Bitcoin and Cryptocurrency Technologies
Lecture 9: Bitcoin Scalability

Yuri Zhykin

May 17, 2021

Transaction Throughput

- Block every 10 minutes (600 seconds).
- Every block - 1 Mb.
- Average transaction size - 500 bytes.
- Throughput

$$T = \frac{1000 * 1000}{500 * 600} \approx 3\text{tx/sec}$$

- Visa throughput is approx. 1700 tx/second.

Throughput Scaling Proposals

- **Increase block size** - more centralization; current chain size growth rate is 51 Gb/year.
- **Increase block frequency** - more centralization, less network stability.
- **Optimize transaction structure** - limited possibilities.
- **Second-layer solutions** - the only viable approach?

Transaction Structure Optimization

- **Segregated Witness** (**SegWit**).
- **Schnorr signatures**.
- **Merkelized Abstract Syntax Trees** (**MAST**).
- **Taproot** (both **Schnorr** and **MAST**).

SegWit 1/4

- Protocol upgrade that was activated in 2017 and solved the following problems:
  - **transaction malleability** - for a non-SegWit transaction it is possible to change the transaction in a way that changes the transaction ID (hash) while the signature remains valid;
  - **block space optimization** - no need to store the usually large unlock script in the block - it is moved into a Witness structure;
  - **future upgrades** - SegWit introduced a clean way for upgrading the protocol via **softforks**.

SegWit 2/4

- SegWit followed the idea of P2SH (BIP-0016) - additional script validation rules based on the pattern in the script.
- Main transaction components are now **inputs**, **outputs** and **witnesses** (one **witness** per **input**).
- For every transaction input, if **lock-script** *being executed* is a **witness program**
  - take the **witness** structure,
  - verify that the **witness program** matches the hash of the **witness** structure,
  - interpret **witness** structure as an **unlock-script**.

SegWit 3/4

- **P2WPKH** - pay-to-witness-public-key-hash
  - Witness: `<signature> <pubkey>`
  - Unlock: `;`
  - Lock: `0 <20-byte-key-hash>;`
- **P2WSH** - pay-to-witness-script-hash
  - Witness: `0 <signature1> <1 <pubkey1> <pubkey2> 2 CHECKMULTISIG>`
  - Unlock: `;`
  - Lock: `0 <32-byte-key-hash>;`
- **P2SH-P2WPKH** - P2WPKH nested in BIP16 P2SH
  - Witness: `<signature> <pubkey>`
  - Unlock: `<0 <20-byte-key-hash>>;`
  - Lock: `HASH160 <20-byte-script-hash> EQUAL;`
- **P2SH-P2WSH** - P2WSH nested in BIP16 P2SH
  - Witness: `0 <signature1> <1 <pubkey1> <pubkey2> 2 CHECKMULTISIG>`
  - Unlock: `<0 <32-byte-key-hash>>;`
  - Lock: `HASH160 <20-byte-hash> EQUAL;`

SegWit 4/4

- Witness structures are included in the block chain via the **witness root hash**, which is recorded in a **lock-script** of the **coinbase** transaction.
- **Witness root hash** is the *merkle root* of the *merkle tree* of the **wtxid**s:

  txid: `[nVersion][txins][txouts][nLockTime]`

  wtxid: `[nVersion][marker][flag][txins][txouts][witness][nLockTime]`

- Block size restriction $(1,000,000)$ is changed as follows:
  - *BlockWeight* is defined as $BaseSize * 3 + TotalSize$;
  - *BaseSize* is the block size in bytes with the original transaction serialization without any witness-related data;
  - *TotalSize* is the block size in bytes with transactions serialized as described in BIP144, including base data and witness data;
  - The new rule is $BlockWeight \leq 4,000,000$.

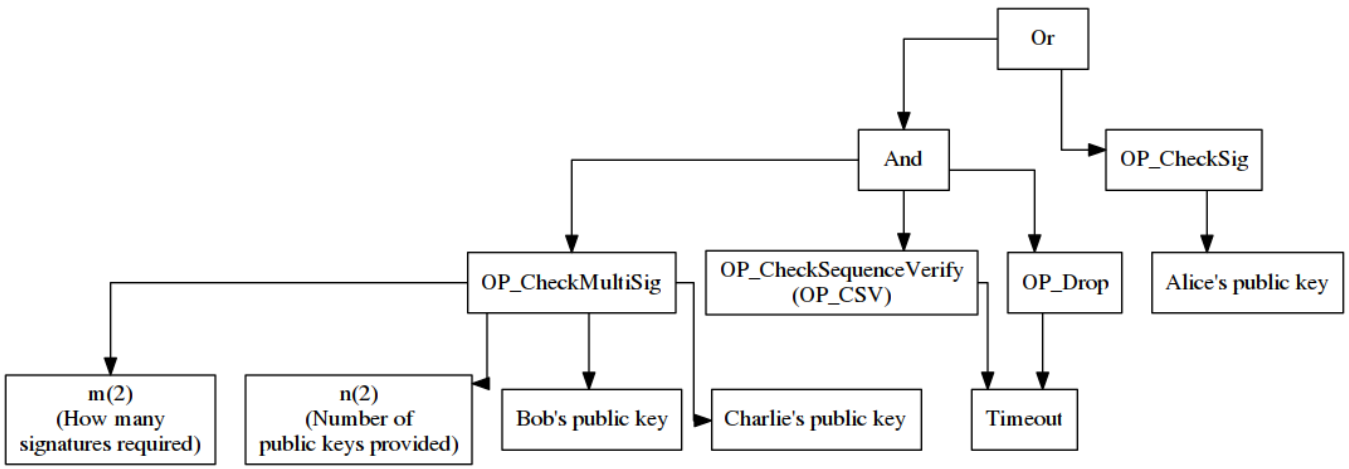
Schnorr Signatures

- **Schnorr signature** is an alternative cryptographic signature scheme that provides certain properties that are desirable for the Bitcoin system, e.g. **linearity** that allows for key/signature aggregation:

$$key_x = key_1 + key_2 + ... + key_n$$
$$sig_x = sig_1 + sig_2 + ... + sig_n$$

MAST and Taproot

- **Merkelized Abstract Syntax Trees** (**MAST**) is an way to support large complex scripts in Bitcoin by building a *merkle tree* from it and only revealing the **tree path** used for unlocking, enhansing privacy.



- Both **Schnorr signatures** and **MAST** are part of the upcoming **Taproot** soft-fork (signalling started May 2, 2021).

Second Layer Solutions

- Perform transactions in a second-layer network and use main Bitcoin network (chain) as a settlement layer.
- Signed Bitcoin transaction is a payment that can be "claimed" by publishing it to the Bitcoin network.
- Second-layer payments can be implemented with signed Bitcoin transactions that are only published when settlement is needed.
- Until **settlement transaction** is published, **double spending** is still possible.

Payment Channels 1/2

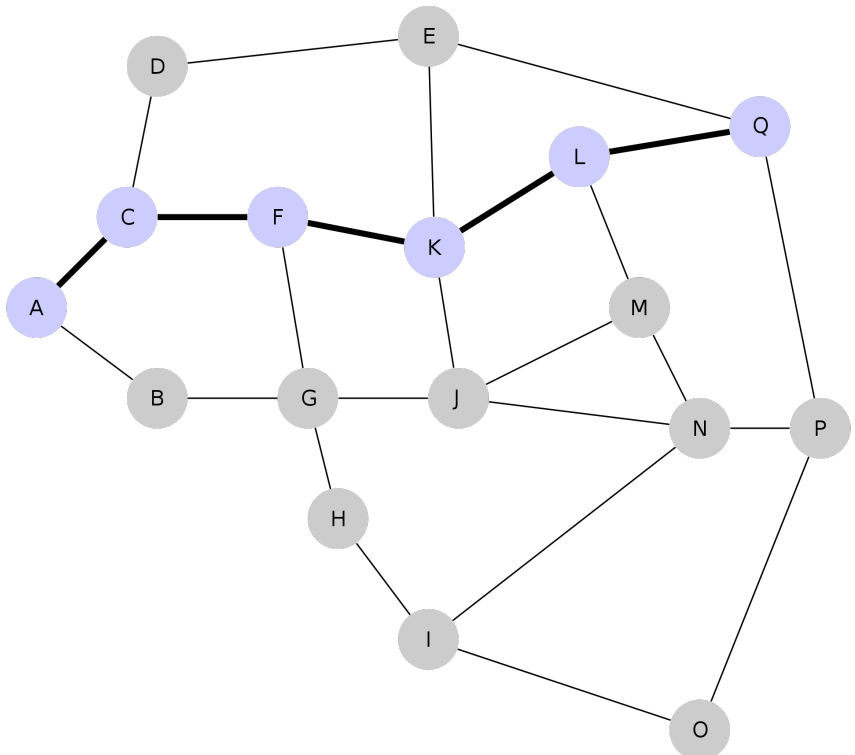
- **Payment channel** is a construction that allows **two** parties to transact Bitcoin without submitting any transactions to the Bitcoin network.
- **Bidirectional payment channel** is somewhat similar to a payment check that splits a joint bank account between two parties.
  - joint bank account with ballance N;
  - both parties A and B "own" $N/2$ portions of the ballance;
  - both parties sign a check that pays $N/2$ money to A and B;
  - when party A wants to pay M money to party B, they **sign a new check** that pays $N/2 - M$ to party A and $N/2 + M$ to party B and **destroy the old checks**.

Payment Channels 2/2

- Several proposals: Spillman, CLTV, Poon-Dryja, Decker-Wattenhofer duplex payment channels, Decker-Russell-Osuntokun eltoo Channels.
- Poon-Dryja payment channels were presented in the Lightning Network paper.
- Channel backing funds are locked into a 2-of-2 multisig.
- Before the funding transaction is even signed, commitment transactions for each party are first written and signed.
- As it requires referring to transactions that have not been signed yet, it requires using a transaction format that separates signatures from the part of the transaction that is hashed to generate the txid, such as Segregated Witness.

Lightning Network 1/2

- A network of bidirectional payment channels that allows to execute multi-hop payments, propagating funds through a series of payment channels.
- Proposed in 2015, mainnet network started operation in early 2018.



Lightning Network 2/2

- Entity A wants to pay entity B and there is a path within the network between them $A, C_1, C_2, ..., C_n, B$:
  - B generates a random value R and computes a hash $H = hash(R)$ and provides H to A;
  - A and creates an HTLC (Hashed Timelock Contract) and updates it's channel with C_1:

```
HASH160 <H> EQUAL
IF
  <B public key> CHECKSIG
ELSE
  <locktime> CHECKLOCKTIMEVERIFY
  <A public key> CHECKSIG
ENDIF
```

  - C_1 updates its payment channels with C_2 and so on, until C_n updates channel with B.
  - B provides R to C_n and pulls funds, C_n provides R to C_{n-1} and so on until C_1 pulls funds from A.

Lightning Network Usage

- 20,478 nodes,
- 45,774 channels,
- 1,332.25 BTC ($52,290,595),
- Ongoing research, improvements and new feature development,
- Games, online shops and other businesses.

The End

Thank you!