mas.s62
lecture 2

2018-02-12
Tadge Dryja

cost

How to prevent sybil attacks?

Hard problem! Arms race; Twitter / FB / etc have tons of bots

Also, don't want anyone in charge

rules out SSN, phone num, captchas

work

pset01 (how's that going) needs many attempts to forge a signature

if hash functions have random output, then there's no shortcut. We know what we want, but only way to get it is to keep trying.

what do you want out of work?

time consuming - like homework, but hw has problems. Trusted setup

deterministic verification

scalable - O(1) to verify

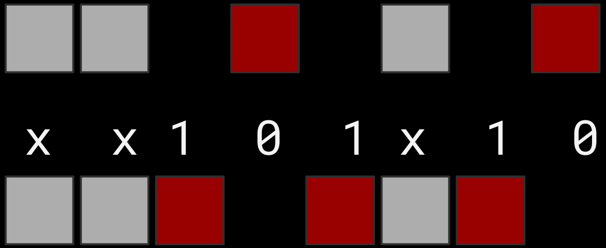
memoryless - everyone gets a chance

homework work

in this case, 5 bits of 8 are constrained

need to try 2^5 messages to find a forgery

must match:

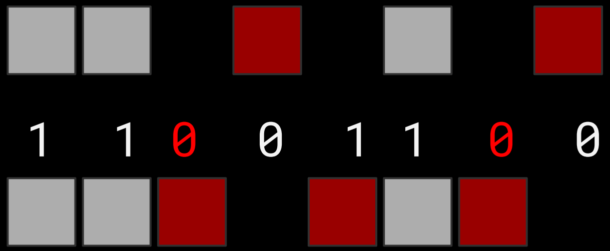


homework work

in this case, 5 bits of 8 are constrained

need to try 2^5 messages to find a forgery

must match:

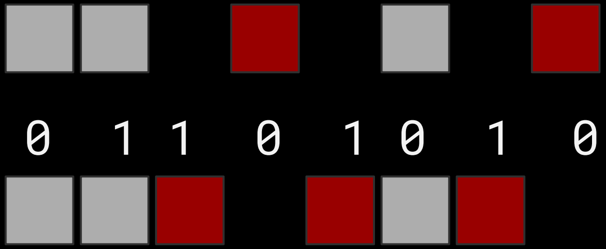


homework work

in this case, 5 bits of 8 are constrained

need to try 2^5 messages to find a forgery

must match:



collisions are work

"Tadge forge 1 154262107" is a message that can be forged given the 4 signatures in the pset

Did I do 154M attempts (work)?

Maybe I did more

Maybe I got lucky and started at 150M

"proof" of work

Maybe not quite a proof; for one proof, lots of chance

For many proofs, averages out

Must estimate collision difficulty;

in this case 2^{31} = 2GH

in this case, the "1" is thread number; 8 threads, so 1.2G, which is pretty good luck but within 2X

simpler proof of work

Signature collision is complex

Has specific target

Some kind of "universal" work

Collide with a fixed string

Collide with… zero?

hashcash

1997, idea was to stop e-mail spam

put a nonce (that's the 154262107 number) in the mail header, try to get a *low* hash output

partial collision with 0

simpler proof of work

```
$ echo "Tadge 4233964024" | sha256sum
000000007e9f5bb5a25b6a0d1512095bd415840a94e2f2fe93386898947dcb07
```

That's 8 zeros! 4 bytes. 2^{32}

I'm a hard worker. Will put this on my resume.

partial collision work

increased 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})

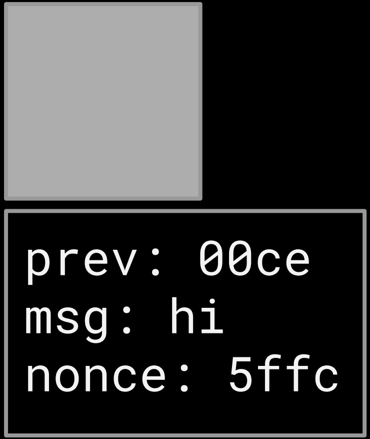
e.g m_2 = ($data_2$, hash($data_1$, r))

block chain

block has: previous hash

current message

nonce (for work)



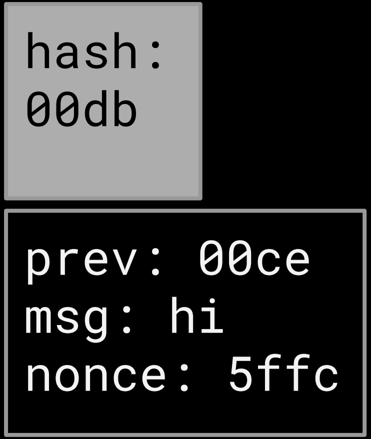
```
prev: 00ce
msg: hi
nonce: 5ffc
```

block chain

hash all block data

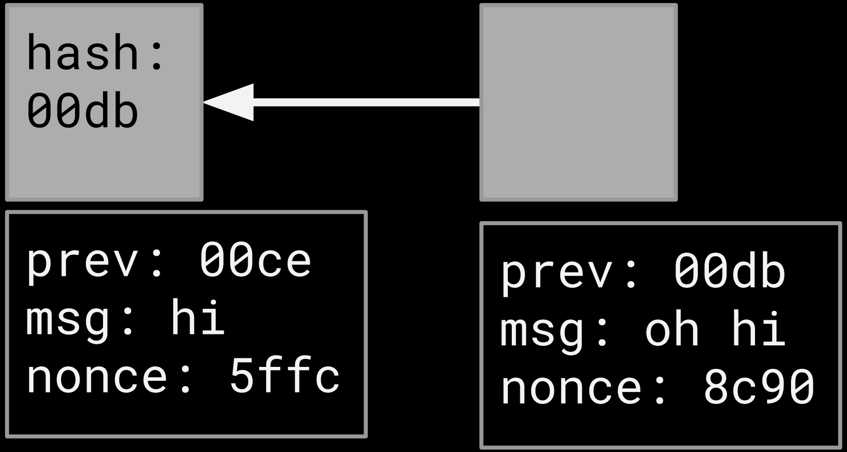
= block hash

use as identifier



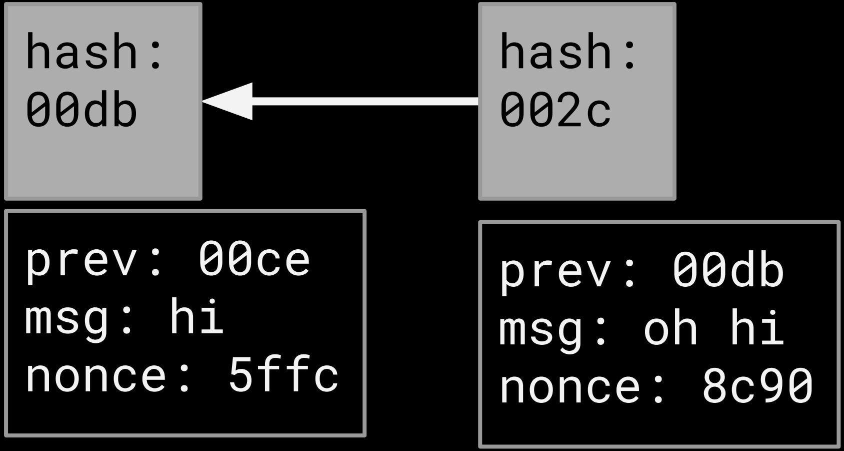
block chain

next block includes hash of last block



block chain

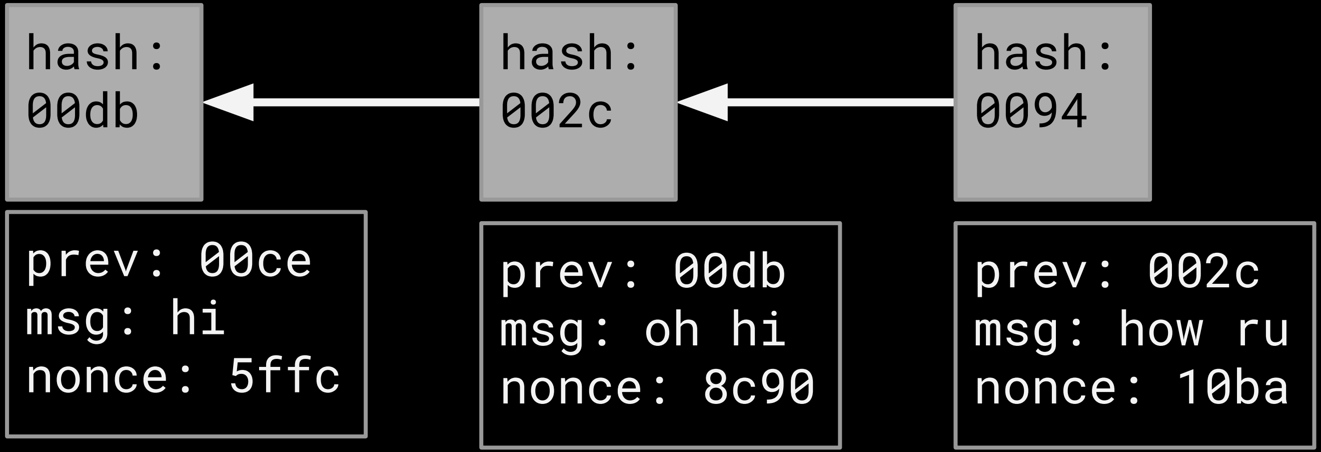
iterates through nonces so



block chain

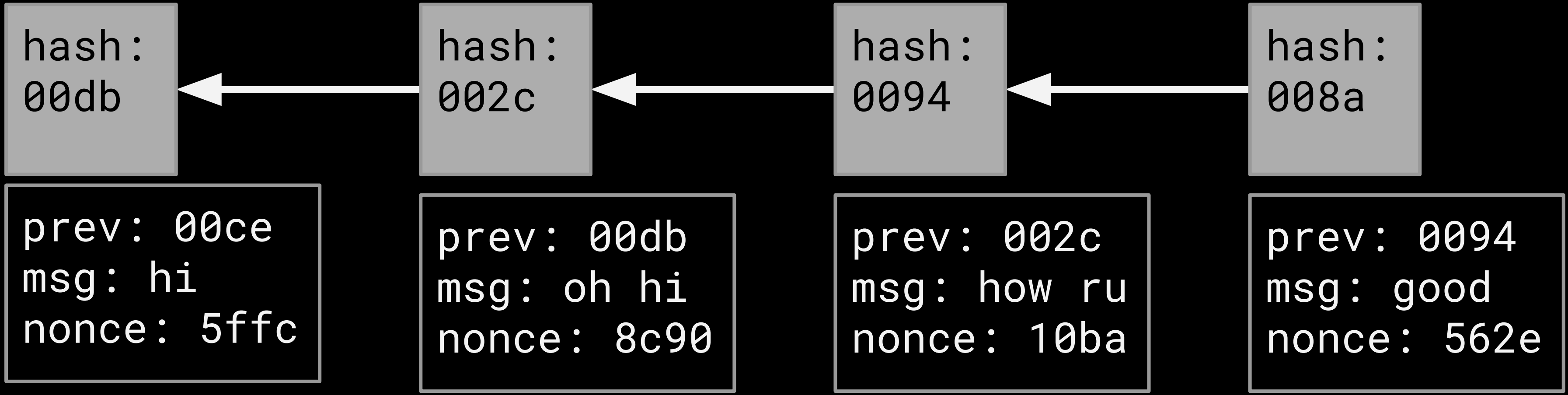
chain keeps building

adding work each time



block chain

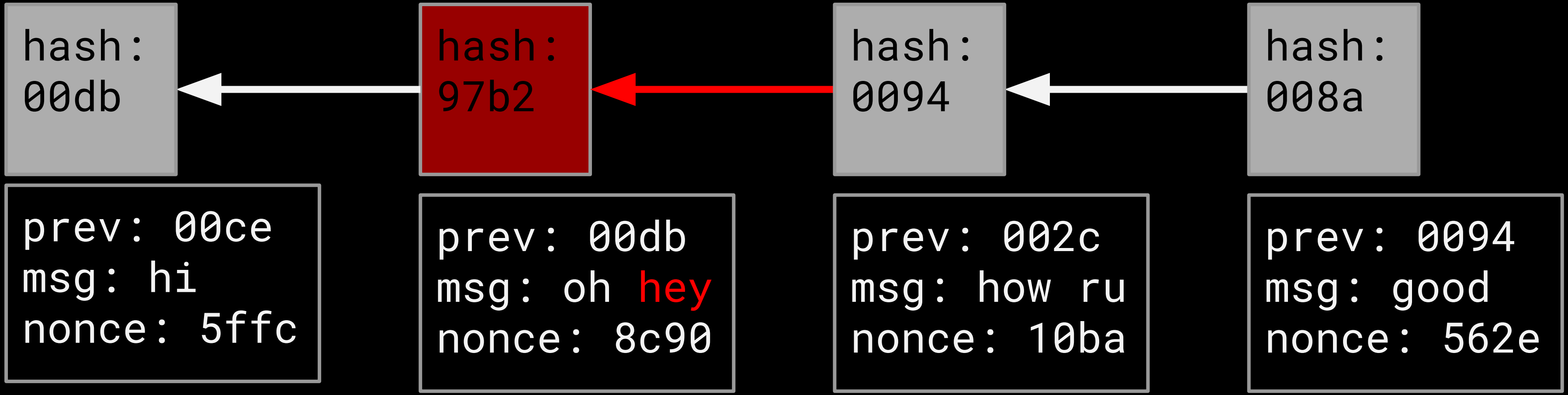
flip any bit in any block . . .



block chain

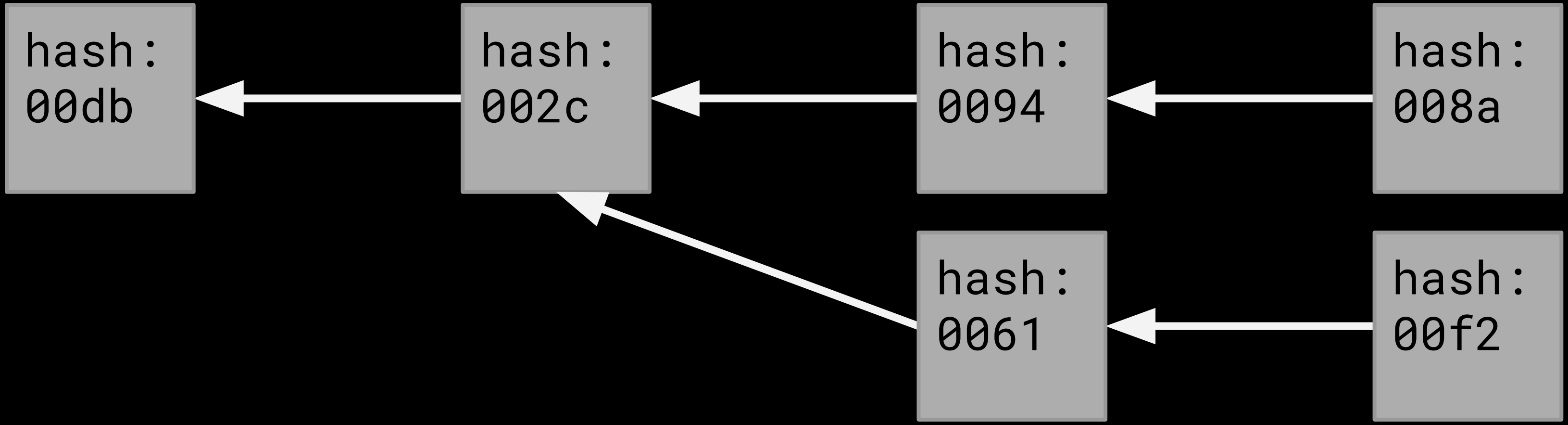
flip any bit in any block

and the chain is broken



chain forks

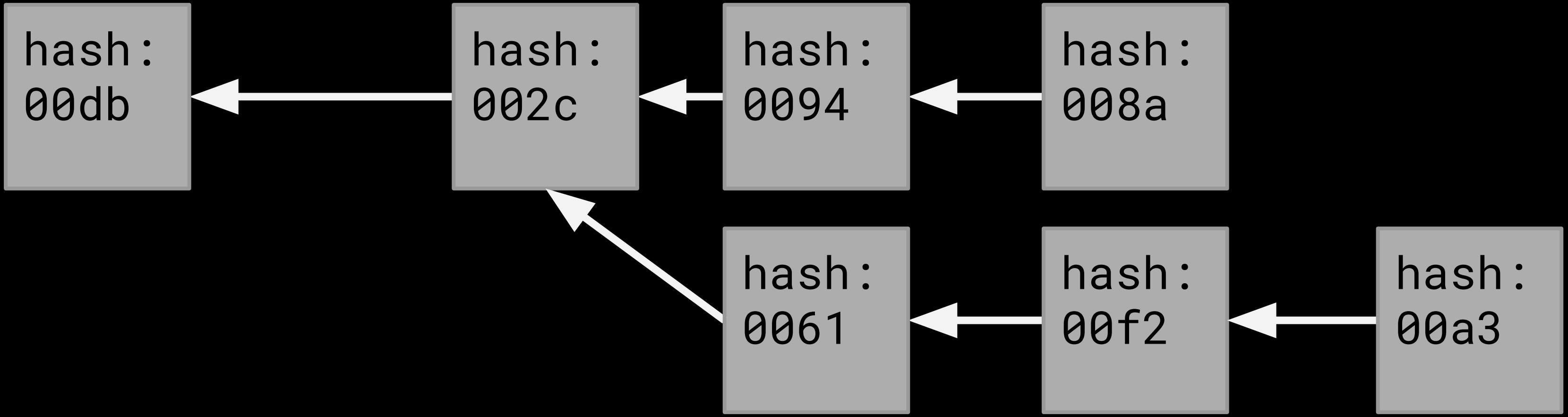
can have two branches at a given height (number of blocks from origin)



chain forks

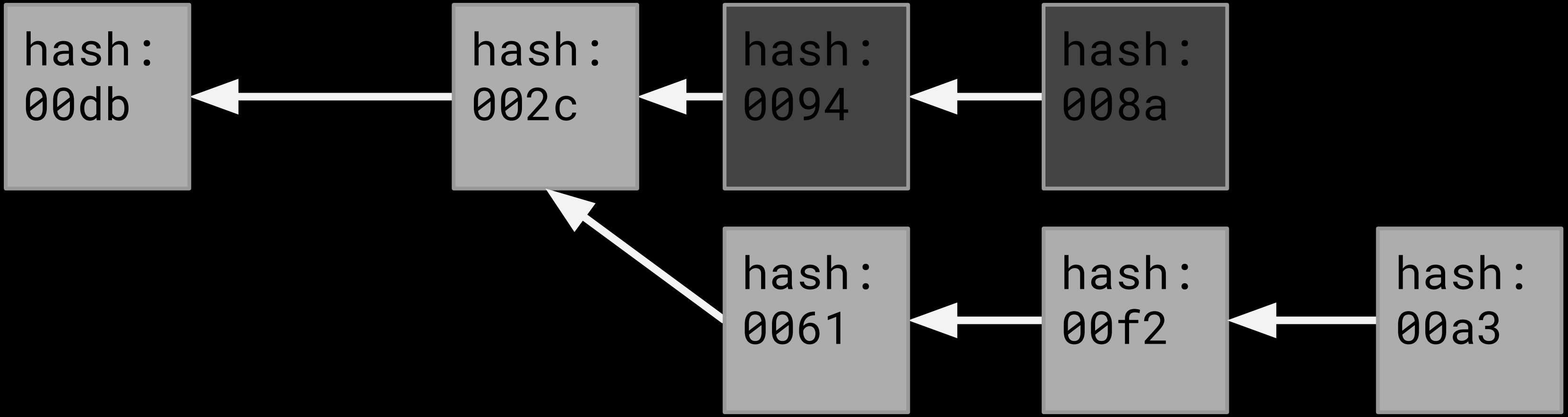
Highest (most work) wins

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

```
0000000000000000003bd84b989235a304590bc9a127c97a2c2226ee51302258
```

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)

pros: non-interactive

1000 chips all trying once

or 1 chip trying 1000 times

equal chances; only communication is when a block is found

pros: real world resources

Like a captcha (turing test)

Prove usage of real world resource

Can't get that time / energy back

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: irregular

Poisson process means sometimes a solution is found in a few seconds

Sometimes it takes an hour

Can deal with it but annoying;
Precludes some use cases

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!"

Proof of Work: it Works

It's been working for 9 years

Blocks keep coming

In practice, infeasible to re-write old messages; tons of work on top

Bitcoin: very few block reorgs (rewrites), most 1 or 2 blocks deep

Next: build on it!

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