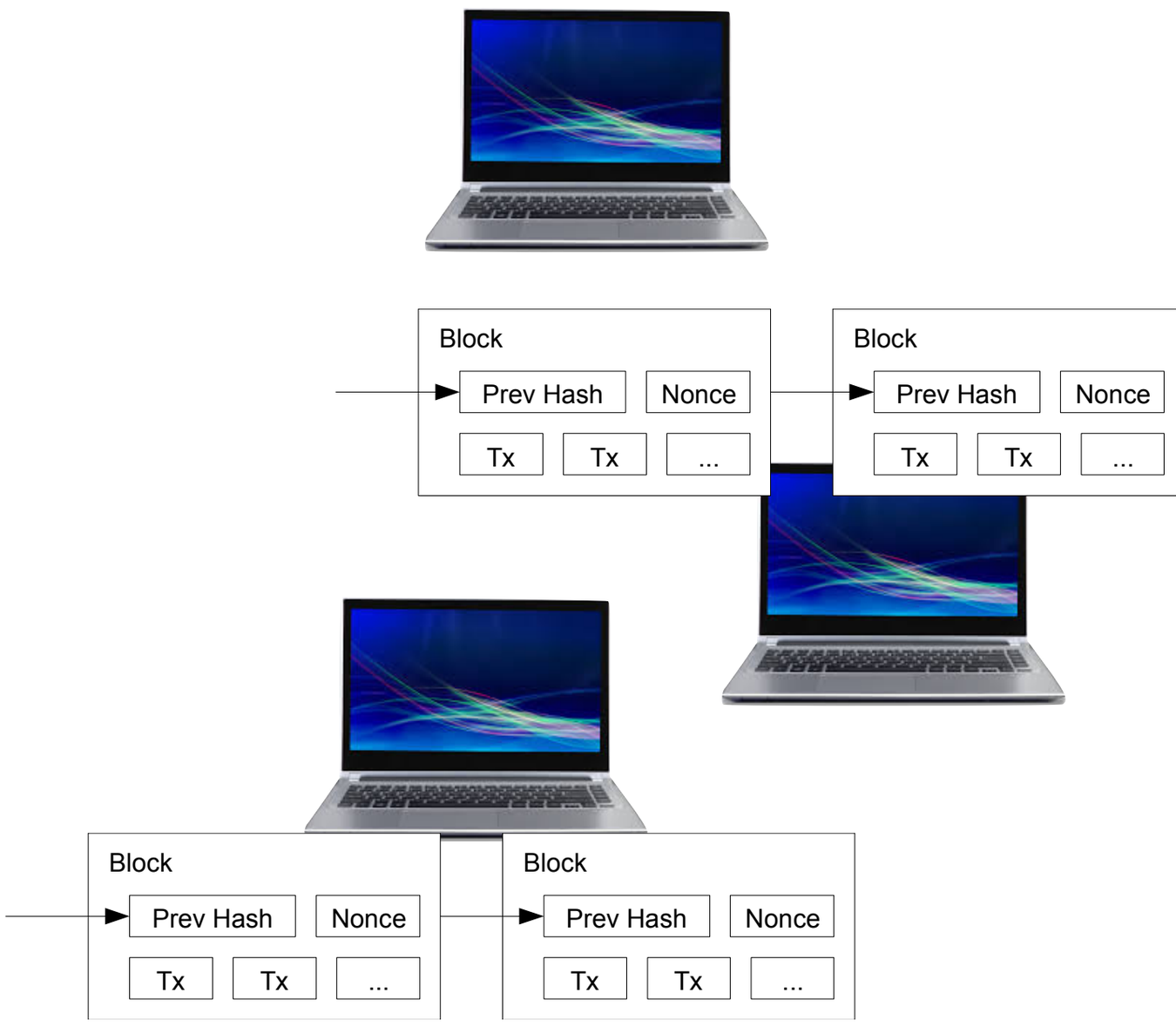


  - Each peer can validate a new transaction against old ones — valid update

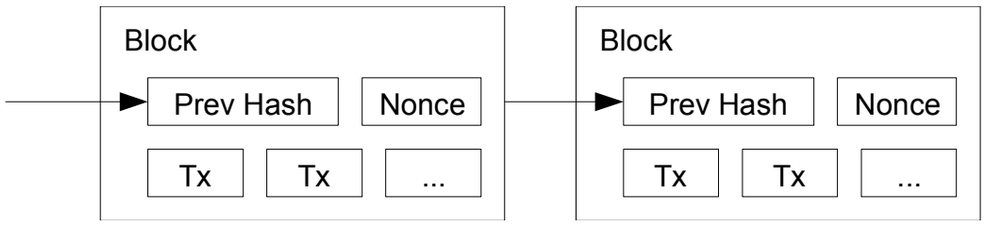
The BitCoin Block Chain



- Each Block:
  - hash(prevblock)
  - set of transactions
  - a “NONCE” — number that causes hash of block to meet a criteria

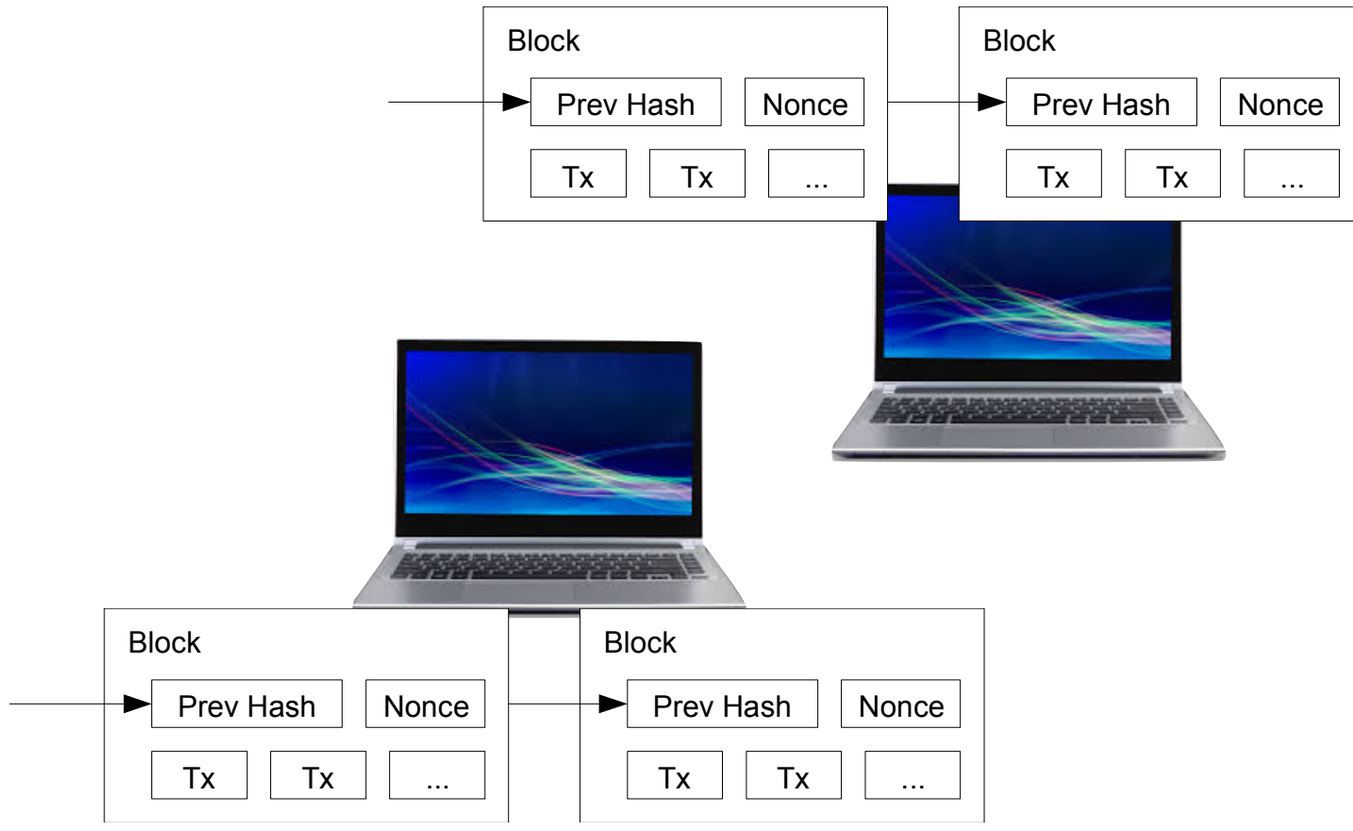


The BitCoin Block Chain



- A transaction is not validated unless it is in the block chain

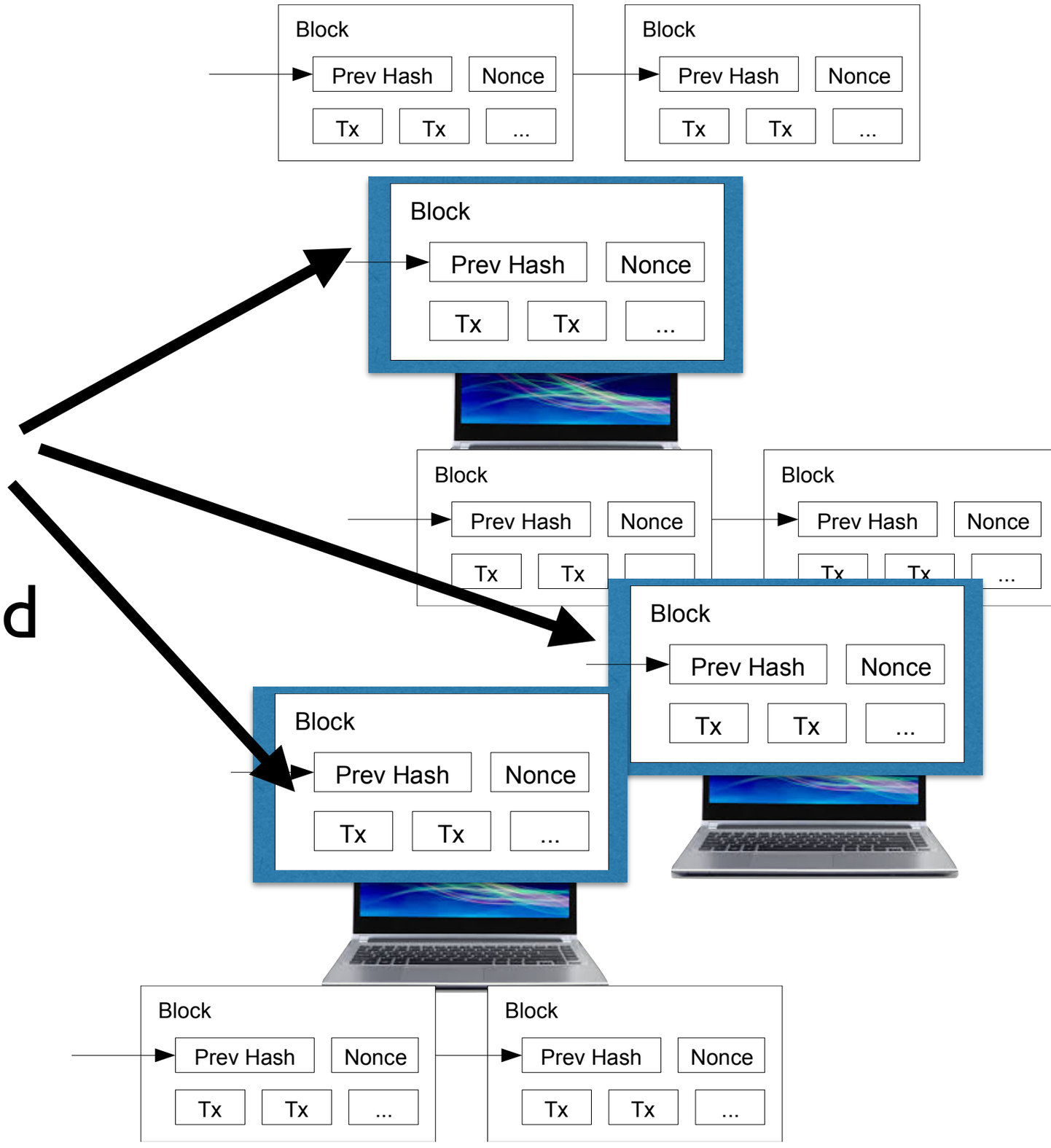
- New Block every 10 minutes containing TR since previous block



Who creates new Block?

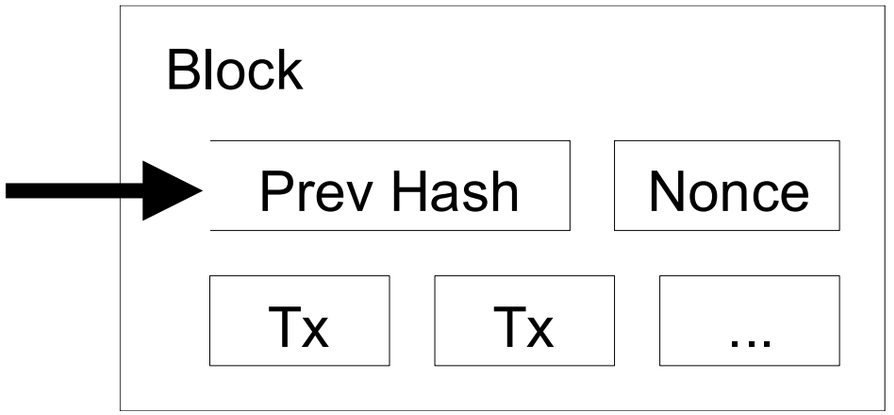
Transactions
sent to all peers.

All peers collect
into block and try to add



Who creates new Block?

SHA256(previous block)



One-way function

The only way to produce
a target hash value
is brute force

If 256 bits for the hash:
2^{256} different values!

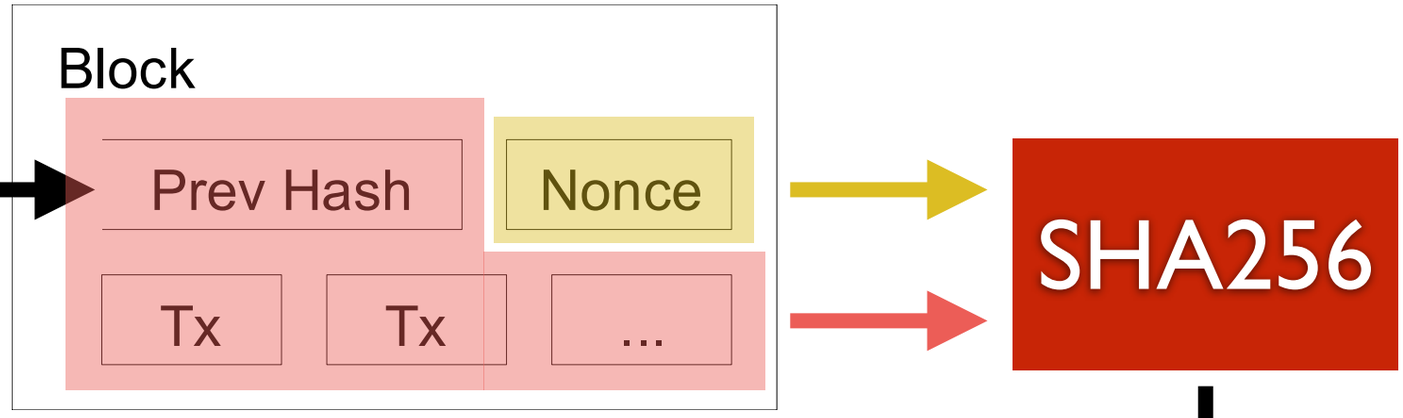
Who creates new Block?

SHA256(previous block)

One-way function

The only way to produce
a target hash value
is brute force

If 256 bits for the hash:
2^{256} different values!

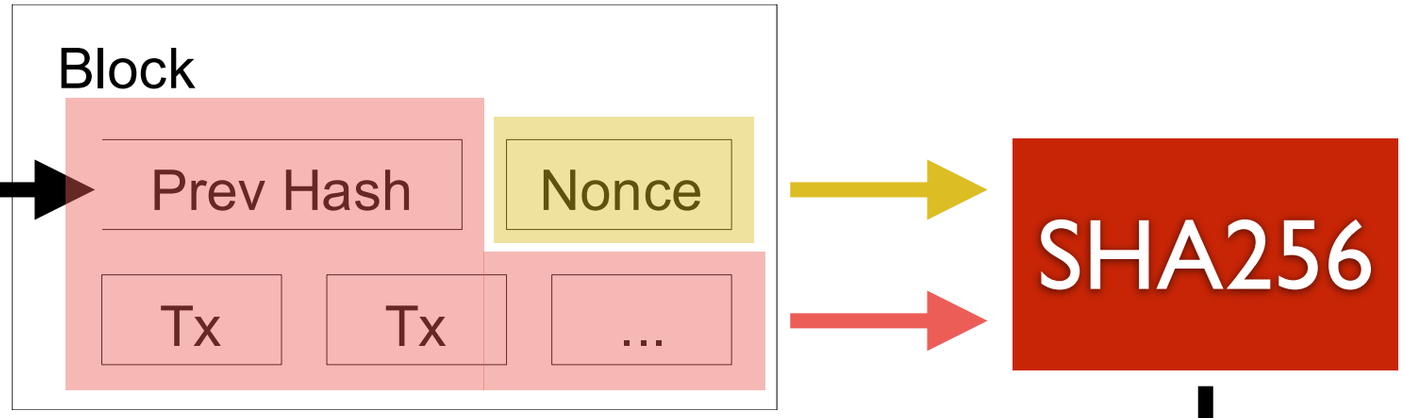


A value with a specific
property, e.g. 9 leading zeros

Who creates new Block?

SHA256(previous block)

One-way function



A value with a specific property, e.g. 9 leading zeros

The only way to produce a target hash value is brute force

If 256 bits for the hash: 2^{256} different values!

16^{64-9} valid targets

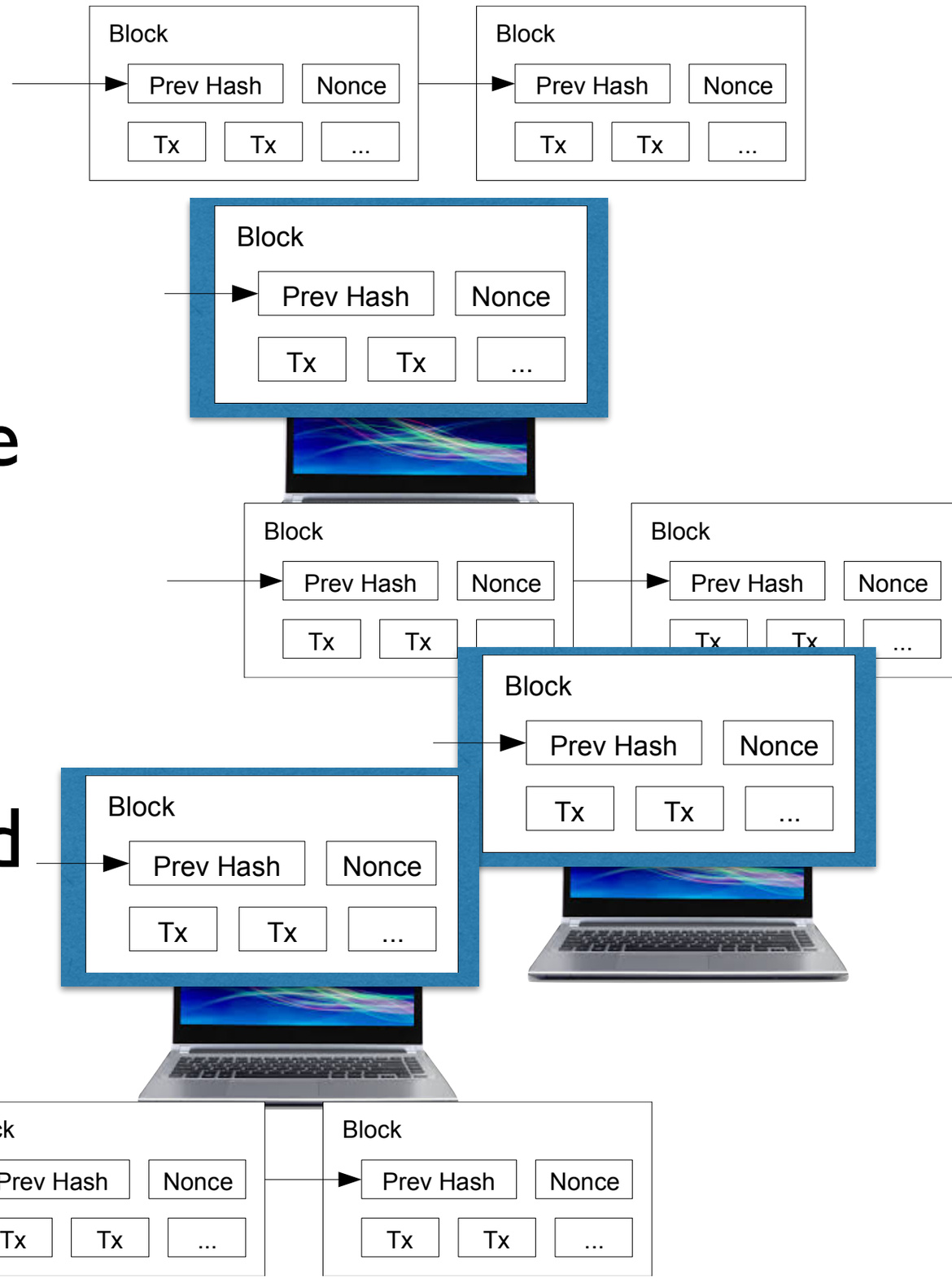
Target: `x00000000????????????????????????????????????????????????????????`

P = 16^{55} (valid targets) / 16^{64} (2^{256} total values) = 16^{-9} = 1.46E-11

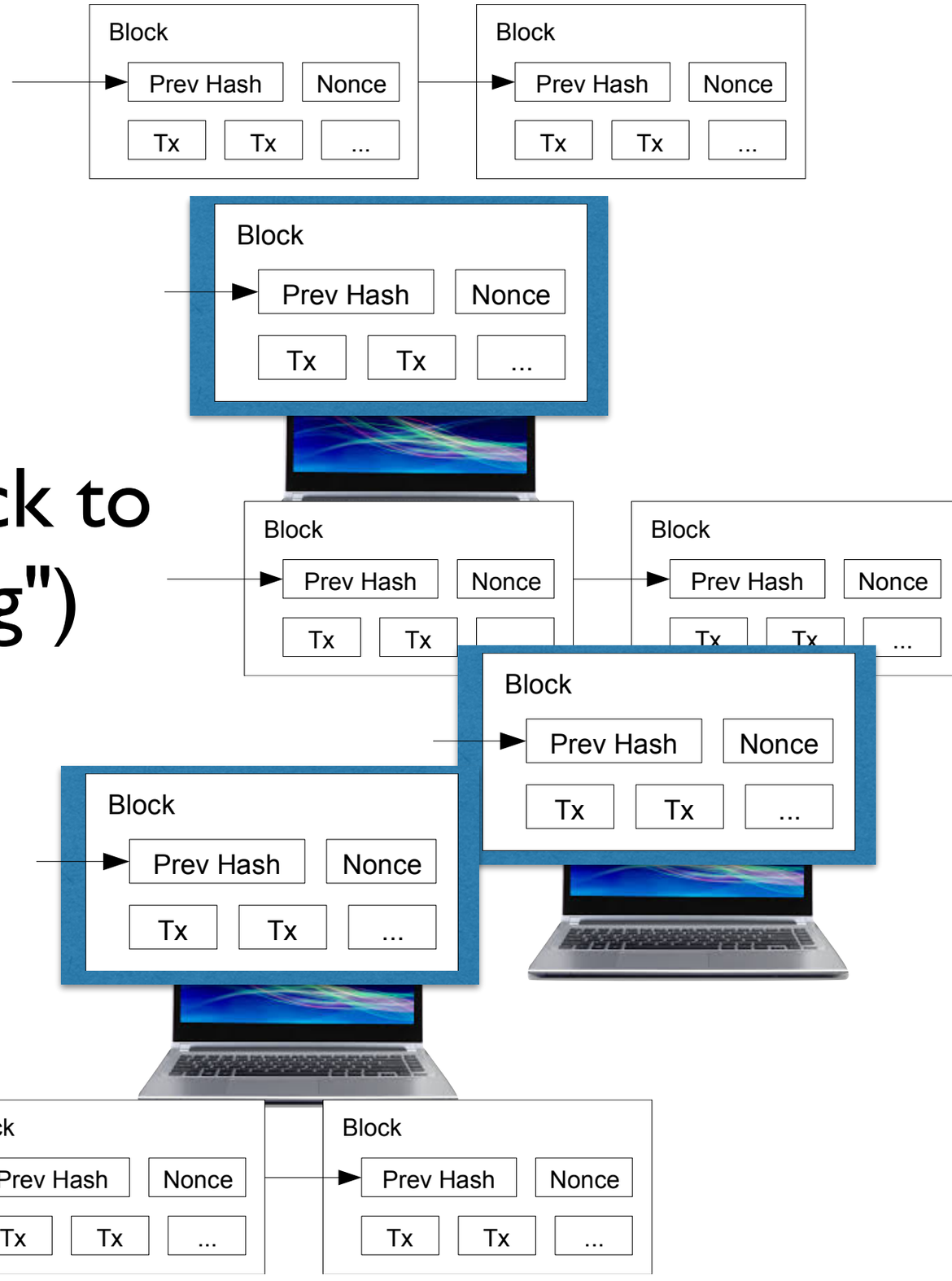
How many tries until a hit on average?
1/P = 68.5 billion tries

Who creates new Block?

Can't predict a winning nonce, since cryptographic hash trying one nonce is fast, but most nonces won't work it would take one CPU months to create one block
but thousands of peers are working on it such that expected time to first to find is about 10 minutes



Who creates new Block?



The winner sends the new block to all peers (this is part of "mining")

Transactions and Block Chain

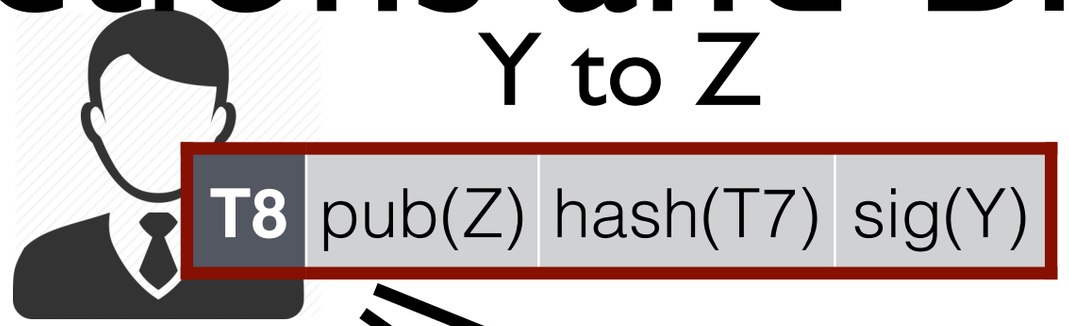
- All peers know Block 5

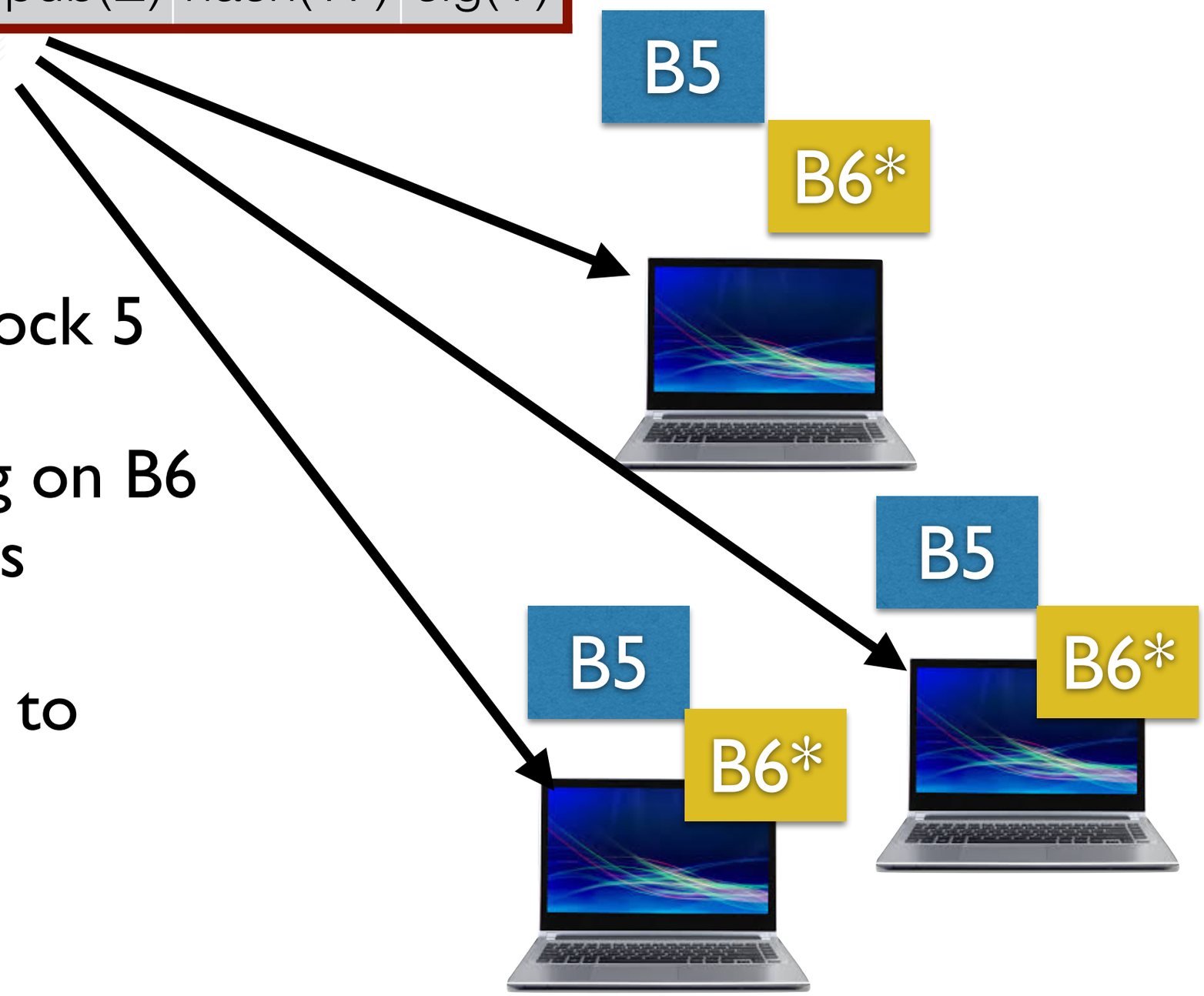
Transactions and Block Chain

- All peers know Block 5

- and are working on B6
  — trying nonces

Transactions and Block Chain

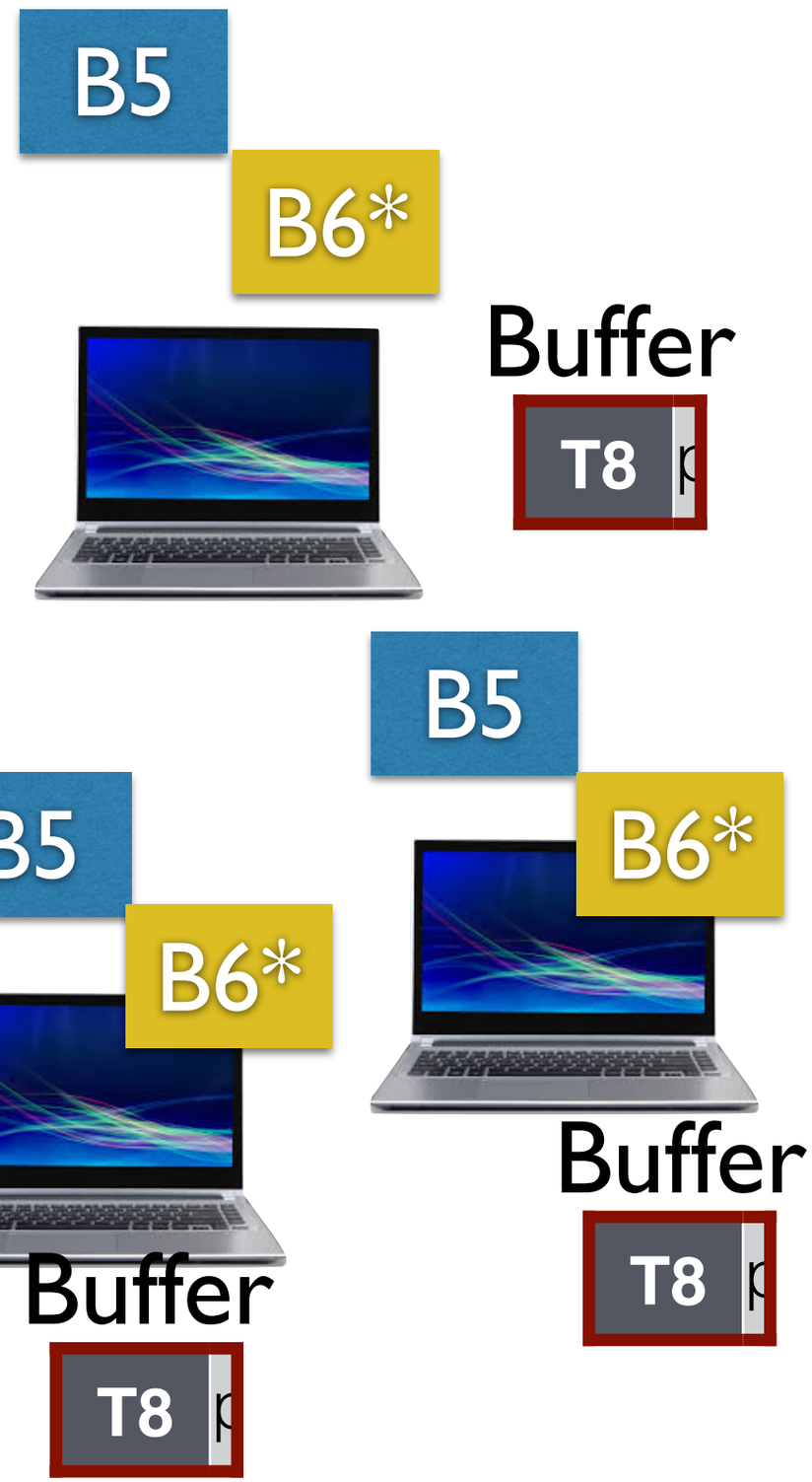




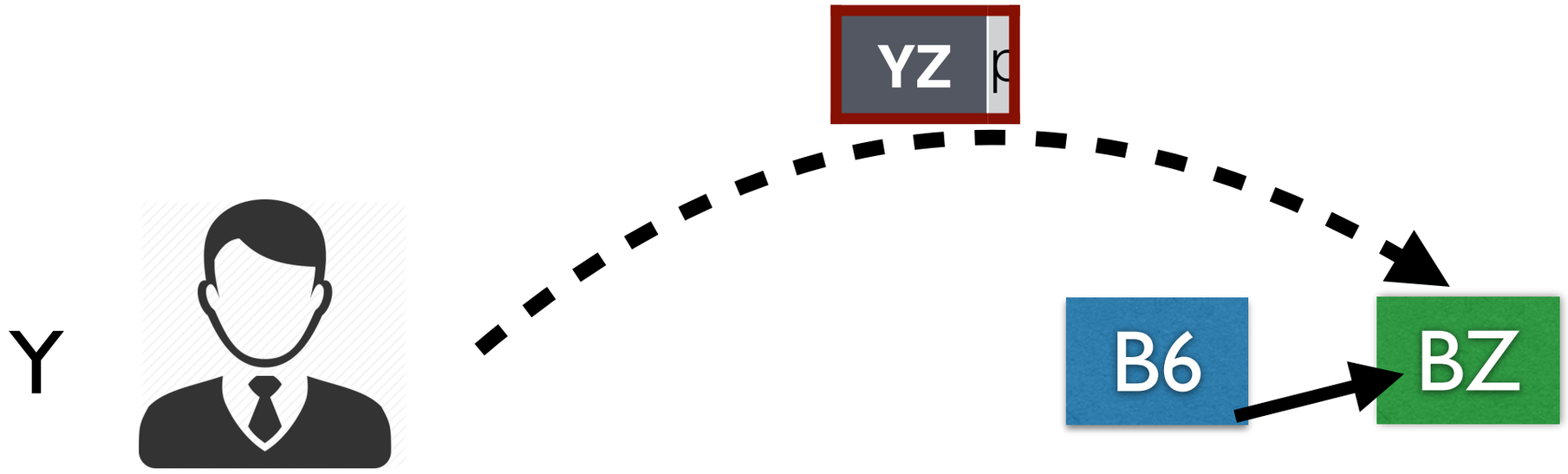
- All peers know Block 5
  - and are working on B6 — trying nonces
- Y sends TR Y to Z to peers — Flood

Transactions and Block Chain

- All peers know Block 5
  - and are working on B6 — trying nonces
- Y sends TR Y to Z to peers — Flood
- Peers buffer until B6 done
- Those that heard T8 include it in next block
- Eventually B5,B6,B7 that include Y to Z



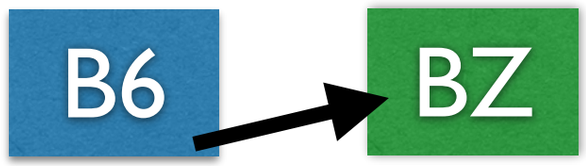
BitCoin: Double Spending



Lets assume bad Y gets Transaction “Y to Z” in Block BZ chained to existing block B6

BitCoin: Double Spending

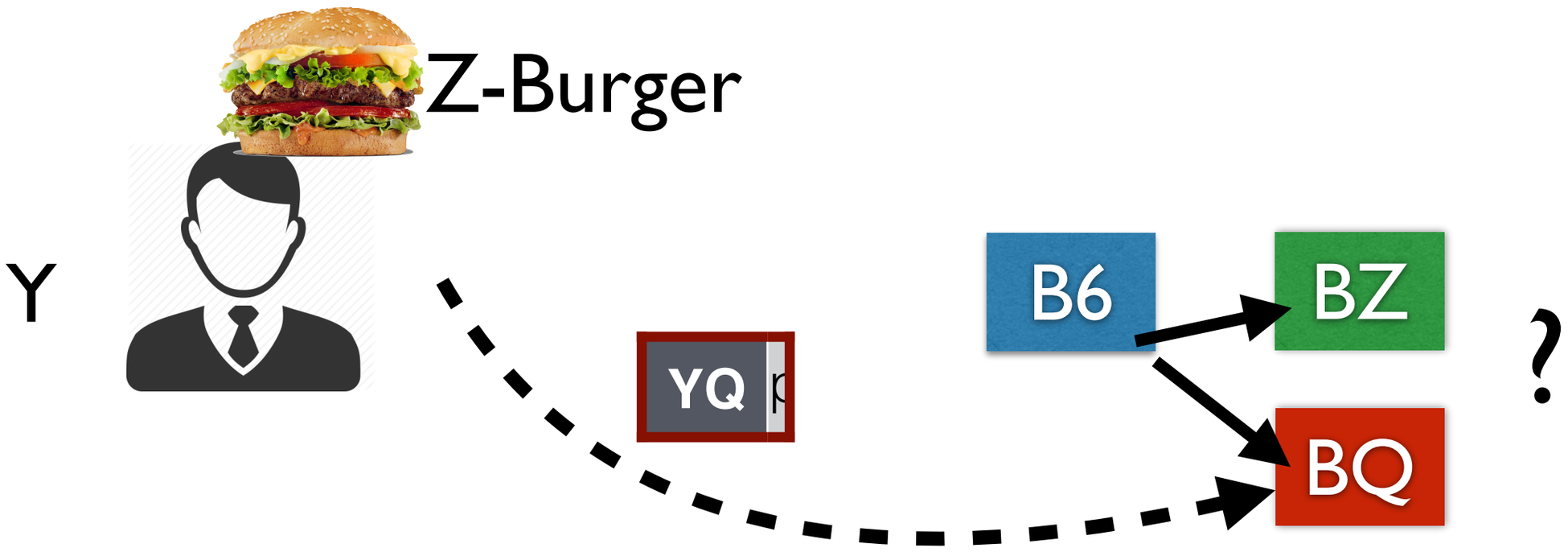
Y 



Lets assume bad Y gets Transaction “Y to Z” in Block BZ chained to existing block B6

Z will see “Y to Z” in chain and give hamburger to Y

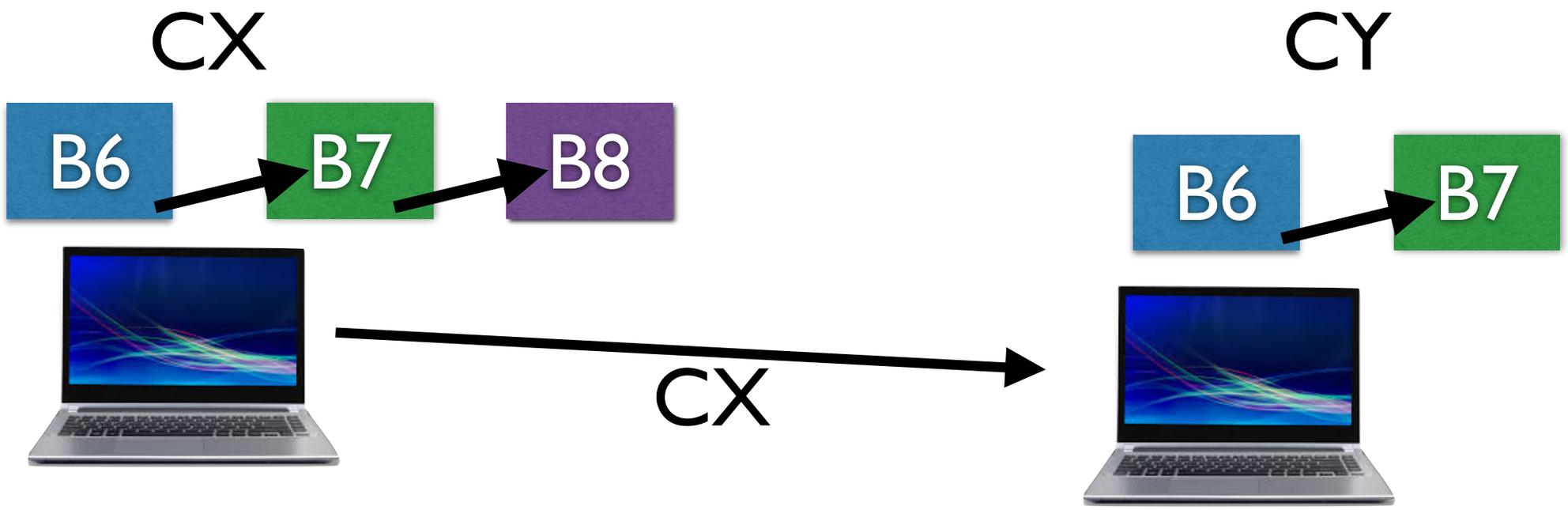
BitCoin: Double Spending



Can Y create another chain from B6 to BQ and persuade peers to accept it in place of B6,BZ?

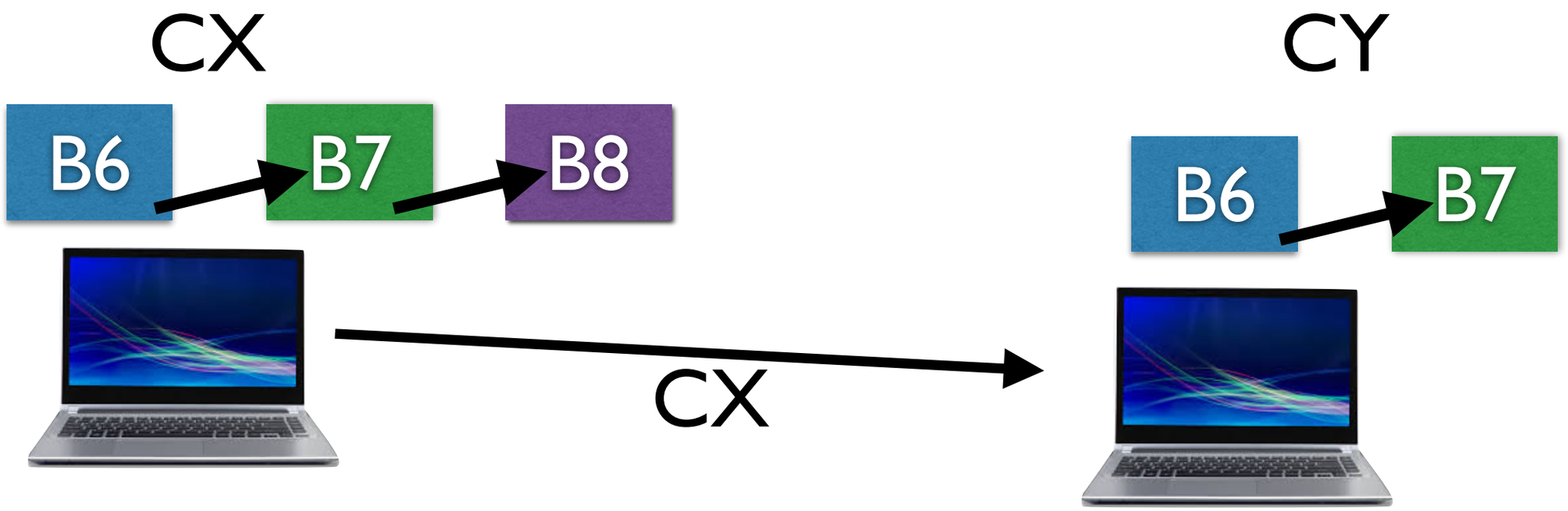
Making Length count

- Rule:
  - Peer will accept chain it hears from another peer if new chain is longer



Making Length count

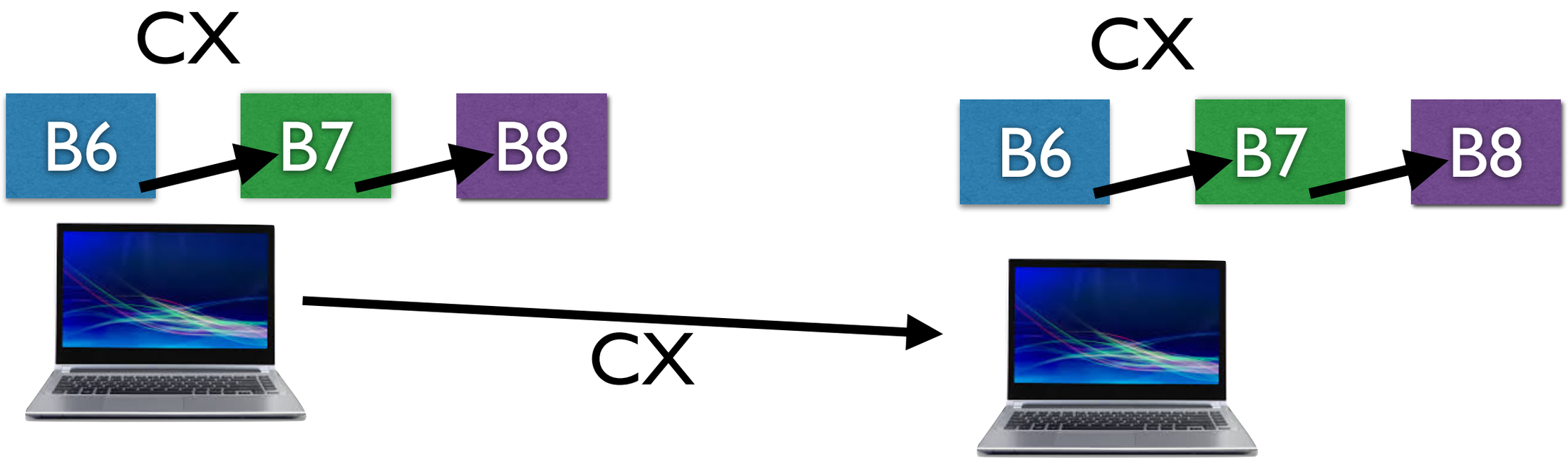
- Rule:
  - Peer will accept chain it hears from another peer if new chain is longer



Accept CX
if len(CX) > len(CY) & valid(CX)

Making Length count

- Rule:
  - Peer will accept chain it hears from another peer if new chain is longer



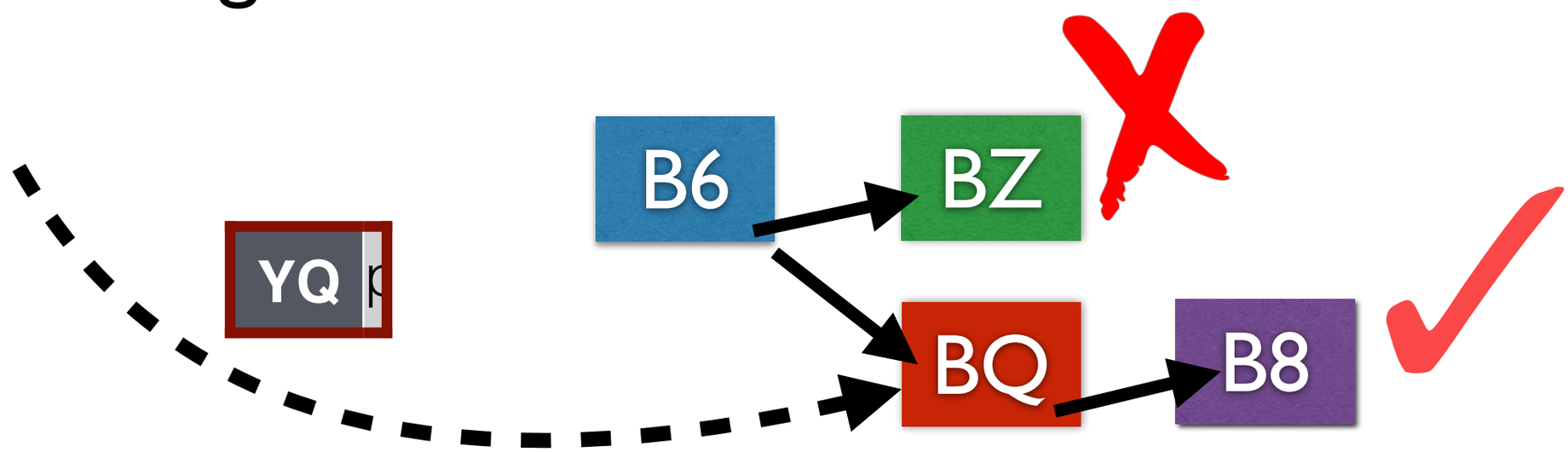
Accept CX
if len(CX) > len(CY) & valid(CX)

Double Spending

Q-Burger

Z-Burger

Y



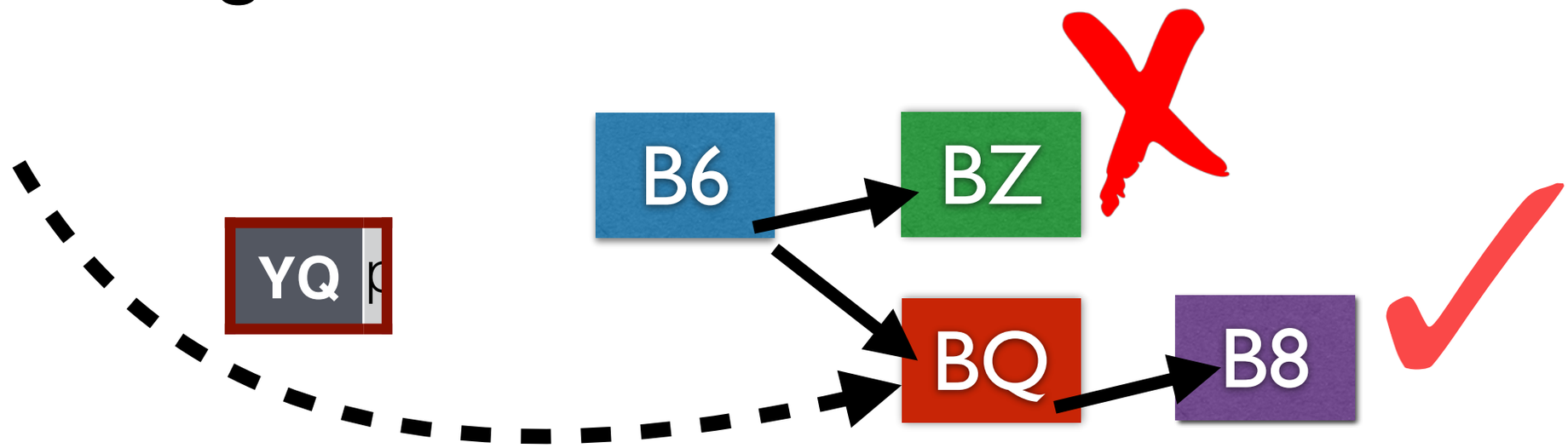
- Attacker has to get a longer chain to double spend
  - and must create it in less than 10 minutes
- 10 minutes is the time it takes the 1000's of honest peers to find one block

Double Spending

Q-Burger

Z-Burger

Y

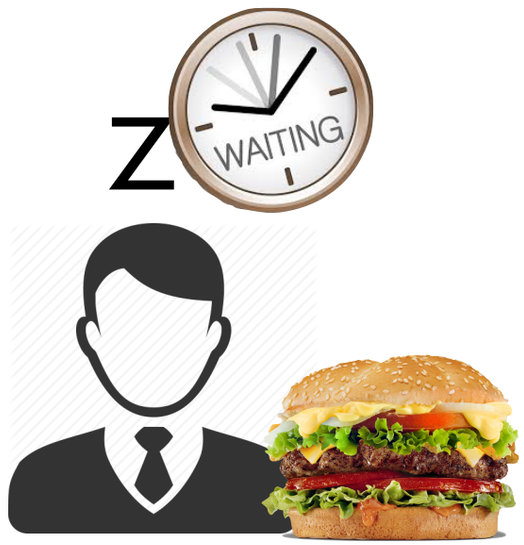


- If attacker has only 1 cpu it will take months
  - by that time the main chain will be very long — no peer will accept the shorter chain — attacker SOL
- But powerful attacker can win — double spend

Double Spending : Attacker can force honest peers to switch chains if attacker controls majority of peer CPU power

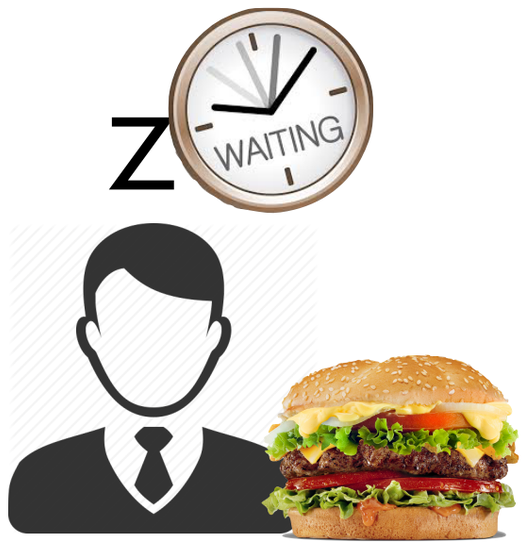
How long to wait?

- Until Z sees Y flood the transaction to many Peers?



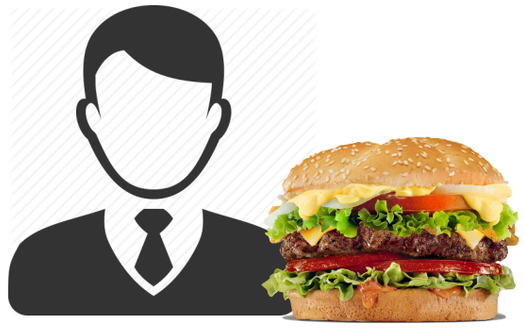
How long to wait?

- Until Z sees chain B7,BZ?



How long to wait?

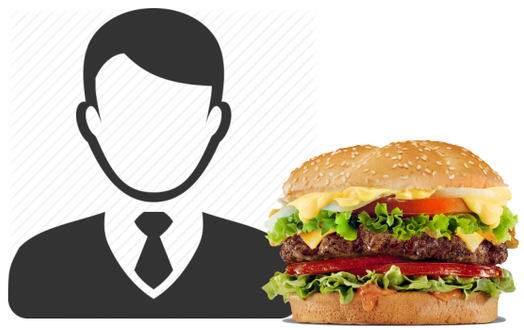
- Until Z sees chain B7,BZ?
  - Maybe — risky since there is a non-zero chance that some other chain will win
  - eg. some lucky machine mined a few blocks in a row, but its messages are delayed and its chain does not have “Y to Z”



How long to wait?

- Until Z sees chain B7,BZ with several following blocks

- Yes — slim chance the attacker with few CPUs could catch up



Summary of validation steps

- Peers on receive of new transaction:
  - no other transaction refers to same prev
  - signature is by private key of previous transaction — then will add to trans set of new block to be mined
- Peer on receive of new Block:
  - hash value < target — nonce is right — proof of work
  - previous block hash exists
  - new chain longer than current chain
  - all transactions in block are valid

Where does each bitcoin come from?

- Each time a peer creates a block — gets some bitcoins — amusing winner for that block
  - It puts its public key in a special transaction in the block
  - This is an incentive for people to operate bitcoin peers
  - value halves every 210,000 blocks (~4 years)

Summary

- Key Idea: Block Chain
  - Public ledger seems like a great idea
  - Decentralization might be good
  - Mining seems like a kludge but does avoid centralized trust
- Tying Ledger to a new currency seems bad — does not seem like it was necessary