EECS 485 Blockchain Guest Lecture

2020-07-27

Tadge Dryja
MIT Digital Currency Initiative

Intro

Hi Everyone

Was at UVA with the esteemed professor of this course, so giving a remote guest lecture!

Please ask questions! I don't know how much you know.

Intro

My background:

2011, Working at a Mie University in Japan. Saw Bitcoin paper, said whoa.

2013 went to UVA to learn more

2014 move to San Francisco to work at Bitcoin startups

2016 started at MIT DCI

past & current work

2015 Bitcoin Lightning Network

2017 Discreet Log Contracts

2019 Utreexo (currently implementing this, github.com/mit-dci/utreexo)

will mention these later

cryptographic currency

2009: Bitcoin (paper 2008)

Later, lots of altcoins, some interesting like Ethereum, Zcash, Mimblewimble, etc.

Will mostly explain how Bitcoin works

It does seem to work!

cryptographic currency

Building blocks / lecture layout:

Transactions (move money)

Signatures

Time keeping mechanism (blockchain)

  Proof of Work

Peer to peer network

building blocks

Transactions / accounts

Signatures

Time keeping mechanism (blockchain)

- Proof of Work

Peer to peer network

What's money

Actually a pretty out there question

From an engineering perspective though:

It's a map[string]int that gives allows trade complexity O(n)

barter is $O(n^2)$

Accounts and transactions

Key/Value store:

  Key: who, value: how much

That's enough for a static ranking, but we also want to spend money, so we need transactions

At the simplest, source, destination, amount.

Accounts and transactions

In the actual system, a bit more complex

Multiple sources, multiple destinations (addresses)

Script / smart contract / conditional payments

Need the sources to authorize the transfer with signatures

building blocks

Transactions / accounts

Signatures

Time keeping mechanism (blockchain)

- Proof of Work

Peer to peer network

What’s a signature?

Signatures are useful! Messages from someone. 3 functions needed:

GenerateKeys()

Sign(secretKey, message)

Verify(publicKey, message, signature)

3 functions

GenerateKeys()

Returns a privateKey, publicKey pair

Takes in only randomness

3 functions

Sign(secretKey, message)

Signs a message given a secretKey.

Returns a signature.

3 functions

Verify(publicKey, message, signature)

Verify a signature on a message from a public key. Returns a boolean whether it worked or not.

RSA

Mentioned in Lecture 5: Encryption

RSA gives both encryption and signatures.

Not used in Bitcoin (or any currency)

(also please don't use it!)

RSA

Basic setup: make 2 primes: p, q
n = p*q

Given p, q computing n is easy.

Given n, finding p, q is hard!

A one way function… but not a hash function.

RSA

Can do some fun math with this.

Set e = 3 (or 65537)

set d = some number you can compute if you know p or q.

$d = e^{-1} \bmod (p-1)*(q-1)$

n is public. d is private.

p, q not needed after setup. e always the same

RSA

Sign: $s = m^d \bmod n$

Verify: $s^e \bmod n == m$

Can sign many times. And do lots of cool stuff.

RSA

RSA key sizes are smaller than hash based signatures; often 2048 bits (256 bytes)

Tricky to implement! Lots of ways to lose your private key

but Bitcoin (& other coins) uses elliptic curve signatures

Elliptic Curve signatures

Smaller, safer than RSA

Public keys are a point on a curve, can be represented in 32 bytes

Signatures are another point and a hash value, so 64 bytes

Make the identities in the system EC keys - pseudonymous accounts

building blocks

Transactions / accounts

Signatures

Time keeping mechanism (blockchain)

- Proof of Work

Peer to peer network

Time keeping

With just transactions and signatures, we can't make a usable currency: Double spends.

Alice has 10 coins. She makes 2 simultaneous transactions:

Alice to Bob, 10 coins.
Alice to Carol, 10 coins.

Time keeping

We need everyone to agree on what happened first. Then first transaction happens, later transactions are invalid.

How to get everyone to agree on message ordering?

Central server?

Blockchain

Decentralized, peer to peer time stamping system: a blockchain.

Uses "proof of work" to make it costly to produce conflicting time stamps

In Bitcoin, proof of work also generates new bitcoins

Proof of Work: Hash functions

Any size input, fixed output… output is “random” looking

What’s that mean? Deterministic, no randomness
But the outputs look like noise;
half the bits are 1s, half are 0s

Example Proof of Work

You can try these at home

```
tadge@computer:~$ echo "Tadge 4233964024" | sha256sum

000000007e9f5bb5a25b6a0d1512095bd415840a94e2f2fe93386898947dcb07  -
```

Example Proof of Work

000000007e9f5bb5a25b6a0d1512095bd415
840a94e2f2fe93386898947dcb07 -

That's 33 bits of 0s in front of the hash output. Weird right? What are the odds?

Well, 1 in 2^{33}, or 1:8,000,000,000

partial collision work

increases costs of equivocation / Sybil resistance

scalable:

O(n) work takes O(1) space to prove and O(1) time to verify

why work? to keep time

Big new idea in Bitcoin 9+ years ago:

Use chained proof of work as distributed time-stamping

Achieves consensus on message sequence

Solves double spend problem

block chain

message m, nonce r, target t

hash(m, r) = h; h < t

block chain

message m, nonce r, target t

hash(m, r) = h; h < t

$m_n = (data, h_{n-1})$

e.g $m_2 = (data_2, hash(data_1, r))$

block chain

block has: previous hash

current message

nonce (for work)

block chain

hash all block data

= block hash

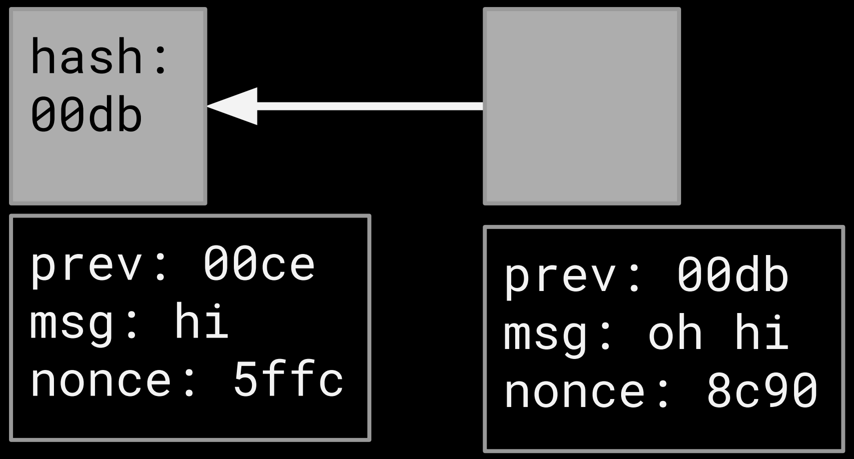
use as identifier

```
hash:
00db
```

```
prev: 00ce
msg: hi
nonce: 5ffc
```

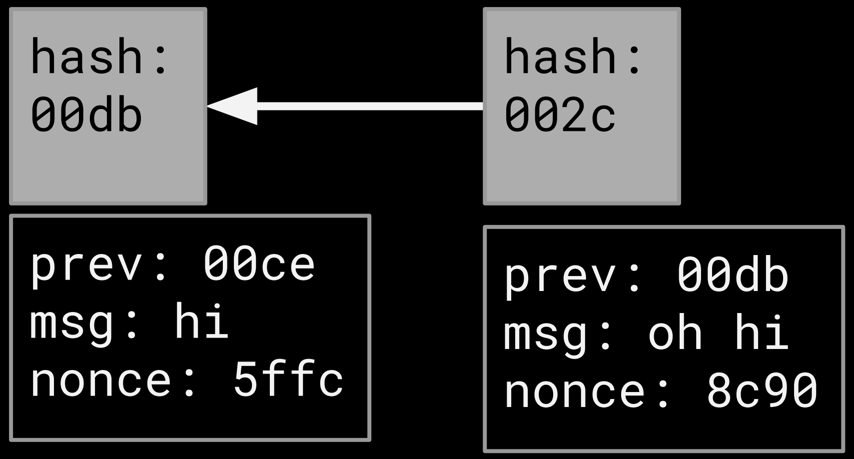
block chain

next block includes hash of last block



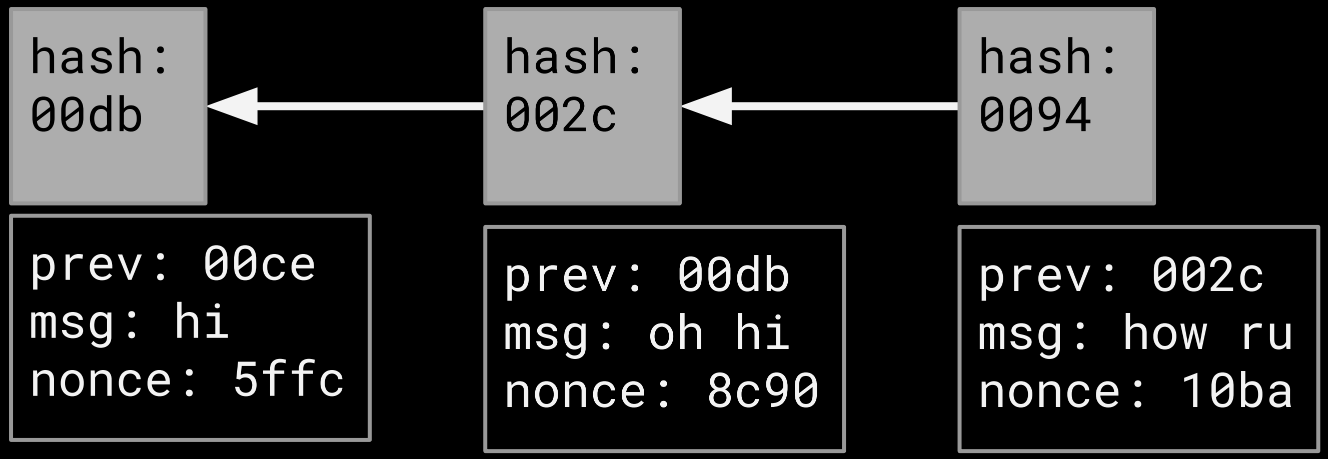
block chain

iterates through nonces so



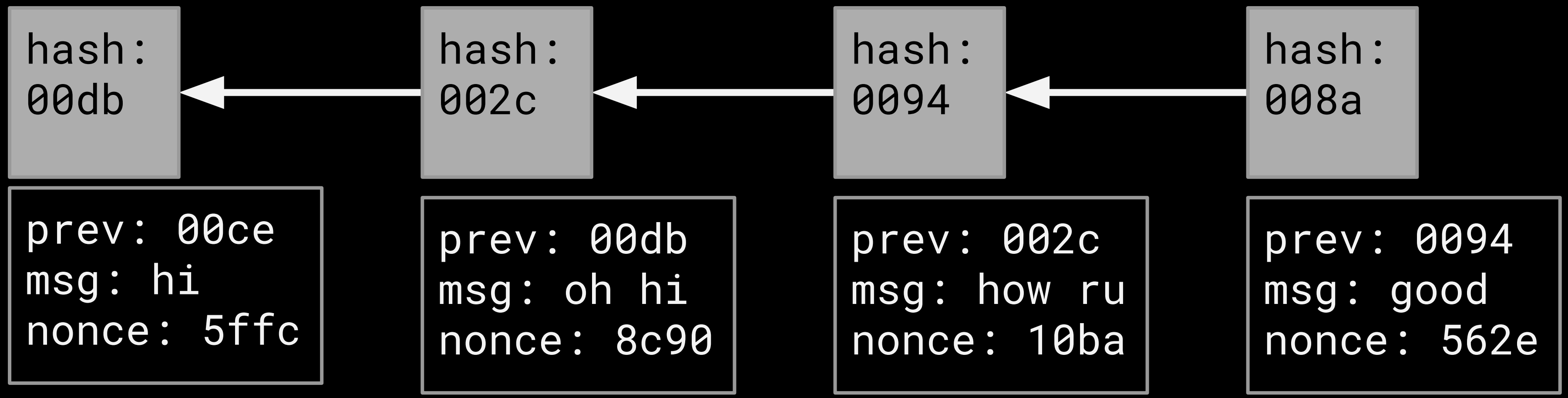
block chain

chain keeps building
adding work each time



block chain

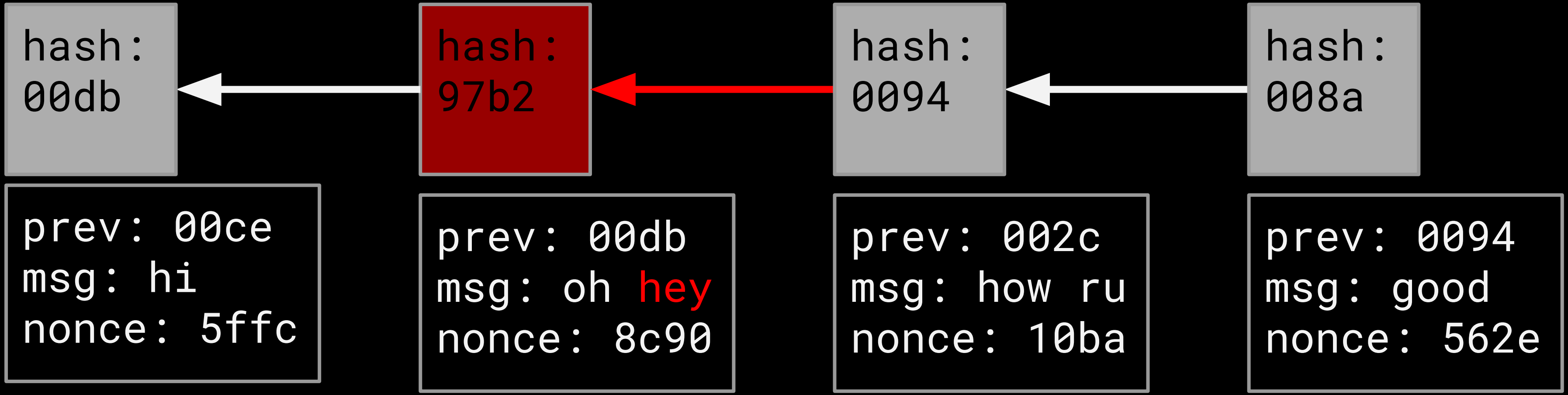
flip any bit in any block . . .



block chain

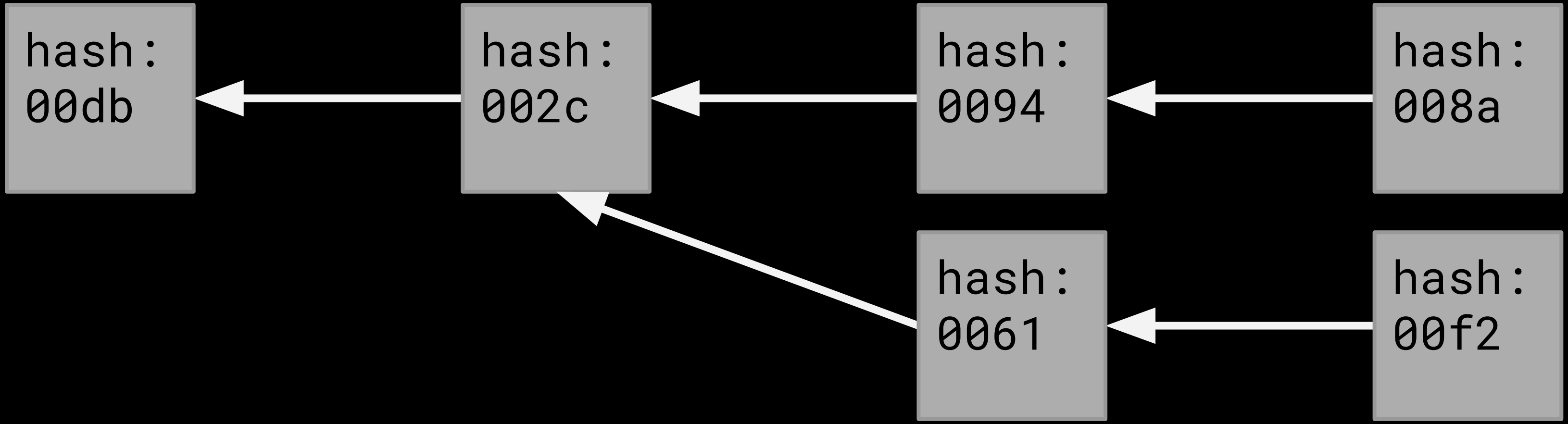
flip any bit in any block

and the chain is broken



chain forks

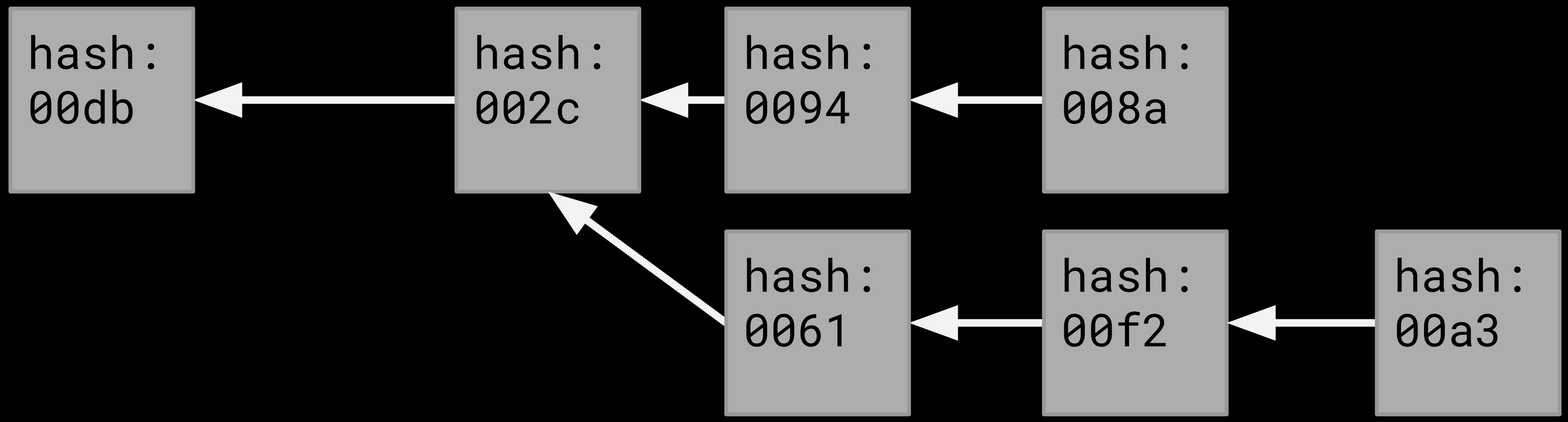
can have two branches at a given height (number of blocks from origin)



chain forks

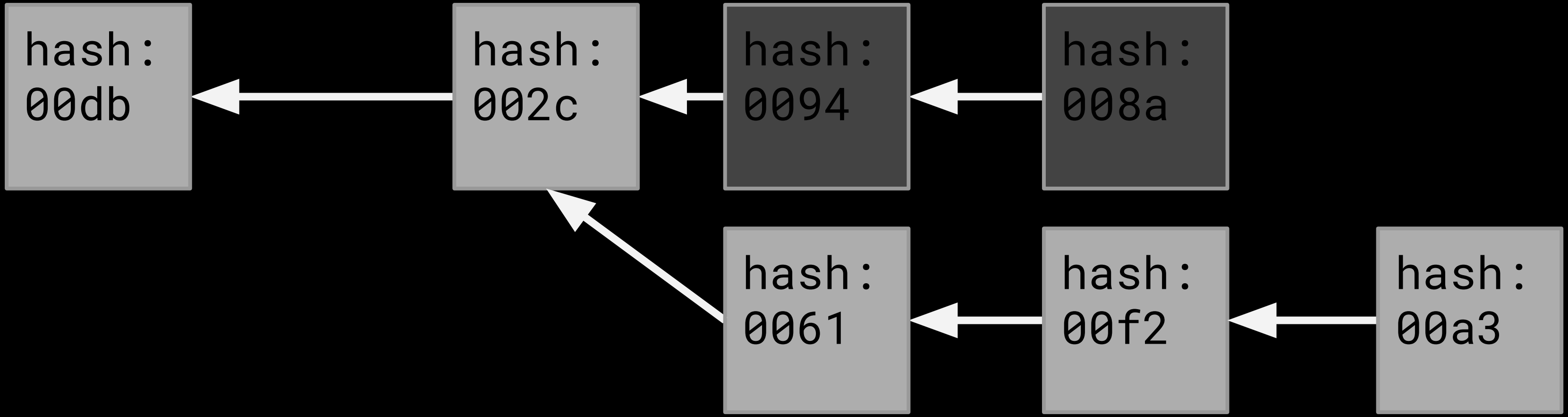
Highest (most work) wins

Everyone uses chain with most work



chain forks

Less work chains can be discarded after the fact. "Reorg"



pros & cons of PoW

pro:	con:
anonymous	~all nonces fail
memoryless	uses watts
scalable	uses chips
non-interactive	51% attacks
tied to real world	people hate it

pros: anonymous

no pre-known key / signature

anyone can go for it

all attempts equally likely

not limited to humans

pros: memoryless

no progress. 10T failed nonces, next nonce just as likely to fail

Poisson process: always expect next block in 10 min

2X attempts / sec means 2X chance of finding next block (linear)

pros: scalable

000000000000000000003bd84b989235a304590bc9a127c97a2c2226ee51302258

Look at those 0s! 18 of em! 9 bytes!

(Seriously, that is 10^{22} attempts. Almost a mole.)

Takes just as long to check as with the 4 bytes of my name & nonce

But it's 2^{40} times more work!

(that's 1 trillion times more)

cons: ~all nonces fail

"inefficient" - almost all attempts fail. That's no fun.

2^{72} attempts needed? You will ~never find a valid proof.

Granularity is high; small players pushed out of the game

cons: uses watts & chips

Lots of electricity

Could use that to charge your car

Uses fabs, which could make more CPUs

Affects markets: GPU prices

Someday could affect electric prices

cons: 51% attacks

Anonymous: don't know who's got hash power. Maybe an attacker!

Attacker with 51% of total network power can write a chain faster than everyone else

Attacker can potentially rewrite history!

cons: people hate it

Not a quantitative / objective reason, but lots of people really don't like proof of work.

"The whole point of sha256 is you can't find collisions!"

"Wastes so much electricity"

"Totally pointless computation!"

Fun facts

How to estimate total work done in the network?

Just look at lowest hash

Can prove total work ever with 1 hash

Can prove close calls as well to other people and show you're working

building blocks

Transactions / accounts

Signatures

Time keeping mechanism (blockchain)

- Proof of Work

Peer to peer network

Connectivity

"gossip" network, where you forward messages even if you didn't originate them

Ideally messages get around to everyone in a few seconds

More blocks, more problems

Functionality

Privacy

Scalability

Smart Contracts on Bitcoin

Discreeet Log Contracts:

Elliptic curve tricks to allow a 3rd party to obliviously activate transactions from a pre-signed set.

Privacy

Everything's public! Not human names, but still a problem.

Like buying things at a convenicence store with $100 bills. Except the bill can be all the money you have.

Privacy

Coin join / shuffle / swap

Single transaction with many participants, unclear whose money went where

Privacy

Confidential Transactions

Fancy cryptography to make operations verifiable but values encrypted.

Alice has X coins, sends Y to Bob and Z to Carol.

Prove X = Y+Z, don't reveal X, Y, Z.

Scalability

Lightning Network

Instead of one-off transactions, open "channels" between users

Update channels to make payments

Can send payments through a network of channels

Scalability

Utreexo

Don't store the key:value set of who owns what. Just keep a hash of it!

Then when people spend money, they prove it exists.

Data storage goes from gigabytes to less than a kilobyte O(log(n))

Lots of research & development

Crypto-currency, blockchain, whatever you call it, lots of interesting things going on!

Also lots of scams. It's like 99% scams. That 1% though is great!

Outro & Questions

contact info: adiabat on github, IRC freenode (#utreexo) @tdryja twitter, tdryja@media.mit.edu

Any questions about the wild world of cyber-coins, feel free to ask!