CS915/435 Advanced Computer Security - Emerging topics

BitCoin and Blockchain

Overview

- Bitcoin address
- Bitcoin transactions
- Locking and unlocking script
- Blocks and Bitcoin mining
- Blockchain

History

- Research on digital currency dates back to early 80’s.
- 1983, David Chaum proposed e-cash using blind signatures. He set up a company called DigitCash (later went bankrupt)
- 1997, Adam Back proposed proof-of-work called Hashcash to limit email spam
- 1998, Wei Dai proposed b-money: proof-of-work, broadcasting, signing, decentralized ledger, incentivisation of mining
- 1998, Nick Szabo proposed Bit Gold, commonly seen as precursor to BitCoin. But Bit Gold has a **double-spending** problem.
- 31 October 2008, Satoshi Nakamoto proposed BitCoin.
- 3 Jan, 2009, Bitcoin network came to existence. BitCoin was born!

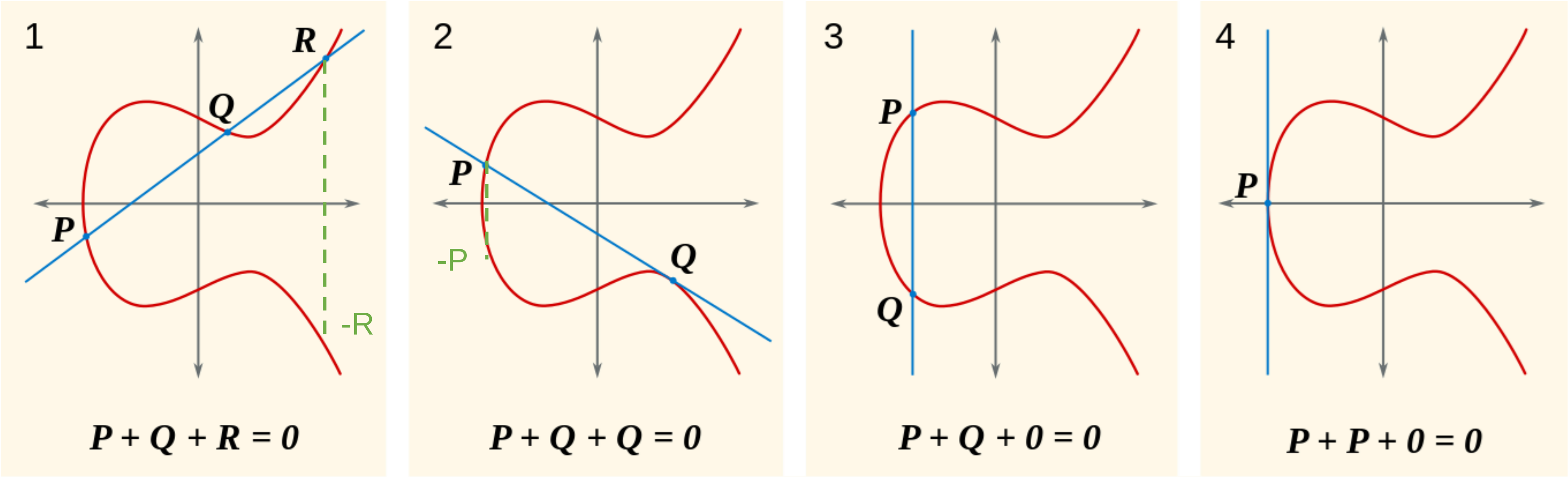
For latest price, see https://coinmarketcap.com/currencies/bitcoin/

Background on Elliptic Curve Cryptography

$$y^2 = x^3 + ax + b$$

FF: $X = g^x \bmod p$

ECC: $X = X.G$



Bitcoin uses Elliptic Curve Cryptography

BitCoin uses a NIST curve secp256k1

$$y^2 = x^3 + 7$$

Generating Public and Private Keys

```
Private-Key: (256 bit)
priv:
    01:68:d2:65:bf:f8:66:88:e0:b0:64:d5:76:cc:7d:
    51:ae:1d:5b:62:64:fd:2e:1e:24:ec:53:eb:5d:9d:
    0c:20
pub:
    04:73:e3:c6:ce:48:da:81:fd:c1:04:86:74:83:4f:
    06:27:85:88:c4:af:59:7b:bf:bc:a6:ef:5a:57:52:
    07:16:bc:b7:15:f8:a4:f5:16:f0:a7:20:2a:1a:59:
    e4:8b:0d:41:f7:ab:ae:ba:86:3c:37:4a:79:7c:02:
    75:3b:34:27:d7
ASN1 OID: secp256k1
```

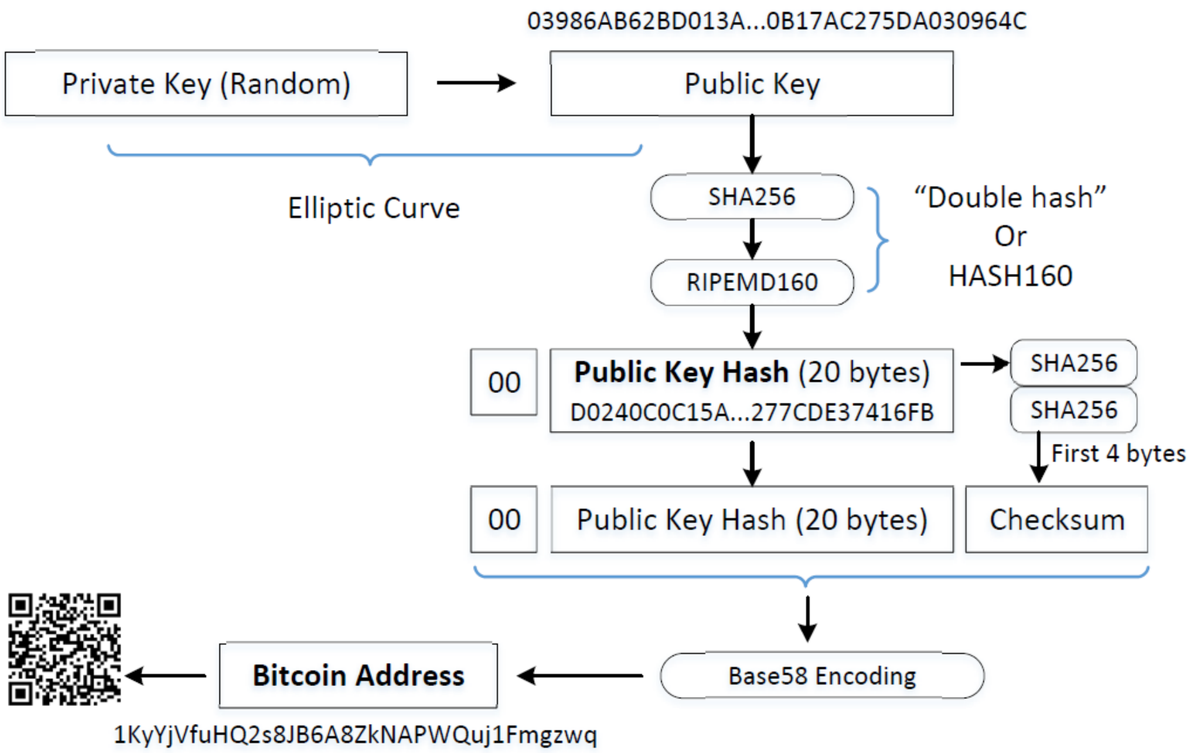
Compressing Public Key

```
pub:
    03:73:e3:c6:ce:48:da:81:fd:c1:04:86:74:83:4f:
    06:27:85:88:c4:af:59:7b:bf:bc:a6:ef:5a:57:52:
    07:16:bc
ASN1 OID: secp256k1
```

Generating Public-Key Hash

```
$ echo 0373e3c6c...5a57520716bc | xxd -r -p \
        | openssl dgst -sha256 -binary      \
        | openssl dgst -ripemd160
(stdin)= 9390b28a0280cde7eac94e410a74f652aed6e937
```

Turning Public-Key Hash into Bitcoin Address



Base58 Encoding

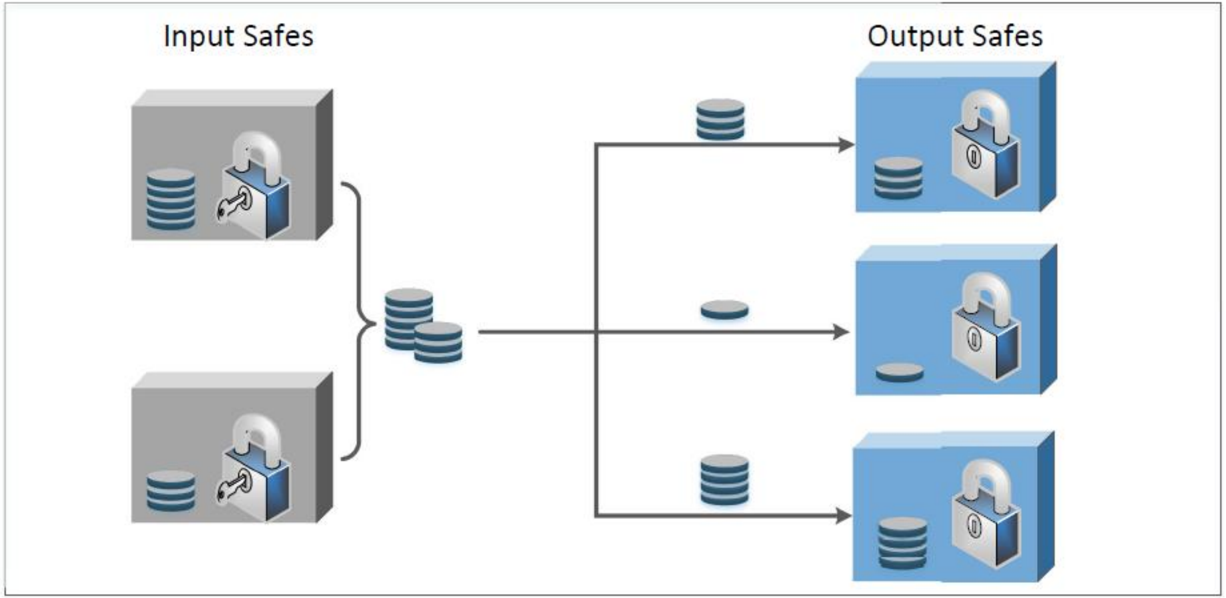
Integer	Division	Quotient	Remainder	Base58 Symbol
2,864,386,338	div 58	49,385,971	20	M
49,385,971	div 58	851,482	15	G
851,482	div 58	14,680	42	j
14,680	div 58	253	6	7
253	div 58	4	21	N
4	div 58	0	4	5

Final Base58 encoding: **5N7jGM**

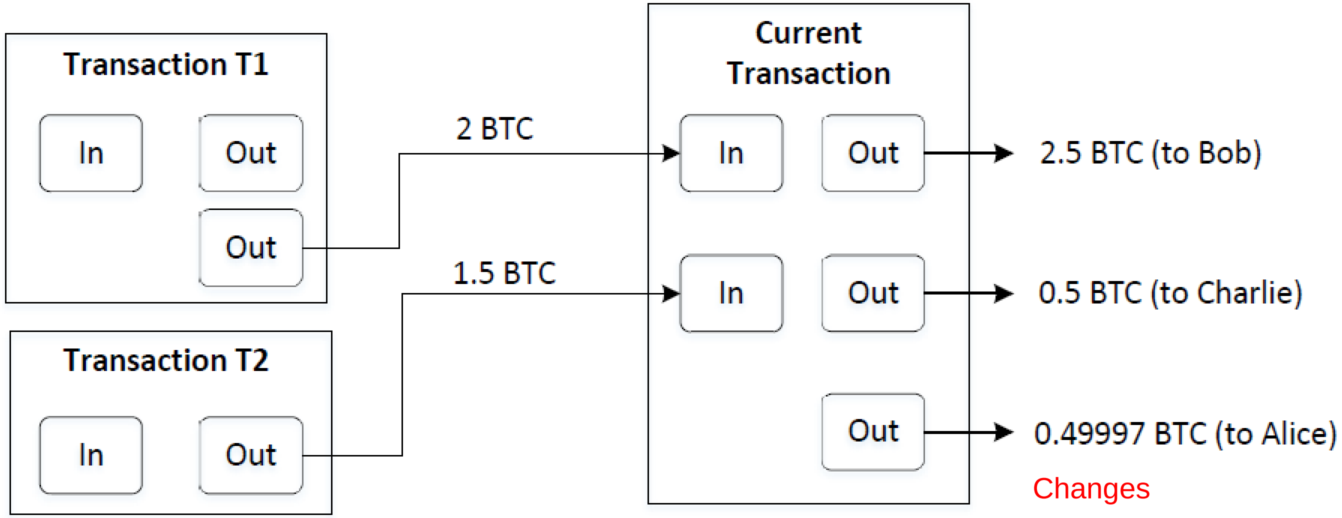
```
$ echo aabb1122 | xxd -r -p | python base58.py && echo
5N7jGM
$ echo 009390b28a0280cde7eac94e410a74f652aed6e937b2b07b3f \
    | xxd -r -p | python base58.py && echo
1ETFfxDNaF8rWsuorMKhdHruxSuT9BDUGE
```

Transactions: Intuition

Components of Transactions & Examples



An Example



Transaction fee = (2 + 1.5) – (2.5 + 0.5 + 0.49997) = 0.00003 BTC

Input

The input to the current transaction specifies the source of the money

Input 0:

Transaction ID: T1

Output Index: 1

ScriptSig (Unlocking Script): … omitted …

Input 1:

Transaction ID: T2

Output Index: 0

ScriptSig (Unlocking Script): … omitted …

Output

The output of a transaction specifies where the money goes.

Output 0:

Value: 2.5 BTC

ScriptPubKey: (a lock that can only be unlocked by Bob)

Output 1:

Value: 0.5 BTC

ScriptPubKey: (a lock that can only be unlocked by Charlie)]

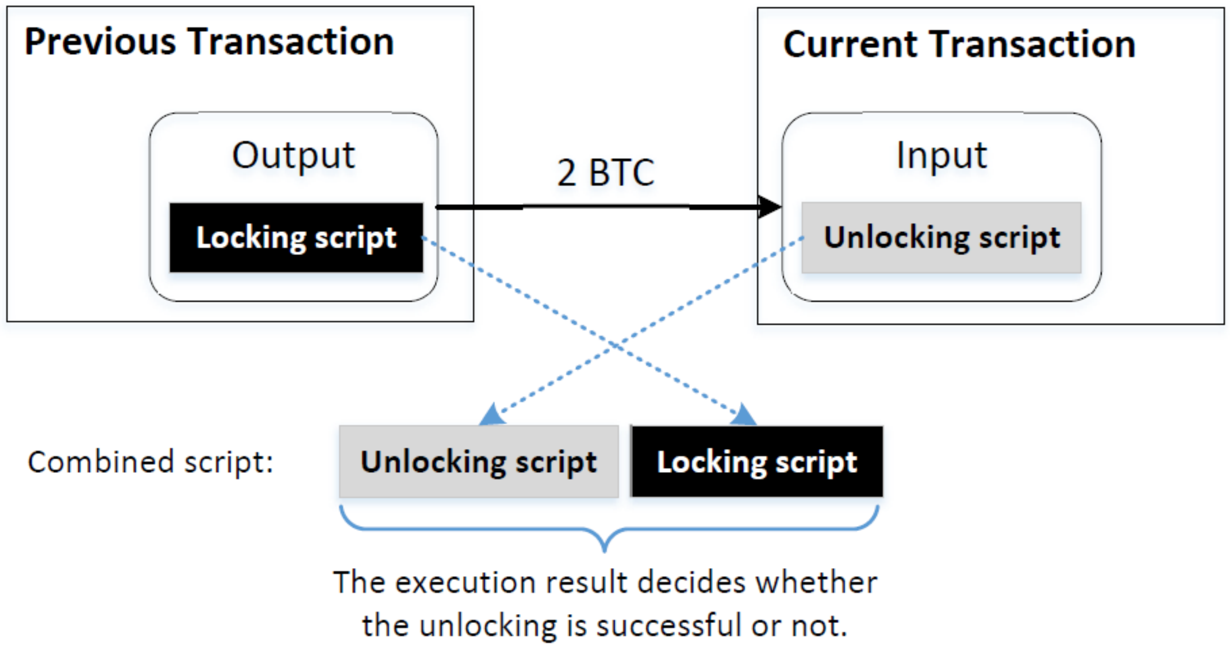
Output 2:

Value: 0.49997 BTC

ScriptPubKey: (a lock that can only be unlocked by Alice)

Alice pays herself (change)

Locking and Unlocking Script



Unlocking the output of a transaction

- Two types of locking scripts
  - Pay-to-Pubkey-Hash
  - Pay-to-Script-Hash
- Locking/unlocking is done by script
- In BitCoin, script is a basic programming language: no loops (not Turing-complete).

Script Examples

No one can spend

```
scriptPubKey: OP_RETURN
scriptSig: ... (does not matter)

Combined script: ... (does not matter) ... OP_RETURN
```

Any one who can find two numbers that add to 100 can unlock

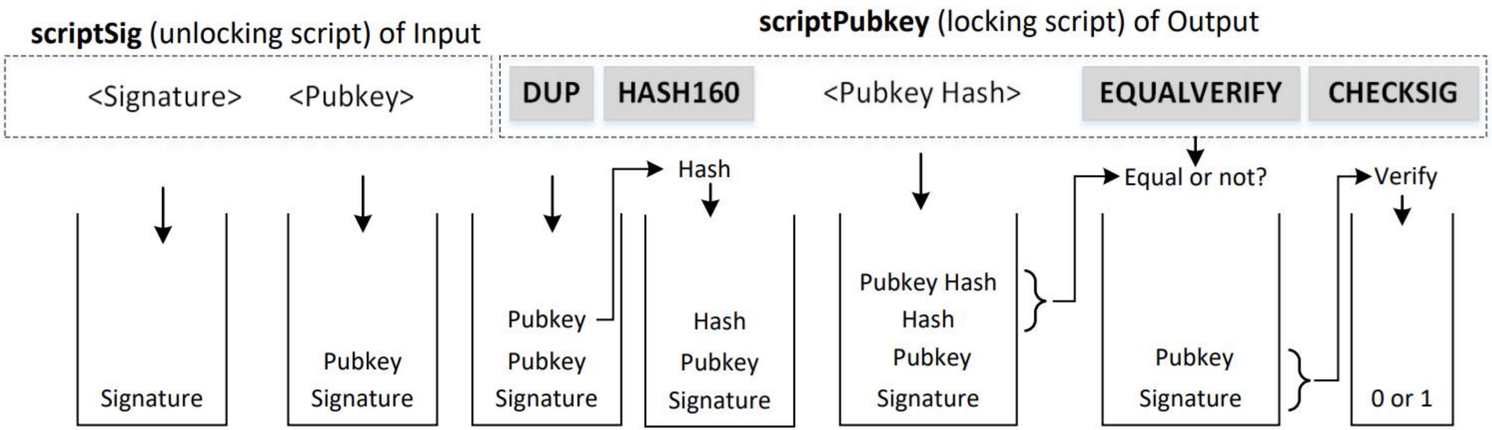
```
scriptPubKey: OP_ADD <100> OP_EQUAL
scriptSig:    <5> <95>

Combined script: <5> <95> OP_ADD <100> OP_EQUAL
```

Anyone who can find preimage of a hash can unlock

```
scriptPubKey: OP_SHA256 <6fe2...3ffe> OP_EQUAL
scriptSig:    <f343...f0f5>

Combined script:
    <f343...f0f5> OP_SHA256 <6fe2...3ffe> OP_EQUAL
```

Pay-to-PubKey-Hash (P2PH)

```
scriptPubKey: OP_DUP OP_HASH160 <Public KeyHash> OP_EQUAL OP_CHECKSIG
scriptSig:    <Signature> <Public Key>

Combined script: <Signature> <Public Key> OP_DUP OP_HASH160
                 <Public KeyHash> OP_EQUAL OP_CHECKSIG
```



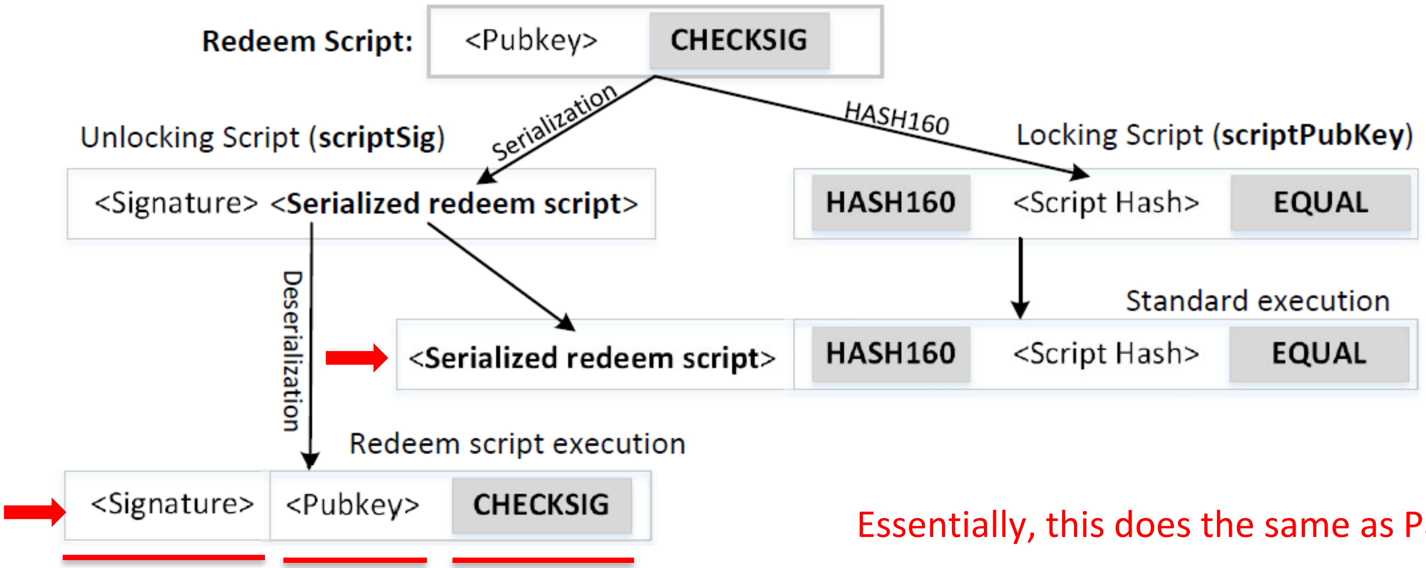
Pay-to-MultiSig (P2MS)

```
scriptPubKey: <2> <PubKey 1> <PubKey 2> <PubKey 3> <3>
              OP_CHECKMULTISIG
scriptSig: <Signature 1> <Signature 2>

Combined script:
    <Signature 1> <Signature 2>
    <2> <PubKey 1> <PubKey 2> <PubKey 3> <3> OP_CHECKMULTISIG
```

- As long as two out of three approve (with a digital signature), they can spend the money.
- However, it’s being replaced by P2SH

Pay-to-Script-Hash (P2SH)

```
scriptPubKey: OP_HASH160 <Script Hash> OP_EQUAL

scriptSig: <Unlocking Script> <Serialized Redeem Script>
```



Essentially, this does the same as Pay-to-PubKey-Hash

Use P2SH for MultiSig

Pay-to-MultiSig (P2MS)

```
scriptPubKey: <2> <PubKey 1> <PubKey 2> <PubKey 3> <3>
              OP_CHECKMULTISIG
scriptSig: <Signature 1> <Signature 2>

Combined script:
    <Signature 1> <Signature 2>
    <2> <PubKey 1> <PubKey 2> <PubKey 3> <3> OP_CHECKMULTISIG
```

Pay-to-Script-Hash (P2SH)

```
Redeem Script:
   <2> <PubKey 1> <PubKey 2> <PubKey 3> <3> OP_CHECKMULTISIG

scriptPubKey: OP_HASH160 <Hash of Redeem Script> OP_EQUAL
scriptSig:    <Sig 1> <Sig 2> <Serialized Redeem Script>
```

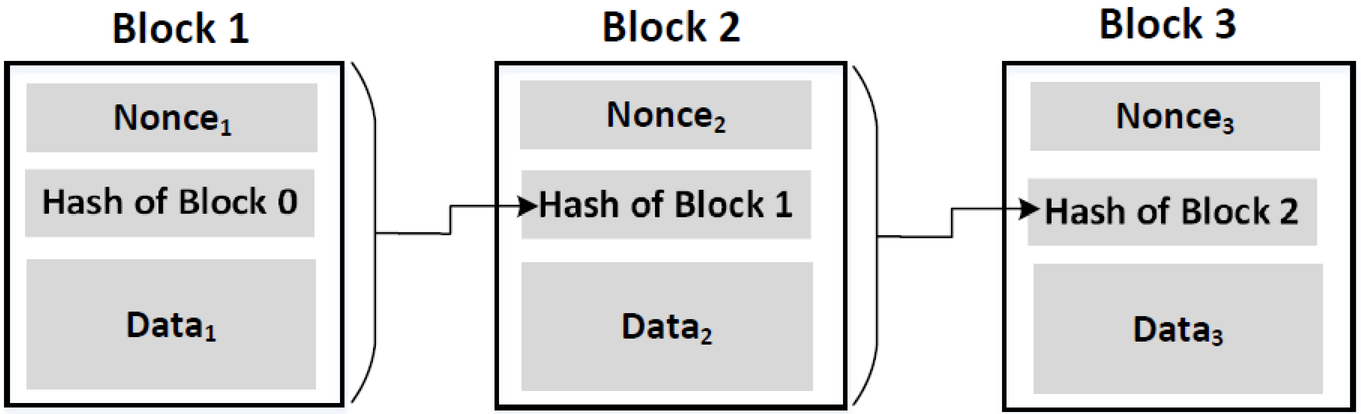
In P2SH, the receiver needs to include the original redeem script. Bigger size -> higher transaction fee. Hence, P2SH moves the cost to the receiver of the payment.

Sending Transaction

- After a node has generated a transaction, it sends the transaction to its peers
- Each peer will verify the transaction, and then forward it to their peers
- Eventually, every node on the network will receive the transaction
- Some special node called miner will be responsible for adding the transaction to the public ledger (i.e., blockchain).

Generating Blocks

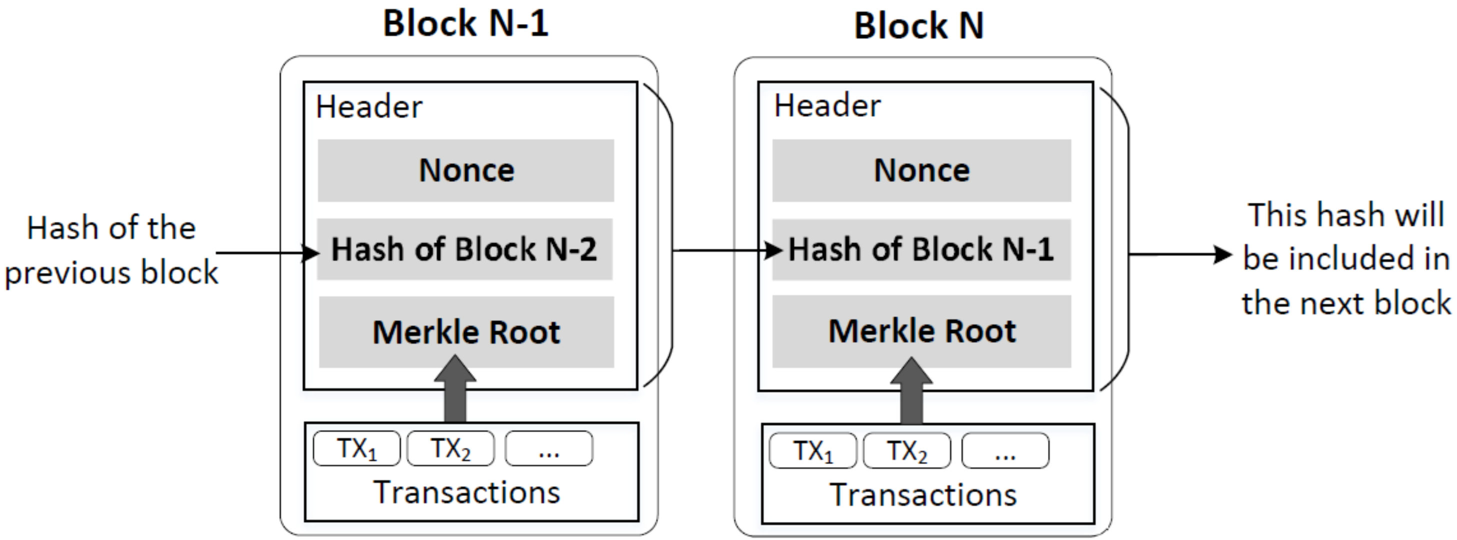
- Miners group transactions into a new block
- The new block is appended to the existing blockchain
- Check [cryptocurrency transactions]() (mining, validation).



Mining

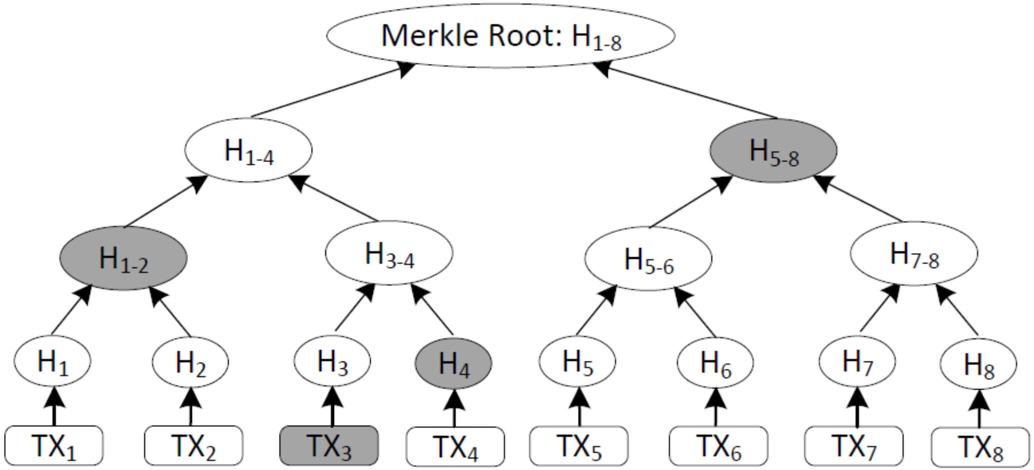
- Proof-of-Work: find a nonce, s.t. when the hash of the block satisfies a special requirement, such as having 20 leading zeros
- Rewarding:
  - Coinbase transaction: new bitcoins are minted and given to the miner (50 BTC in 2008; halved every 210K blocks. Now 6.25 BTC)
  - Transaction fees
- Once a miner has found a block, it immediately sends the block to its peers, who will verify the block and then forward the block to their peers.
- Eventually, all the nodes will see this new block, and add it to their ledgers.

Include Merkle Root in Block



Transactions are organised in a tree-like structure, and only the root of the tree is included into the calculation of the block hash.

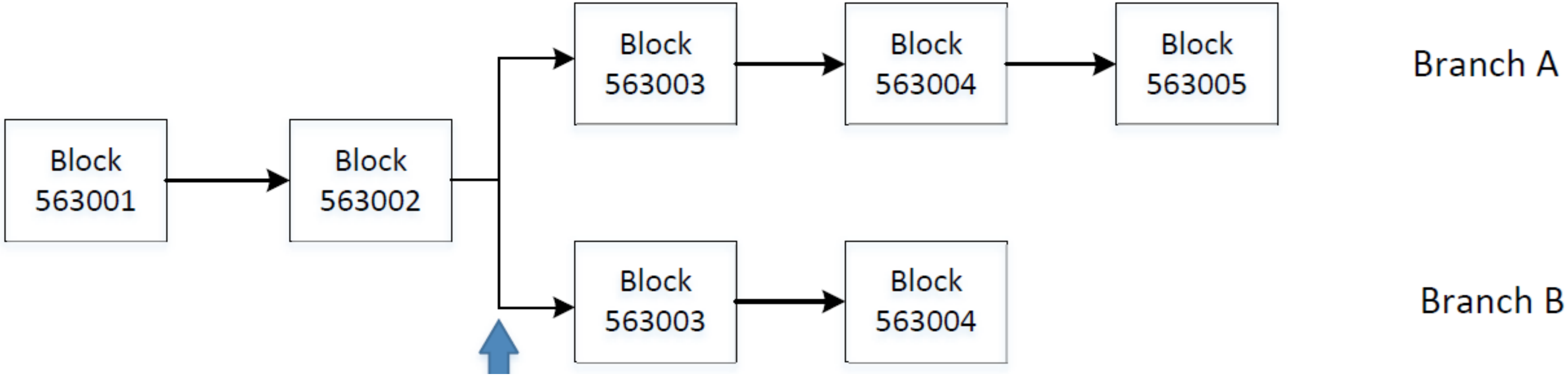
Merkle Tree



Benefit:

- To find whether a transaction is included in a block, you don't need all the transactions.
- The cost of checking inclusion is O(log n) rather than O(n)

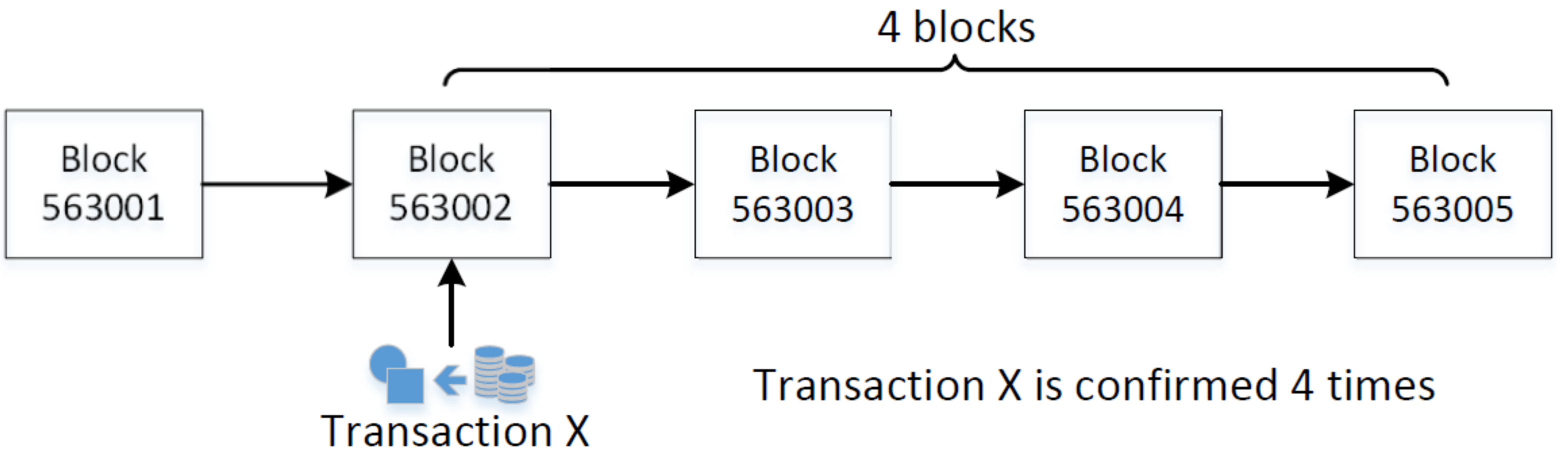
Branching



Branching occurs when two valid blocks are found at about the same time

The longest chain wins

Confirmation Number

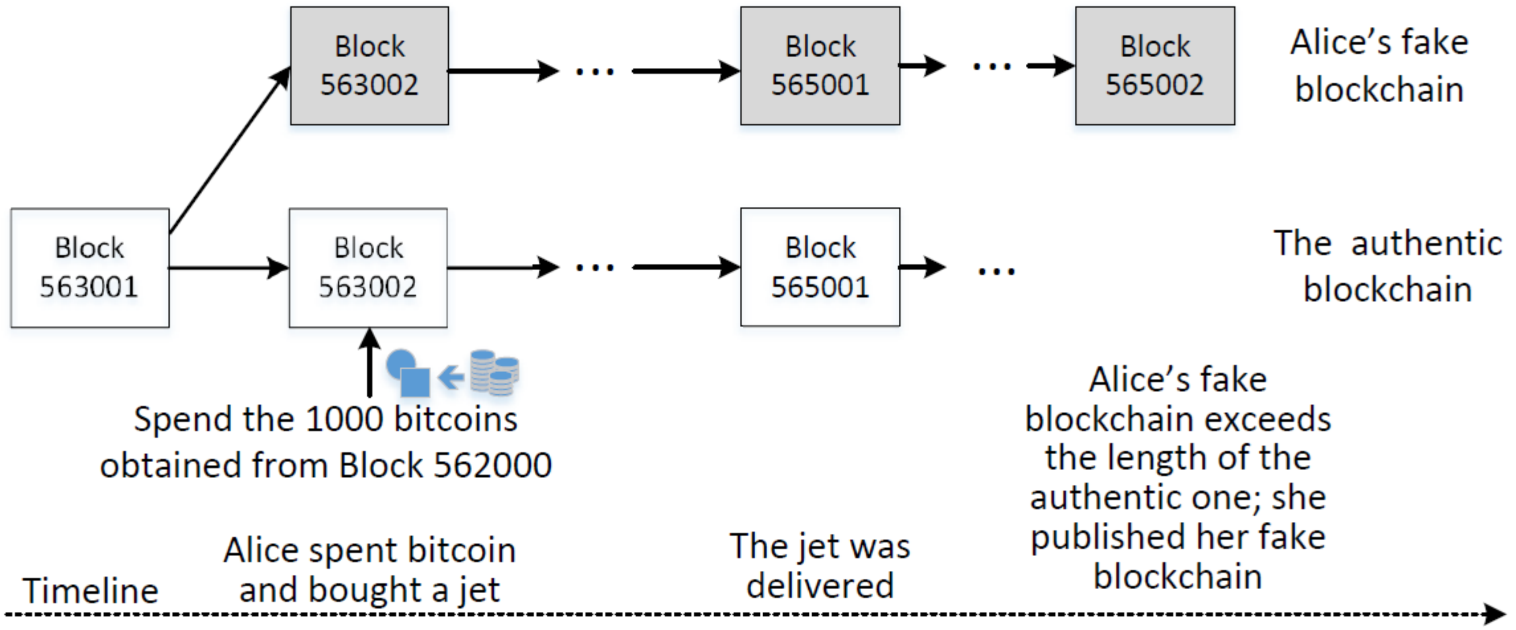


The larger a block's confirmation number is, the less likely it will be removed from the blockchain

Probability of Double Spending

Confirmation	2%	8%	10%	20%	30%	40%	50%
1	4%	16%	20%	40%	60%	80%	100%
2	0.237%	3.635%	5.600%	20.800%	43.200%	70.400%	100%
3	0.016%	0.905%	1.712%	11.584%	32.616%	63.488%	100%
4	0.001%	0.235%	0.546%	6.669%	25.207%	57.958%	100%
5	≈ 0	0.063%	0.178%	3.916%	19.762%	53.314%	100%
6	≈ 0	0.017%	0.059%	2.331%	15.645%	49.300%	100%
7	≈ 0	0.005%	0.020%	1.401%	12.475%	45.769%	100%
8	≈ 0	0.001%	0.007%	0.848%	10.003%	42.621%	100%

Double Spending with Majority Hash Power



Actual incidents

- July 2014, the mining pool ghash.io briefly exceeded 50% of the bitcoin network hash power.
- It voluntarily reduced the mining power to 40%
- 2018, Bitmain mined 42% of the Bitcoin blocks during a week in June.
- With 42% computing, the success rate to do double spending is high: 58% with 5 confirmation.
- Hence, double spending is feasible for major mining pools, but this doesn’t mean the mining pools have the incentive do the attack.

Hardware security – Crypto wallet

- Currently, an HSM can cost between £20.000 to £40.000
- How secure are crypto wallets (e.g., Ledger, Trezor)?
- It turns out that there are several attacks against these wallets using side-channel attacks and voltage glitching.
- You may check this YouTube video where Joe Grand was able to retrieve the PIN from a Trezor wallet

Summary

- Bitcoin address
- Transactions, locking and unlocking script
- Bitcoin mining
- Blockchain, branching, confirmation number, and double spending