mas.s62
lecture 23
New Directions in Crypto

2018-05-07

Tadge Dryja

today

future developments:

- block / committed bloom filters
- sharding
- accumulators
- UTXO commitments

block filters

first: what is a bloom filter

  makeFilter([]obj) -> filter

  matchFilter(filter, obj) -> bool

  can have false positives but not false negatives

block filters

current SPV model

- client makes filter of all their utxos and addresses
- sends filter to server
- server matches filter w/ each block
- server sends only matching txs

block filters

current SPV model

- bad for privacy
- sending filter, not utxo / adr list
- but nearly the same effect
- slow for servers

block filters

new(ish) idea: reverse this model

server makes filter from txs in block

client requests filter

client matches fitler to own utxos

client requests whole block on match

block filters

better privacy: server only learns which blocks interesting to client

low CPU use for server

harder to lie / omit (?)

higher network traffic for client
current development: "neutrino"

sharding

mainly worked on for Ethereum

common from database world:

- d data, n servers
- don't store d*n, store ~d, and shard data over all servers, so each holds (lim) d/n data

sharding

difficult in blockchain / consensus / adversarial environment

need to prevent spending invalid coins
split single utxo set into multiple smaller shards
need swaps between shards

multicoin vs shards

multiple utxo sets is what we've got!

Is this "sharding"?

multicoin vs shards

multiple utxo sets is what we've got!

Is this "sharding"?
want more than just swaps; need fungibility between shards
real scalability improvement (if it works!)

Accumulators

cryptographic sets
inclusion / exclusion proofs

add(accum, obj) -> accum

del(accum, obj) -> accum

prove(accum, obj) -> bool

Accumulators

cryptographic sets
inclusion / exclusion proofs

add(accum, obj) -> accum

del(accum, obj) -> accum

prove(accum, obj) -> bool

simple example: composite numbers

prime accumulator

accumulates primes. To "add", multiply. To "delete", divide.

add(3, 5) -> 15

prime accumulator

accumulates primes. To "add", multiply. To "delete", divide.

add(3, 5) -> 15

add(15, 7) -> 105

prime accumulator

accumulates primes. To "add", multiply. To "delete", divide.

add(3, 5) -> 15

add(15, 7) -> 105

del(105, 5) -> 21

prime accumulator

accumulates primes. To "add", multiply. To "delete", divide.

add(3, 5) -> 15

add(15, 7) -> 105

del(105, 5) -> 21

prove(21, 7) -> true

RSA accumulators

constant size accumulator, proofs

efficient operations

... but trusted setup

(composite n= p*q with unknown p, q)

other accumulators

some are 1-way (can't delete)
some can be batched, some can't
some have trusted setup

different tradeoffs for use case

utxo vs stxo inclusion

accumulators

great if you could get it working

no more UTXO set, just accumulator

constant size, regardless of set

small proofs; wallets track proofs

accumulators

profs are O(1)? O(log(n))?
n = txs? blocks? aggregation?

transitioning: need some bridge node

actually faster? Bitcoin UTXO set only ~4GB

UTXO commitments

exists in some coins (ETH), not yet in Bitcoin

simplest: take hash(UTXO set), put it in coinbase tx

UTXO commitments

somewhat more useful: Merkle root of UTXO set in coinbase tx every block

Can then "prove" an output exists

UTXO commitments

somewhat more useful: Merkle root of UTXO set in coinbase tx every block

Can then "prove" an output exists

(prove with SPV security)

UTXO commitments

skip years of initial block download!

only verify last ~6 months of txs

if everyone's been wrong for 6 months we have bigger problems, right?

UTXO commitments

issues
timing: adding even 1s in creating / verifying a block centralizes mining

encourages more SPV-level verification (trust the miners)

"there's a better way to do this"
hash based, EC, RSA

more research required

Lots of topics in this area to improve:

- privacy
- scalability
- functionality

MAS.S62 Cryptocurrency Engineering and Design

Spring 2018