

UnB

MIT Digital Currency Initiative and the University of Brasilia presents

Cryptocurrency Design and Engineering

Lecture 2: Trust Minimization

Taught by: Dr. Ethan Heilman

September 9, 2025

MAS.S62

Ethan Heilman

I'm Ethan Heilman (he/him)

<https://ethanheilman.com>,
ethan.r.heilman@gmail.com

I'll be doing some guest lectures over the course (including this one).

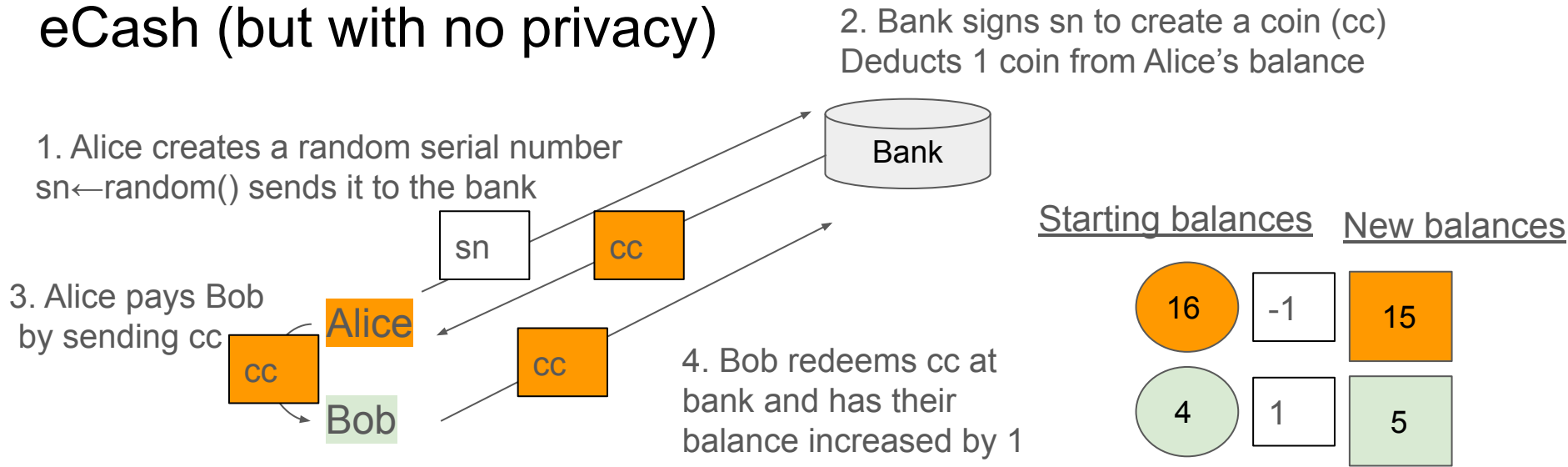
My background:

- I am a cryptographer who does research on cryptocurrencies, hash functions and networking
- I contribute to Bitcoin-core, involved in the Bitcoin standards process

Lecture overview

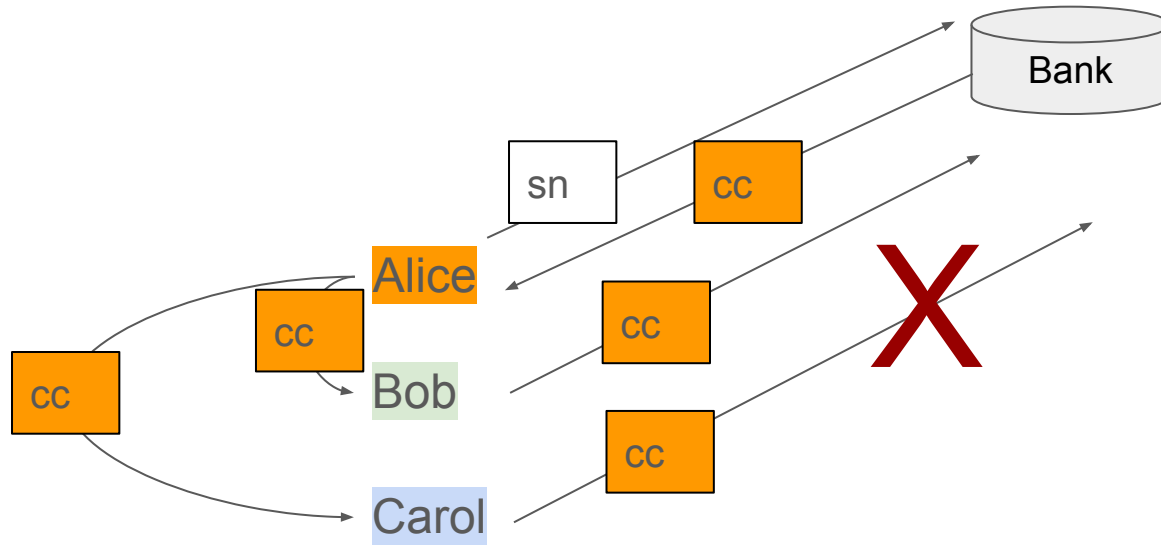
1. We are going start with a look at an early cryptocurrency: eCash
2. Then see how we got from eCash to where we are today
3. Finally we are going to get into some details of Bitcoin

eCash (but with no privacy)



eCash bank issues and redeems coins

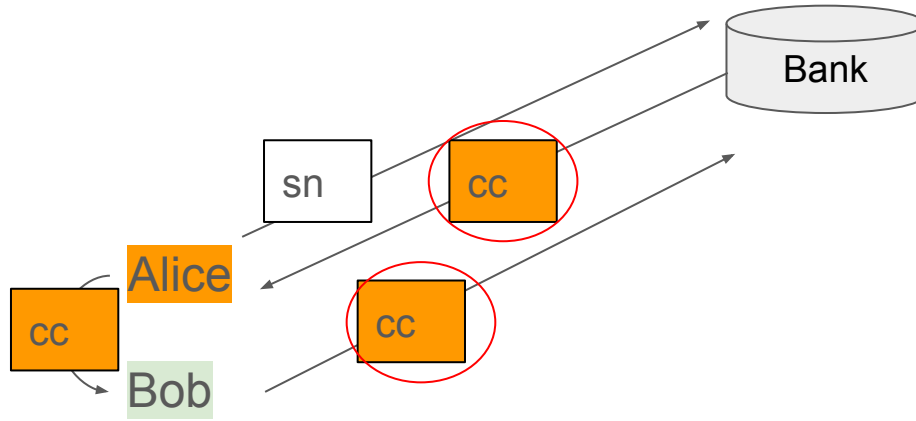
eCash (but with no privacy)



Bank has a list of spent serial numbers (sn). If bank has already redeemed this sn, it does not credit the account

What if Alice Doublespends?

eCash (but with no privacy)



Bank can see the cc it issued to Alice is the same one Bob redeemed. Bank learns Alice paid Bob.

No privacy from the bank! eCash has a fix for this

eCash with privacy (Chaumian eCash)

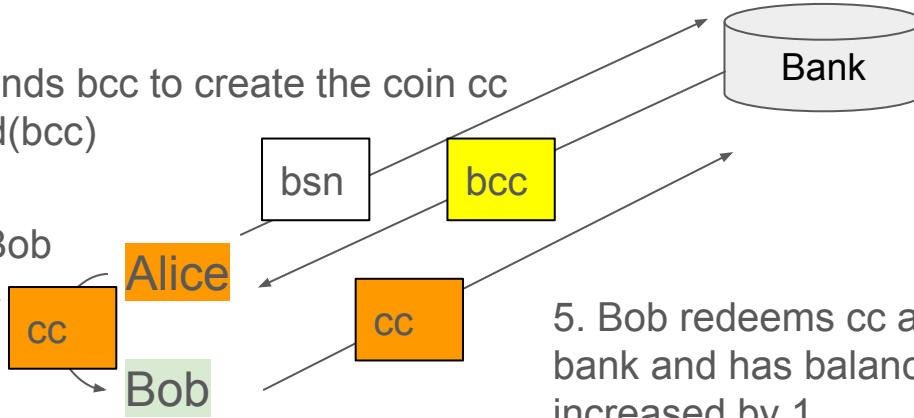
1. Alice creates a random serial number
 $sn \leftarrow \text{random}()$, blinds this sn to create
 $bsn \leftarrow \text{blind}(sb)$ and sends bsn to the bank

2. Bank signs bsn to create a blinded coin (bcc)
Deducts 1 coin from Alice's balance.

3. Alice unblinds bcc to create the coin cc
 $cc \leftarrow \text{unblind}(bcc)$

4. Alice pays Bob
by sending cc

5. Bob redeems cc at
bank and has balance
increased by 1



Bank can't link the bcc it issued with the cc Bob redeemed

Each cc , aka coin, is unlinkable

eCash with privacy (Chaumian eCash)

Alice { $sn \leftarrow \text{random}()$ // serial number
 $b \leftarrow \text{random}()$ // blinding factor
 $bsn \leftarrow \text{Blind}(b, sn)$

Bank $bcc \leftarrow \text{Sign}(sk, bsn)$

Alice { $cc \leftarrow \text{Unblind}(b, bsn)$

Assume your signature algorithm is commutative (can change order)

$cc \leftarrow \text{Unblind}(b, \text{Sign}(\text{Blind}(b, sn)))$

$= \text{Sign}(\text{Unblind}(b, \text{Blind}(b, sn)))$

$= \text{Sign}(sk, sn)$

Blind and Unblind cancel out

RSA Blind signatures (mostly)

$\mathbf{pk} \leftarrow (e, N), \mathbf{sk} \leftarrow d,$

d and e are inverses: $(x^d)^e = x \pmod{N}$

$\mathbf{RSA_Sign}(d, m): m^d \pmod{N} \rightarrow \text{sig}$

$\mathbf{RSA_Verify}(e, \text{sig}): (\text{sig}^e) \pmod{N} == m \pmod{N}?$

Alice

$\text{sn} \leftarrow \text{random()} // \text{serial number}$

$b \leftarrow \text{random()} // \text{blinding factor}$

$\text{bsn} \leftarrow \text{Blind}(b, \text{sn}) = \text{sn} * b^e \pmod{N}$

Bank

$\text{bcc} \leftarrow \text{Sign}(\text{sk}, \text{bsn}) = \text{bsn}^d = (\text{sn} * b^e)^d = \text{sn}^d * b \pmod{N}$

Alice

$\text{cc} \leftarrow \text{Unblind}(b, \text{bsn}) = \text{bsn} / b = \text{sn}^d * b / b = \text{sn}^d \pmod{N}$

RSA Blind signatures (actually there are hashes)

$\mathbf{pk} \leftarrow (e, N), \mathbf{sk} \leftarrow d,$

d and e are inverses: $(x^d)^e = x \pmod{N}$

$\mathbf{RSA_Sign}(d, m): m^d \pmod{N} \rightarrow \text{sig}$

$\mathbf{RSA_Verify}(e, \text{sig}): (\text{sig}^e) \pmod{N} == m \pmod{N}?$

Alice { $\text{sn} \leftarrow \text{random()} // \text{serial number}$
 $b \leftarrow \text{random()} // \text{blinding factor}$
 $\text{bsn} \leftarrow \text{Blind}(b, \text{sn}) = \underline{\text{Hash}}(\text{sn}) * b^e \pmod{N}$

Bank $\text{bcc} \leftarrow \text{Sign}(\text{sk}, \text{bsn}) = \text{bsn}^d = (\underline{\text{Hash}}(\text{sn}) * b^e)^d = \underline{\text{Hash}}(\text{sn})^d * b \pmod{N}$

Alice { $\text{cc} \leftarrow \text{Unblind}(b, \text{bsn}) = \text{bsn} / b = \underline{\text{Hash}}(\text{sn})^d * b / b = \underline{\text{Hash}}(\text{sn})^d \pmod{N}$

Without hashes this isn't secure, why?

What happened with eCash?

partnership deals worth tens of millions of dollars. Similarly, Visa reportedly offered a \$40 million investment, while Netscape had interest as well: eCash could have been included in the most popular web browser of that era.

Still, the biggest offer of all probably came from none other than Microsoft. Bill Gates wanted to integrate eCash into Windows 95 and is said to have offered DigiCash some \$100 million to do so. Chaum, so the story goes, asked for two dollars for each version of Windows 95 sold. The deal was off.

How David Chaum's eCash Spawned a Cypherpunk Dream (2018), Aaron Van Wirdum,
<https://bitcoinmagazine.com/culture/genesis-files-how-david-chaums-ecash-spawned-cypherpunk-dream>

Why didn't an anonymous party launch an eCash bank?

TRUST!

Trust Minimization

“Trust” in computer science has a special meaning:

- A trusted party is a party must be is trusted to be honest,
...if they get hacked/misbehave, the system is no longer secure.
- Trust is not the same as trustworthiness

“Trust but **verify**”



...but how to verify

Cryptography is the science of minimizing trust.

Trust Minimization: Example

You want to send a confidential message



What bad things can the messenger do?

Trust Minimization: Example

What bad things can the messenger do?

Confidentiality: They can read the message, share it with others.

Integrity: They could alter the message, make it say something else

Availability: They could throw the message away or never deliver it

You could **trust** the messenger not to do these things

OR

design a protocol to not have to trust the messenger

Trust Minimization: Example

What bad things can the messenger do?

Confidentiality: They can read the message, share it with others.
Encrypt the message so no one can read it but the recipient

Integrity: They could alter the message, make it say something else
Sign the message so it can not be altered

Availability: They could throw the message away or never deliver it
Send multiple messengers (only need 1 out of N honest messengers)

Trust minimization for digital money

We are going to look at a series of examples of digital money that go from completely trusted → to less and less trusted.

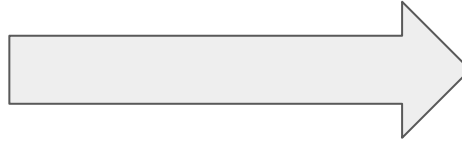
Trust Minimization

If cryptography is the science of minimizing trust
...then is a cryptocurrency a trust minimized currency?

“FIDES EXERCITUUM”
TRUST* THE ARMY



Roman Denarius, 68 AD
Civil War Era (27 BC – AD 215)

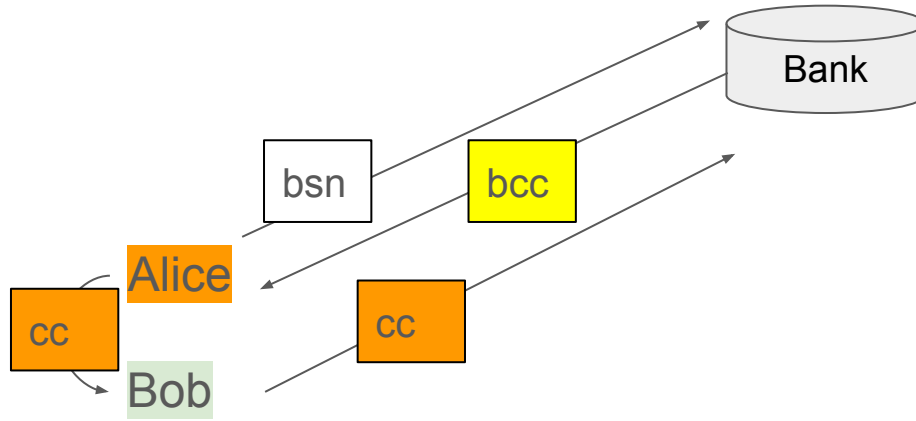


“VIRES IN NUMERIS”
STRENGTH IN NUMBERS



Fake metal Bitcoin, 2011 AD
Post-Cold War Era (1991 AD - ?)

eCash



<u>Starting balances</u>		<u>New balances</u>
16	-1	15
4	1	5

eCash bank issues and redeems coins

Is eCash the first cryptocurrency? I'd argue yes because it uses cryptography, Blind Signatures, to minimize trust in the bank for privacy

eCash

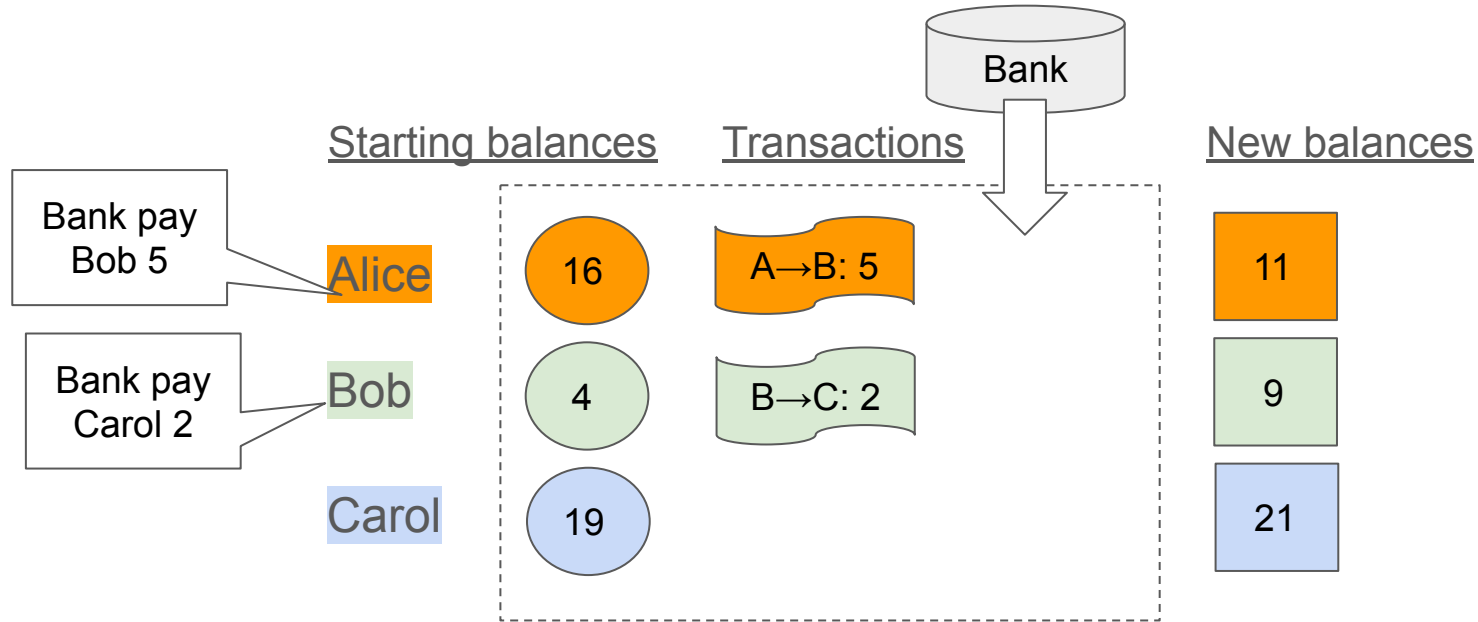
Think about all the bad stuff the eCash bank can do:

1. Inflate the currency without anyone noticing
2. Change balances without detection
3. Disappear/Go offline (all balances lost)
4. Allow doubling spending

...

If we give up privacy we can solve these three problems with a public bulletin board

Bank + Public Bulletin Board

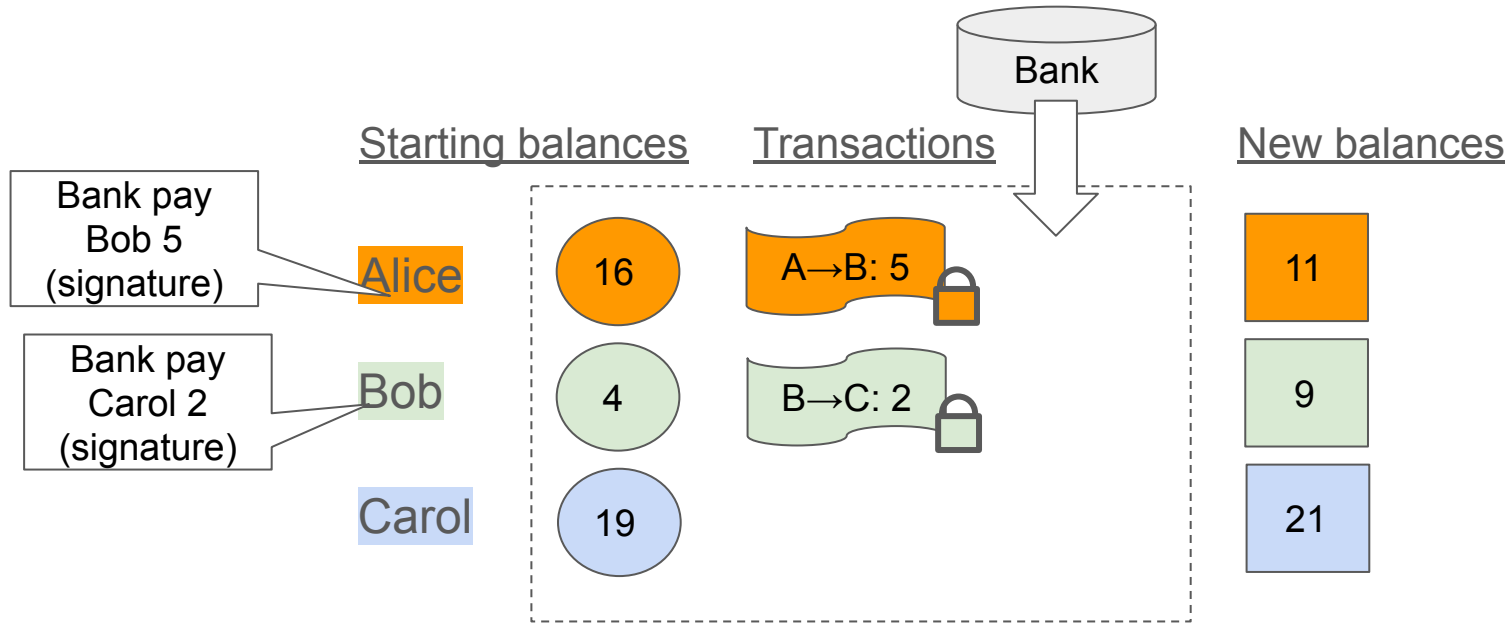


Trust Minimization

Bad things the bank can **still** do:

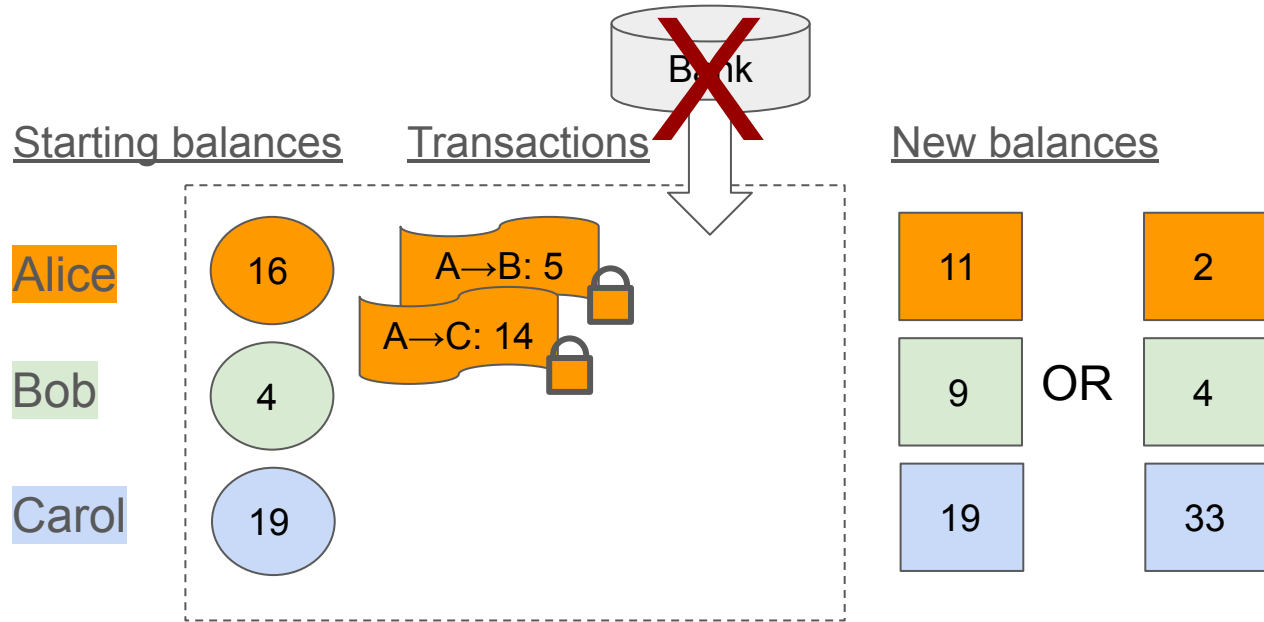
- Create fake transactions that the users never asked for
- Show inconsistent states (reorder transactions maliciously)
- Go offline forever so no one can transact
(but we can replace the bank since we know the balances)
- Censor transactions
- Anything else?

Bank + Signed transactions



We can prevent the bank faking transactions by requiring signatures
... do we still need the bank???

Double spending



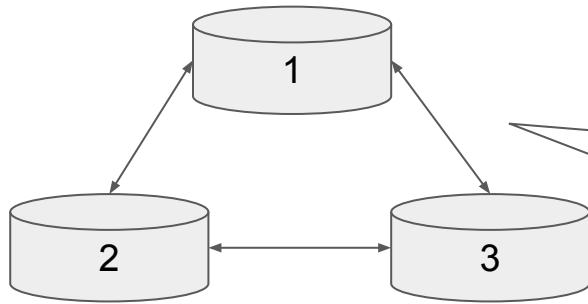
What if we just let parties post their own transactions?

We need a way to order transactions

Ordering transactions without a central party

One option: Have a fixed number of known parties vote.

This is actually harder to build than it sounds (future lecture).



2 of 3 of us say the next transaction to add to the public bulletin board is **A**→**B**: \$4

What happens if the fixed number of parties all go offline or become malicious?

Maybe let anyone vote, but how to prevent ballot stuffing? Sybil attacks?

PoW (Proof of Work): Vote by solving puzzles with computation power

PoS (Proof of Stake): Vote with your money

Proof-of-Work (PoW): one-CPU-one-vote

The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. **Proof-of-work is essentially one-CPU-one-vote.** The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto

Proof of Work (PoW): Vote by solving puzzles

1. Take 256 coins and number them 1 to 256



2. Put them in a jar and shake them up and then dump them on table
3. Record if a coin was heads or tails for each number from 1 to 256

$1 \rightarrow H, 2 \rightarrow T, 3 \rightarrow T, 4 \rightarrow H \dots = HTTH\dots$

The weight is the number of heads in a row at the beginning

THTHH... has a weight of 0

HTHTH... has a weight of 1

HHHHT... has a weight of 4

How many times would have to do this to get a weight of 58?

Proof of Work (PoW): Vote by solving puzzles

How many times would have to do this to get a weight of 58?

$\frac{1}{2}$ probability first coin is heads (T)

$\frac{1}{2}$ probability second coin is heads (T)

$\frac{1}{2} * \frac{1}{2} = \frac{1}{4}$ probability first and second coin is heads

$(\frac{1}{2})^{58}$ probability of getting at least 58 heads at the beginning

You need to repeat this $2^{58} = 288,230$ trillion

Use cryptography to do deterministic coin flipping.

Output \leftarrow Hash(input)

Cryptographic Hash Functions

Hash(Full text of Hamlet) → 256-bit output

Hash functions:

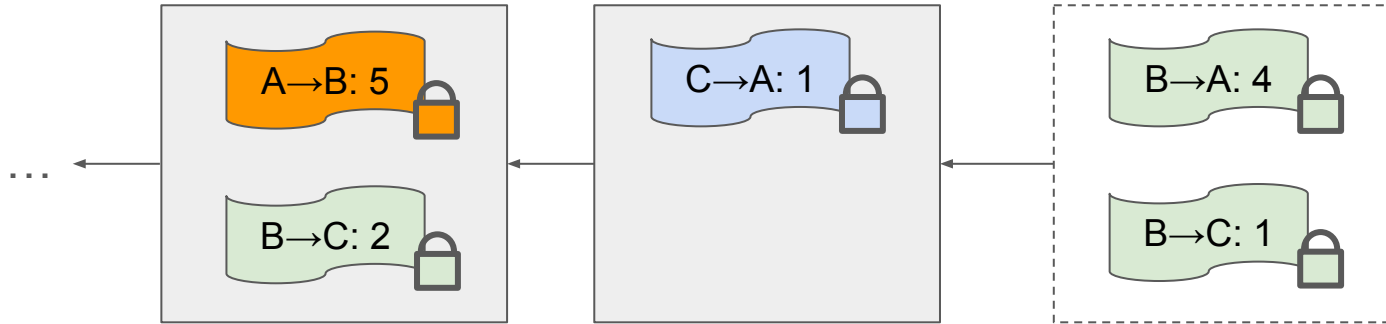
- Takes input of any length (books, programs, your name)
- Produces output of fixed length (256-bit, 43 letters in base64)
- Output is random but deterministic (you get same answer)

To be covered in greater detail in future lectures

Proof-of-Work (PoW): Vote by solving puzzles

$\text{HASH}(\text{prev_block}, \text{B} \rightarrow \text{A}: 4, \text{B} \rightarrow \text{A}: 1, \text{random number})$

Find an output that starts with with a 58 of zeros




Proof-of-Stake (PoS): Vote with money (coins)

Allow parties like Alice, Bob and Carol
to stake their coins for the right to participate in consensus.

To be covered in greater detail in future lectures

Trust minimization for digital money

Trust assumption	Remove trust with...
To not secretly inflate currency supply	Public bulletin board
Creating transactions honestly	Signatures on transactions
Ordering transactions/Consensus (preventing double spending)	Decentralized ordering PoW (Proof of Work), PoS (Proof of State)
Don't go offline forever	Lots of miners, anyone can be a miner
Don't censor transactions	Lots of miners, anyone can be a miner
Privacy	??? Lots of approaches 

TumbleBit (uses Blind Signatures), zCash (uses ZKPs), ...

Trust minimization for digital money: Incentives

Additional questions we will looking at this semester:

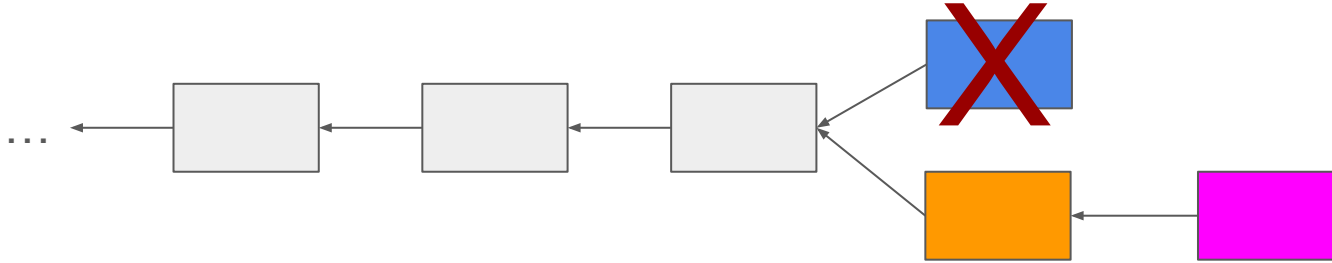
- How can the system reward parties for produce blocks (mining)?
 - Fees, block subsidy
- How can the system reward honest behavior and punish dishonest behavior?
- How can we ensure we don't produce incentives for dishonest behavior?

Bitcoin details

To ground our discussion
let's look at some more specific details of how Bitcoin works.

Additional notes on Bitcoin: Bitcoin's blockchain

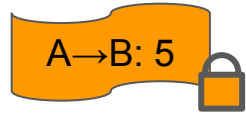
What happens if two miners create two blocks at the same time?



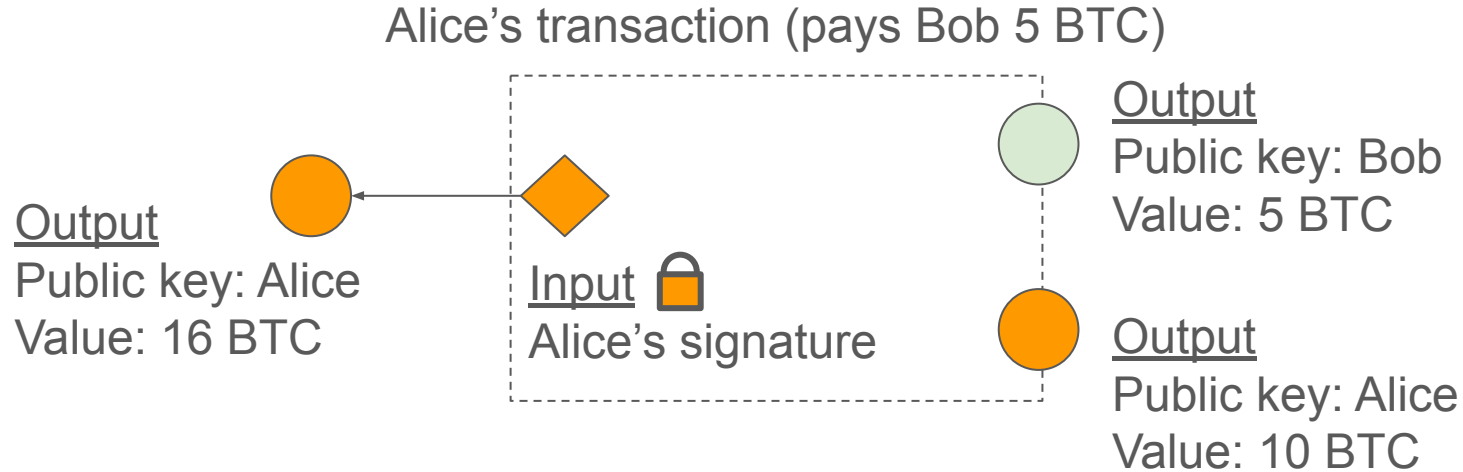
The next block mined builds on one of the two previous blocks

..but why?

Additional notes on Bitcoin: Outputs and Inputs



Bitcoin transactions spend outputs to create new outputs



$$16 - (10+5) = 1 \text{ BTC fee}$$

Additional notes on Bitcoin: Outputs and Inputs

dc0a44c7054855f60f4632a546b5a03e8fd09af15fae2fdb45a1d9d8d2f49513

Details +

#0 40fbb96f76249f3b9bb24664c55829e61785d0d512a3 38f3f57fce408e6b9368:1 0.19380376 BTC

>

#0 17Tz3Dkbr48U3VGFqi6yUXWoSWnQ2hjsdV 0.01054072 BTC

#1 bc1q9r00w029jvwIze2sv2fpves4nuwg9kpqhz6dz2 0.18297504 BTC

1 CONFIRMATION 0.19351576 BTC

<https://blockstream.info/block/000000000000000000000000ef38f53b488e99aa85684cf9acc49971283f7e24b17c>

Reminders & Next Week & Questions

- Reminder:
 - Sign up for the class Discord
 - Register for the class if you haven't yet
 - Source of truth <https://github.com/mit-dci/cde-2025>
- Next week I will be covering (it will be fun):
 - Cryptographic hash functions,
 - Commitments,
 - Merkle trees,
 - Digital signatures
- Questions?

The End

