today
payment channels

recap

optimizations: key addition,
hash trees
cross chain swaps
## revokable tx

<table>
<thead>
<tr>
<th>Commit Tx (held by Alice)</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
</tr>
<tr>
<td>fund txid</td>
</tr>
<tr>
<td>Bob's signature</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
revokable tx

<table>
<thead>
<tr>
<th>Commit Tx (held by Bob)</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td></td>
</tr>
<tr>
<td>fund txid</td>
<td>Alice address</td>
</tr>
<tr>
<td>Alice's signature</td>
<td>2 coins</td>
</tr>
<tr>
<td></td>
<td>Bob key &amp; 100 blocks or Alice &amp; BobR key</td>
</tr>
<tr>
<td></td>
<td>8 coins</td>
</tr>
</tbody>
</table>
add and delete states

In Lightning, states are added sequentially, and validity is enforced by revealing private keys to previous states.
In Lightning, states are added sequentially, and validity is enforced by revealing private keys to previous states.
In Lightning, states are added sequentially, and validity is enforced by revealing private keys to previous states.
In Lightning, states are added sequentially, and validity is enforced by revealing private keys to previous states.
reveal to revoke
Either party broadcasts & has to wait
Alice gives Bob the AliceR privKey
Bob gives Alice the BobR privKey
Now if they broadcast the counterparty can take all funds while they wait!
preimage or private key
KeyA && time
|| (KeyB && KeyC)
optimizations here?
preimage or private key

KeyA && time

|| (KeyB && KeyC)

KeyC could be a hash/preimage pair, 20 bytes instead of ~70

Even smaller?
Adding keys
Add KeyB and KeyC

B + C = R

what's the private key for R?
Adding keys
Add KeyB and KeyC

B + C = R

what's the private key for D?

bG + cG = rG

(b + c)G = rG

sum of private keys works
reduced script

KeyD || KeyA && time

opcodes:

OP_IF KeyR OP_ELSE

<delay> OP_CHECKSEQUENCEVERIFY

OP_DROP KeyA OP_ENDIF OP_CHECKSIG
reduced script
stack: 1 SigR

OP_IF KeyR OP_ELSE
<delay> OP_CHECKSEQUENCEVERIFY
OP_DROP KeyA OP_ENDIF OP_CHECKSIG
reduced script
stack: 0 SigA

OP_IF       KeyR       OP_ELSE
<delay>     OP_CHECKSEQUENCEVERIFY
OP_DROP     KeyA       OP_ENDIF   OP_CHECKSIG
reveal key, revoke state
need to keep track of old secrets
one for each state
32 bytes each... not great for scaling
hash tree
reveal secrets 1 at a time
store only $\log(n)$ secrets
recompute any received secret
Elkrem

left child: append 0, hash

right child: append 1, hash
Elkrem

left child: append 0, hash

right child: append 1, hash
Elkrem
left child: append 0, hash
right child: append 1, hash
Elkrem
left child: append 0, hash
right child: append 1, hash
Elkrem
left child: append 0, hash
right child: append 1, hash
Elkrem

left child: append 0, hash

right child: append 1, hash
Elkrem
left child: append 0, hash
right child: append 1, hash
Elkrem

left child: append 0, hash

right child: append 1, hash
Elkrem
left child: append 0, hash
right child: append 1, hash
Elkrem

left child: append 0, hash

right child: append 1, hash
Elkrem

left child: append 0, hash

right child: append 1, hash
intermission
0x7f sec to stretch
cross chain
there are altcoins
most of them (used to) work like Bitcoin, as they just copied the whole codebase on github
(see e.g. coingen.io)
some recent coins very different
cross chain
people trade altcoins for bitcoins
they even trade altcoins for altcoins
how to trade? use "exchanges"
coin exchanges
exchange model:
give us all your coins
post orders on our site to swap
ask for your coins back
coin exchanges
exchange model:
give us all your coins
  (this part works fine)
post orders on our site to swap
ask for your coins back
coin exchanges
exchange model:
give us all your coins
post orders on our site to swap
ask for your coins back

(here's where the model tends to fail)
cross-chain swaps
no custody
you get coinA iff I get coinB
use HTLCs just like in lightning network
channels are on different networks
Preimage determines who spends

<table>
<thead>
<tr>
<th>Commit Tx (held by Bob)</th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fund txid, Alice's signature</td>
<td>Alice address: 2 coins</td>
</tr>
<tr>
<td></td>
<td>Bob key &amp;&amp; 100 blocks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HTLC Alice &amp;&amp; R</td>
<td></td>
</tr>
</tbody>
</table>
HTLC construction

\[ H = \text{hash}(R) \]
HTLC construction

Alice → Bob (DOGE)
Bob → Alice (VTC)

H → H, R
HTLC forwarding

HTLC:
Bob && R ||
Alice && 17:00
HTLC forwarding

HTLC:
Bob & & R ||
Alice & & 17:00

Bob

H

HTLC:
Alice & & R ||
Bob & & 16:00

H, R

Alice

VTC

DOGE

H
HTLC clearing

HTLC:
Bob && R ||
Alice && 17:00

reveal R, clear HTLC
HTLC clearing

HTLC:
Bob && R ||
Alice && 17:00
HTLC clearing

reveal R, clear HTLC
HTLC clearing

reveal R, clear HTLC

Alice
H, R

Bob
H, R

Alice
H, R

DOGE

VTC
cross chain swaps
H can be revealed on either chain, so both parties need to watch both blockchains
They have channels on each chain so that makes sense
Receiver doesn't need to be initiator, but probably will be
how to trade
good for trade execution, but what about discovery?
post orders on blockchain?
non-binding, frontrunning, non-scalable
how to trade multiple models:

central orderbook & counterparty exchange is one side of every trade and keeps the spread similar centralization to current custodial model, but less risk
how to trade multiple models:
central orderbook, multiple counterparties

connecting to many counterparties is costly

how to enforce trade execution?
how to trade
multiple models:
distributed orderbook

how to ensure fairness?
how to enforce trade execution?
scalability of orders?
cross-chain swaps
basic idea works, but still many unsolved questions
further research required
people working on this here! (ask)