Ecclesiastes (Eccl) 12:13

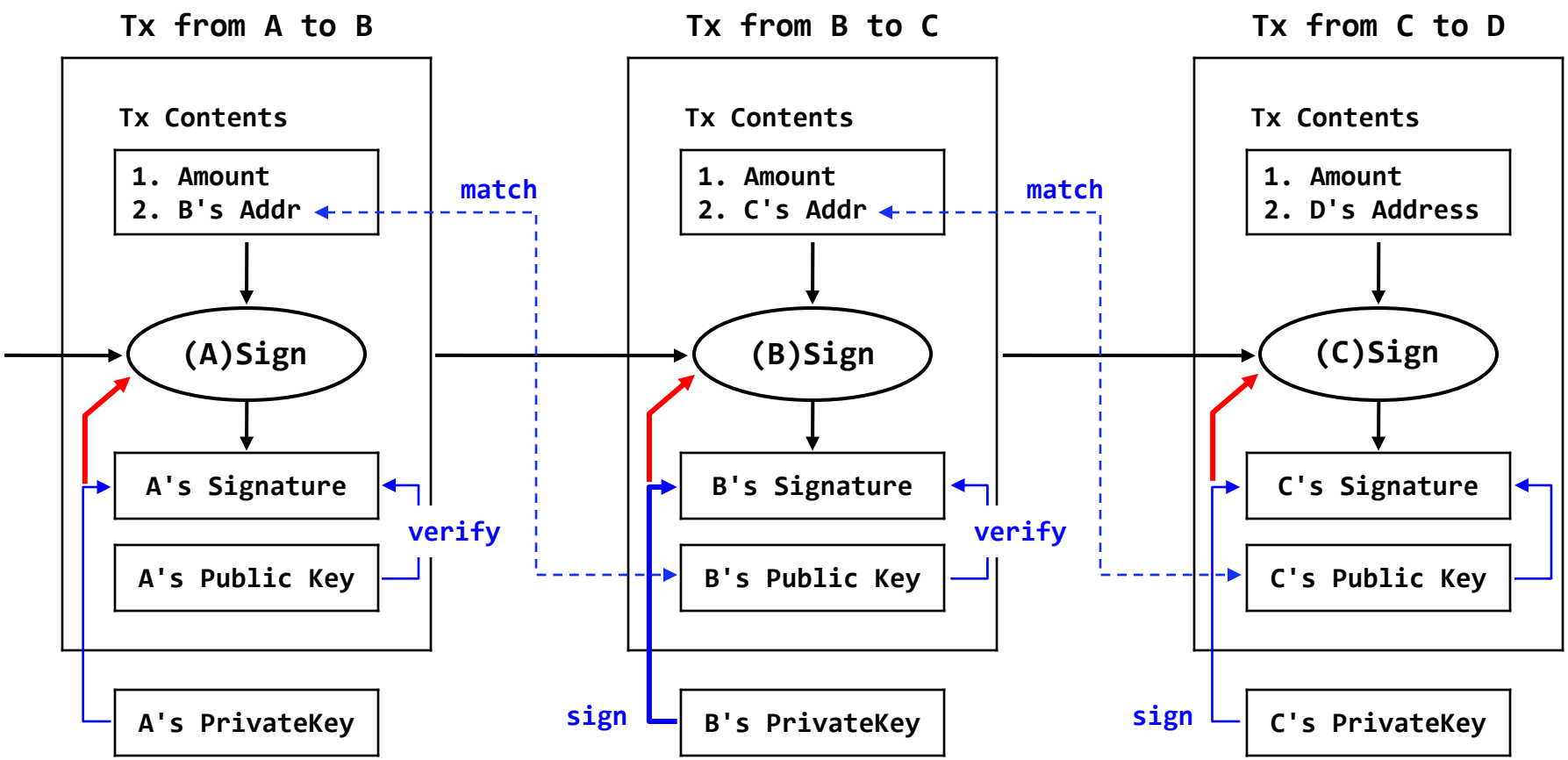
Now all has been heard;
here is the conclusion of the matter :

Have reverence for God, and obey his commands,
because this is all that man was created for.

Fear God and keep his commandments,
for this is the whole duty of man.

UTXO

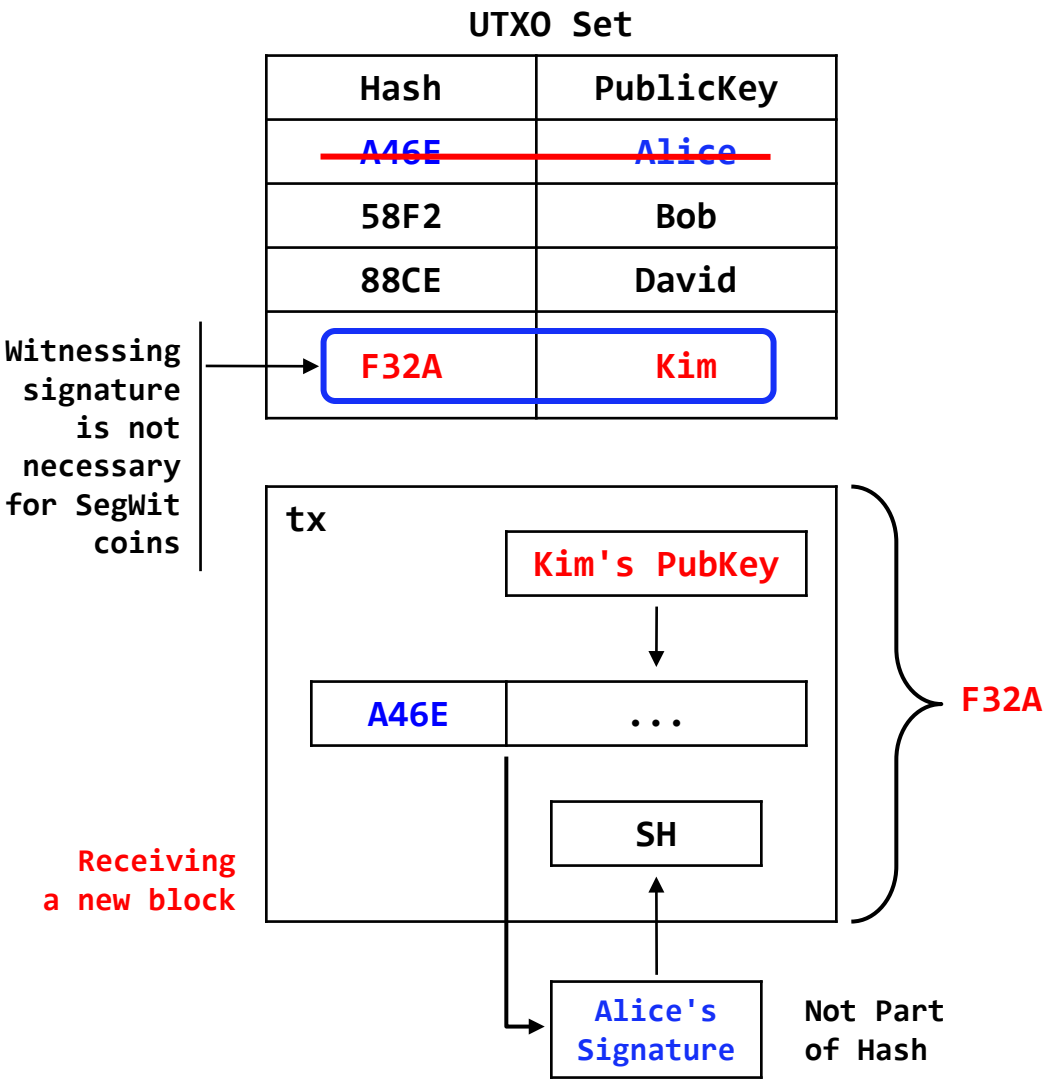
Transaction Flow



✝ Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)

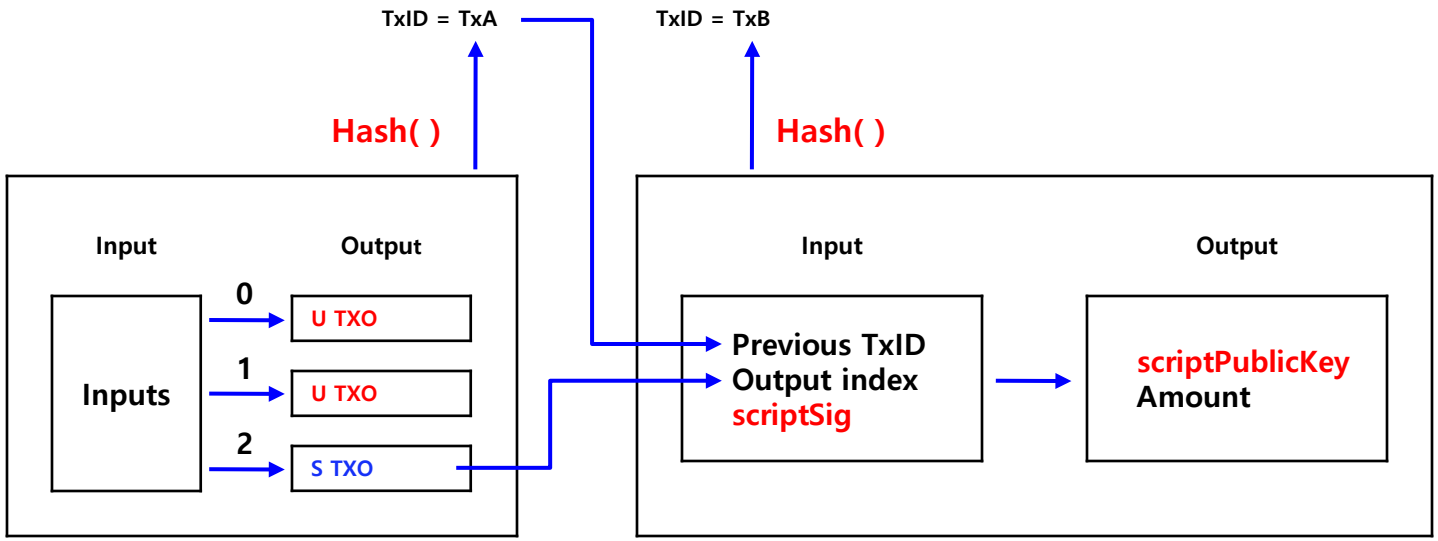
Transferring Ownership without Witnessing Signature

- Each node maintains a ledger of which coins belong to which entities (UTXO set)
- Upon receiving a new block, miner parses transactions, removing spent outputs from his UTXO set and adding a newly-created outputs
- For Bitcoins, since outputs are identified by hash, miner cannot update his UTXO set without witnessing the signatures that authorize the transfer
  - For SegWit coins, this does not hold



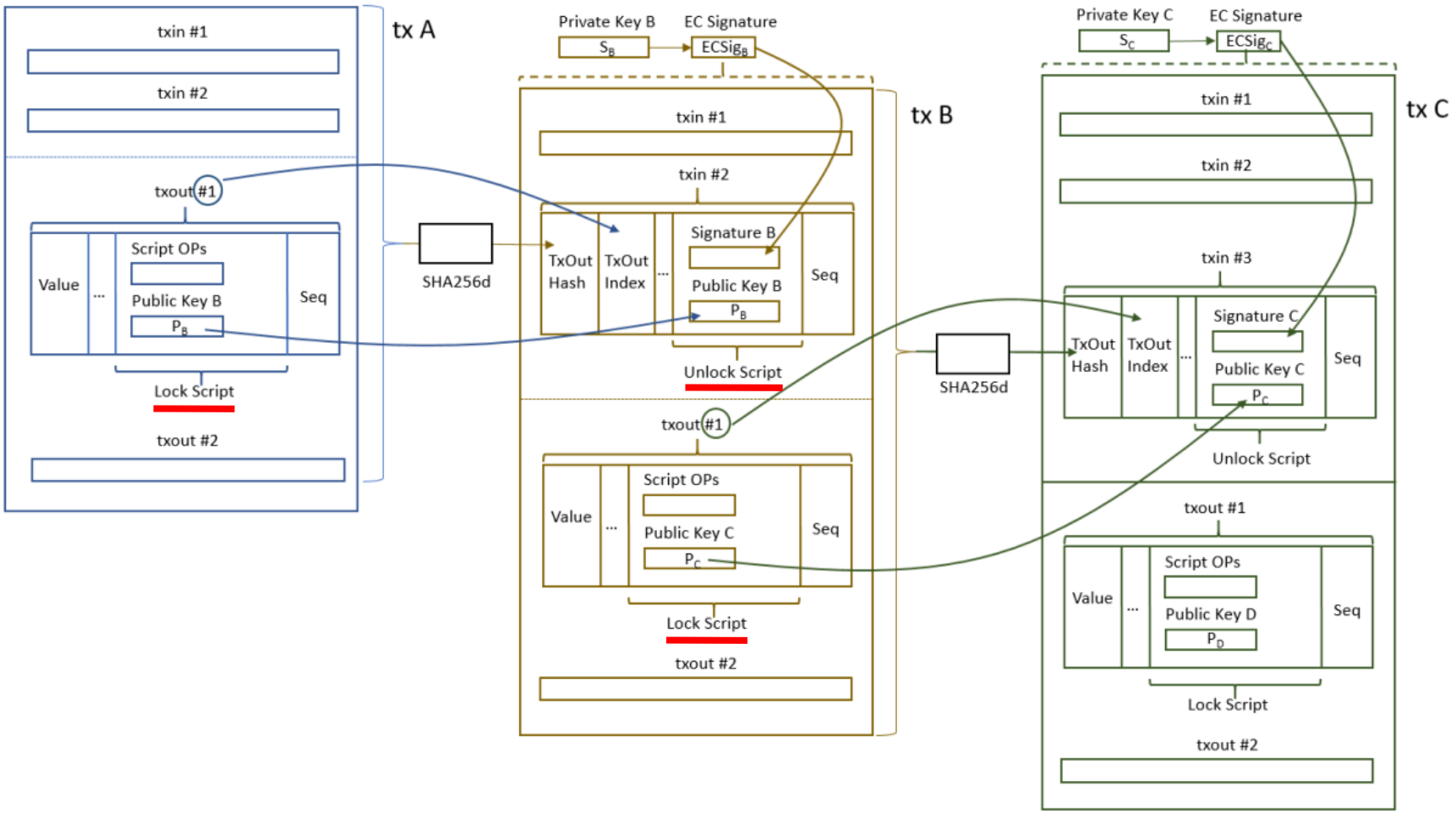
✝ Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)

U (UnSpent) vs S (Spent) TXO



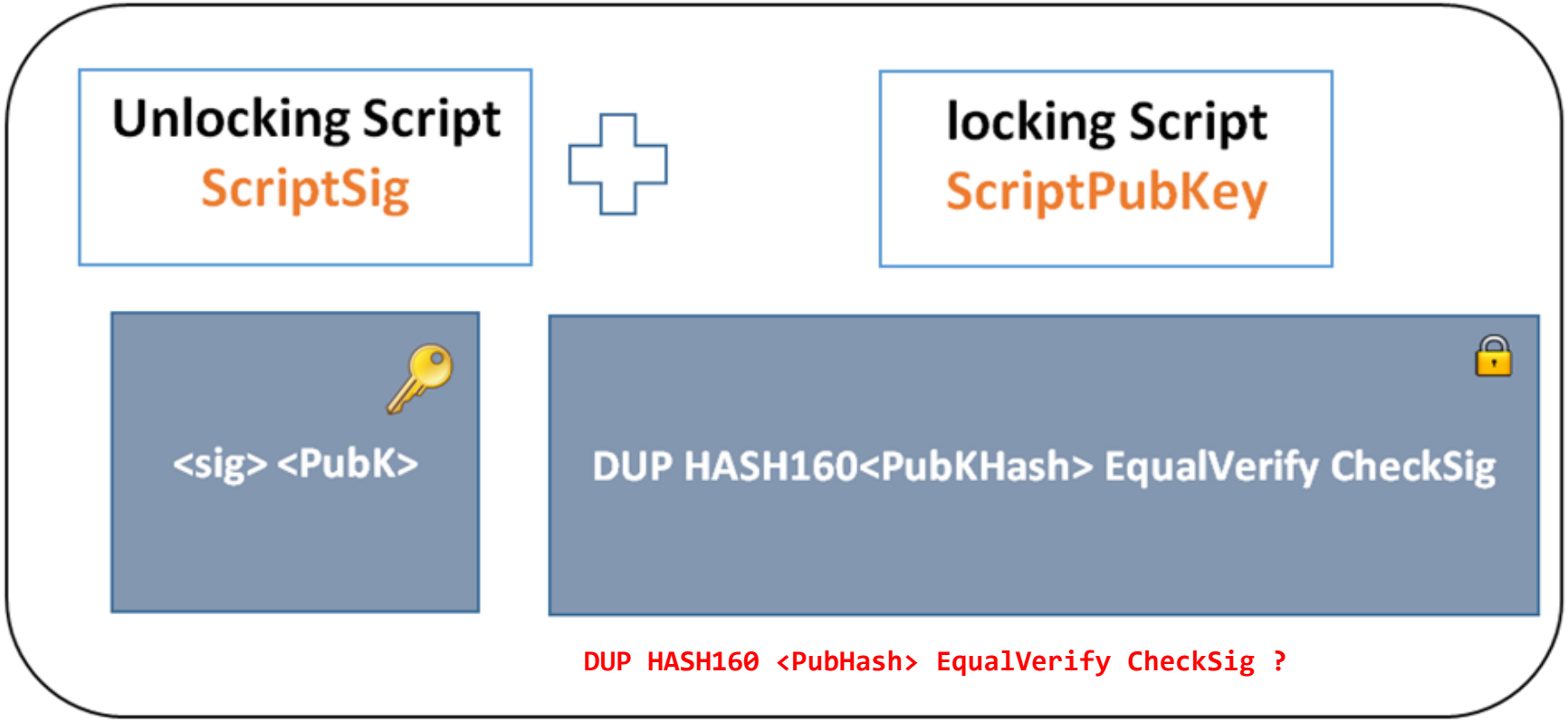
✝ Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)

https://privatekeys.org/2018/04/17/anatomy-of-a-bitcoin-transaction/



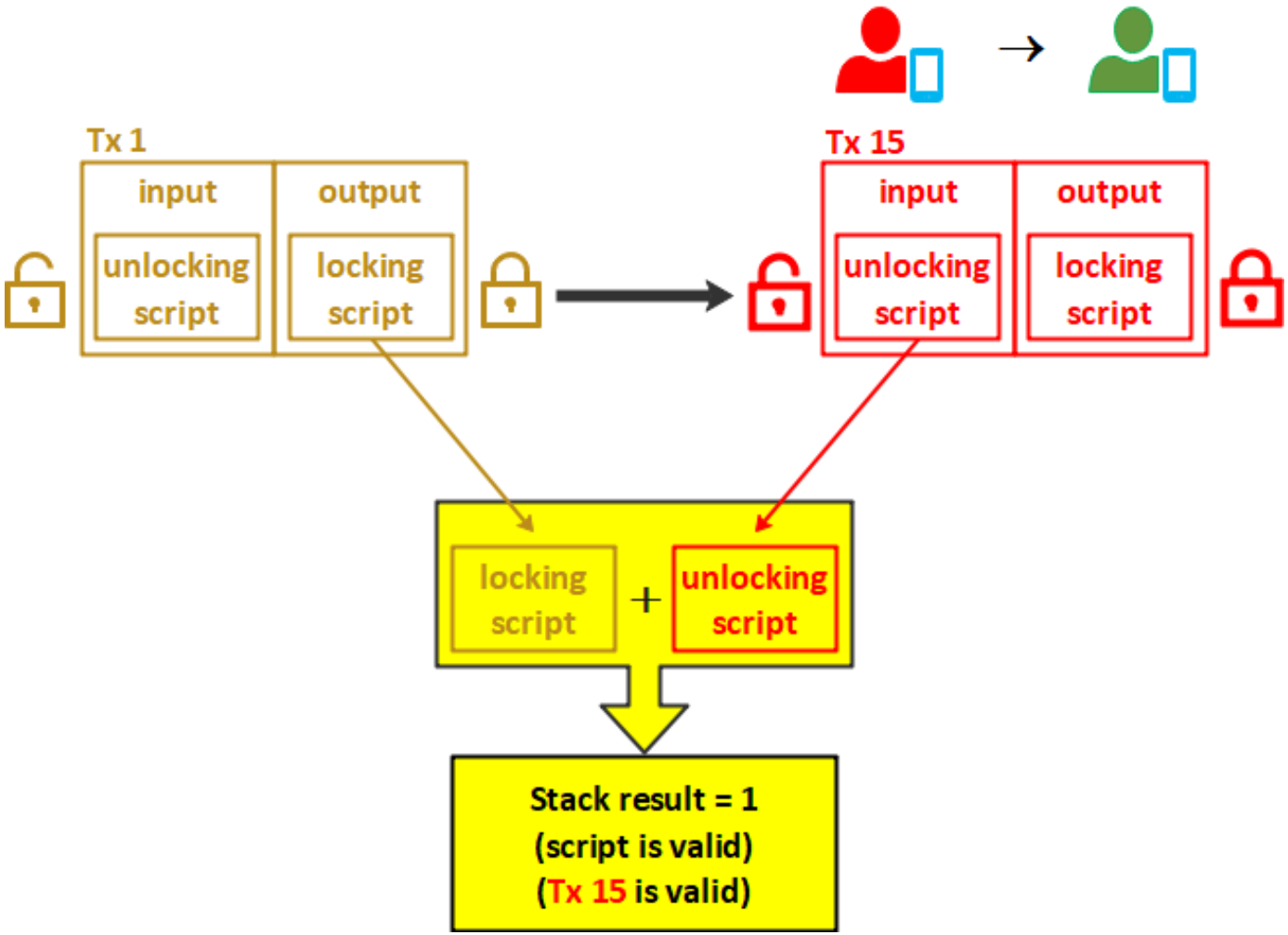
✝ **Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)**

Unlocking and Locking Scripts



✝ Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)

Scripts in Tx



✝ **Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)**

Bitcoin Address

Bitcoin Address

- Legacy address : 1xx
- Nested Segwit address : 3xx
  - Use Compressed Pub Key
- Native SegWit address (Bech32) : bc1xx, tb1xx
  - github.com/bitcoin/bips/blob/master/bip-0173.mediawiki

- BIP (Bitcoin Improvement Proposal)
  - https://github.com/bitcoin/bips/

Bitcoin Address

- Pay to Public Key Hash (P2PKH) : 1xx
  - Legacy addresses
  - Example: 15e15hWo6CShMgbAfo8c2Ykj4C6BLq6Not

- Pay to Script Hash (P2SH) : 3xx
  - Nested SegWit address
  - Example: 35PBEaofpUeH8VnnNSorM1QZsadrZoQp4N

- Pay to Witness Public Key Hash (P2WPKH) : 'bc1xx' or 'tb1xx'
  - Native SegWit address
  - Example: bc1q42lja79elem0anu8q8s3h2n687re9jax556pcc
  - tb1xxx : testnet SegWit address

Bitcoin Address

Seed Phrase

BIP-39 (Mnemonic Phrase)
24 Words
github.com/bitcoin/bips/blob/master/bip-0039/english.txt

↓

Private Key

WIF Format = Version(1) + PrivateKey(32) + Checksum(4) = 37-Byte

↓ ECDSA (Elliptic Curve Digital Signature Algorithm)

Public Key

Uncompressed form : 1 + 32+32 = 65-byte (0x4)
Compressed form : 1 + 32 = 33-byte (0x2, ox3)

↓ Hash160 : SHA256 -> RIPEMD160

Pub Key Hash

Version + 20 bytes Hash + 4-byte Checksum

Version Byte
1. P2PKH (Pay-to-Public-Key-Hash) : 0x0 mainnet, 0x6F testnet
2. P2SH (Pay-to-Script-Hash) : 0x05 mainnet, 0xC4 testnet
3. P2WPKH (Pay-to-Witness-Public-Key-Hash) : 0x0 mainnet, 0x06 testnet
4. P2WSH (Pay-to-Witness-Script-Hash) : 0x0 mainnet, 0x0A testnet

↓ Base58 Encoding

Bitcoin Address

✝ Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)

Mnemonic Code

- Bitcoin Seed PassPhrase
  - BIP39 (Mnemonic Phrase)
  - 24 Words
  - https://github.com/bitcoin/bips/blob/master/bip-0039/bip-0039-wordlists.md

- https://bitaps.com/mnemonic

Mnemonic Code

- Bitcoin Seed PassPhrase
  - BIP39 (Mnemonic Phrase)
  - 24 Words
  - https://github.com/bitcoin/bips/blob/master/bip-0039/bip-0039-wordlists.md

- https://bitaps.com/mnemonic

Mnemonic Code

- Mnemonic 2048 words
  - https://github.com/bitcoin/bips/blob/master/bip-0039/english.txt
- Mnemonic word
  - 11-bit -> 2^11 = 1024 * 2 = 2048 words
- Mnemonic 24 words
  - 11 * 24 = 240 + 24 = 264 = 256 + 8
    - Entropy 256-bit + First 8-bit of Checksum Hash
  - 11 * 23 (My Selected Words) = 230 + 23 = 253
    - Need 3-bit + Checksum 8-bit => Last Word
    - My 23 Words + Last Word = mnemonic 24 Words

Bitcoin Address Generation Process

- http://gobittest.appspot.com/Address

```
0 - Private ECDSA Key (32 bytes * 8 = 256 bits)
1 – Prefix + Public ECDSA Key
        04 : Uncompressed public key (1 + 64 = 65 bytes)
        02 or 03 : Compressed public key (1 + 32 = 33 bytes)
  2 - SHA-256 hash
  3 - RIPEMD-160 Hash (20 bytes)
4 - Adding Version (Network ID) byte to 3 (21 bytes = network ID + 20 bytes)
  5 - SHA-256 hash (32 bytes)
  6 - SHA-256 hash (32 bytes = 4 bytes + 28 bytes)
  7 – Get First Four bytes of 6
8 – Adding 7 at the end of 4 (25 bytes = 21 bytes + 4 bytes)
9 - Base58 encoding of 8 (17 bytes) => 17 bytes bitcoin address
```

- Digital signature : ECDSA (Elliptic Curve Digital Signature Algorithm)
- Hash160 : sha256, RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest)
- Compress string : BASE58, BASE58 Encoder/Decoder
- WIF (Wallet Import Format) format

Base58 Encoding

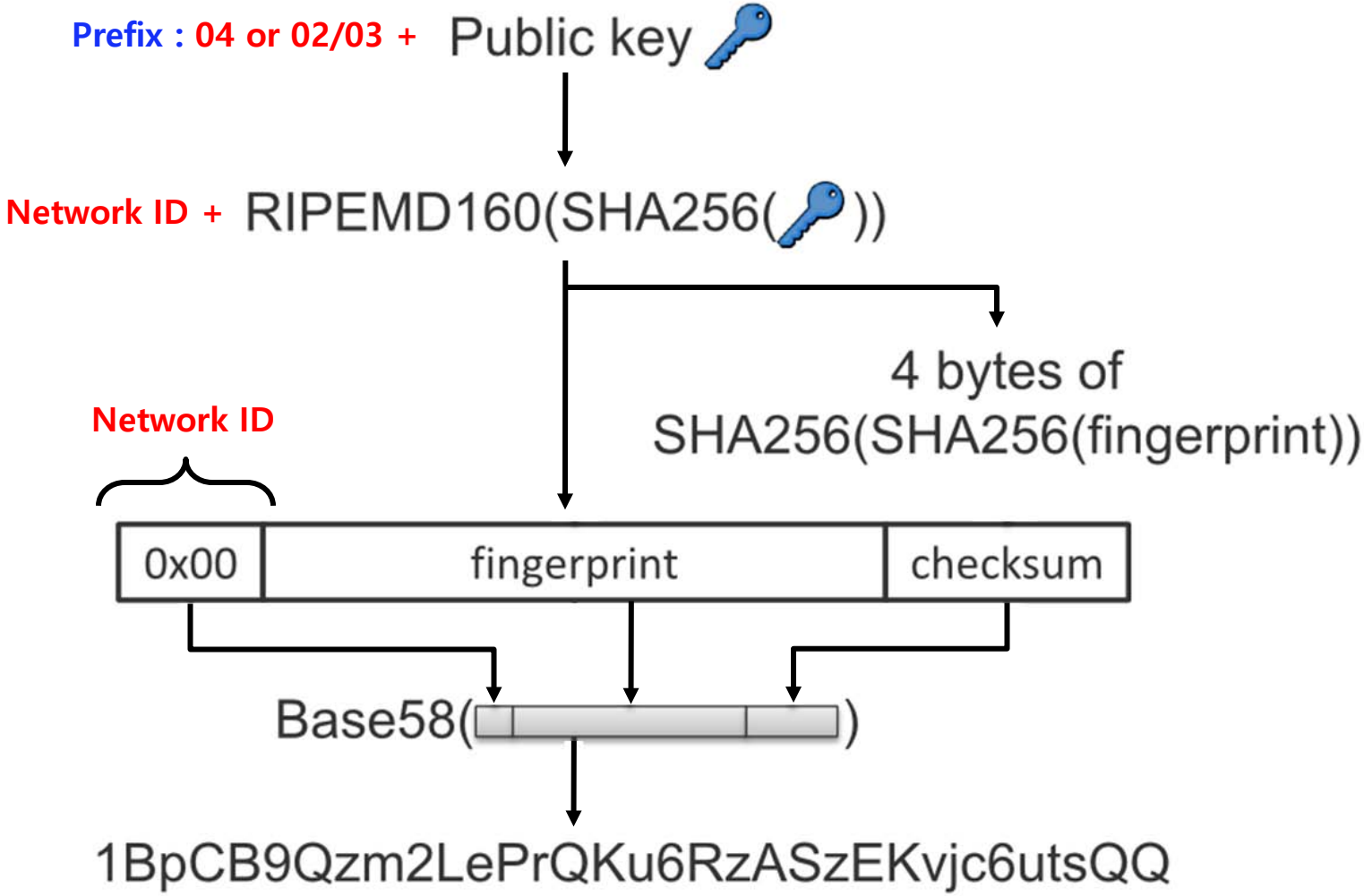
- Base58 (Base10, Base16, Base58, Base64) ?

  1 2 3 4 5 6 7 8 9 0
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  a b c d e f g h i j k l m n o p q r s t u v w x y z

- 0, I, O, l

- https://appdevtools.com/base58-encoder-decoder
- https://www.browserling.com/tools/base58-encode
- https://digitalbazaar.github.io/base58-spec/

- sha256
  https://emn178.github.io/online-tools/sha256



✝ **Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)**

WIF (Wallet Import Format) Format

- Standard format to represent Private Key
- Base58 (Prefix + private Key + Compression Byte + Checksum)
  - Version Byte Prefix
    - 0x80 for Mainnet
    - 0xEF for Testnet
  - Compression Byte : 01
    - create a Compressed Public Key
  - Checksum <= (Prefix + private Key + Compression Byte)
- In ECDSA (Elliptic Curve Digital Signature Algorithm)
  - Private Key is 256-bit (32-byte)
- To create a WIF format
  - Checksum is added to this key to verify its correctness
  - Encode in Base58

- Example
  - PrivateKey = "ef23 ... 3db2"
  - extended = "80" + PrivateKey + "01"
  - extendedCheckSum = extended + CheckSum(extended)
  - WIF = Base58_encode(extendedCheckSum)

from Private Key to WIF

- https://learnmeabitcoin.com/technical/wif

- Private Key = bbc27228ddcb9209d7fd6f36b02f7dfa6252af40bb2f1cbc7a557da8027ff866
- Add Version info to Private Key
  Version : MainNet = 0x80. TestNet = 0xef
  80 + bbc27228ddcb9209d7fd6f36b02f7dfa6252af40bb2f1cbc7a557da8027ff866
- if Private key use Compressed Public key => Add 0x01 in end
  Val = 80 + bbc27228ddcb9209d7fd6f36b02f7dfa6252af40bb2f1cbc7a557da8027ff866 + 01

- Double hashing with SHA256
  sha256(sha256(val)) =
  f9b7646236ceecd59e09bb8cfdeab26a364a72921a1a2039652a52a389ffa0bd
- finVal = Add first 4 bytes to tail in Val
  80+bbc27228ddcb9209d7fd6f36b02f7dfa6252af40bb2f1cbc7a557da8027ff866+01+f9b76462

- WIF = Base58(finVal) = L3Wh2WPg21MWqzMFYsVC7PeBXcq1ow32KRccRihnTUnAhJaZUvg1
  This is Private Key in Wallet Program

✝ Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)

Mnemonic Code Converter

- https://iancoleman.io/
- https://iancoleman.io/bip39/

Lab

Legacy Bitcoin Address

```
c:\...> pip install cryptos requests

c:\...> python
>>> from cryptos import *
>>> priv = sha256('your brain wallet password')
>>> priv

>>> pub = privtopub(priv)
>>> addr = pubtoaddr(pub)
>>> addr
'1Mfy6uiX3cZc15RmPkLR5hpGR32taCjY4g'

>>> b = Bitcoin()
>>> pub = b.privtopub(priv)
>>> pub
'04dc2e4727338487a6d5d64cf455e98ceac8e0a97ad42ba5abfac9f73b5ce65484fe5d45dd55527a4de849082a0c48
  5eeb72798be80a2ea9822050cec5e9cb9c66'
>>> addr = b.pubtoaddr(pub)
>>> addr
'1Mfy6uiX3cZc15RmPkLR5hpGR32taCjY4g'
```

Send Tx

```
c:\...> pip install cryptos requests
c:\...> python
>>> from cryptos import *
>>> priv = sha256('dkkim')
>>> b = Bitcoin()
>>> fromAddr = 'bc1q8pfc63383al0wxzzx65p5uczw4k9t2f7xuj2hr'
>>> toAddr = b.privtop2wpkh_p2sh(priv)
>>> inputs = b.unspent(fromAddr)
>>> inputs[0]['segwit'] = True
>>> inputs
>>> balance= inputs[0]['value']  // balance = sum(i['value'] for i in inputs)))
>>> fee = 200

>>> outs = [{'value': balance - fee, 'address': toAddr}]
>>> outs

>>> tx = b.mktx(inputs,  outs)
>>> tx

>>> signed = b.signall(tx, priv)
>>> signed

>>> b.pushtx(signed)
```

✝ **Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)**

https://mempool.space/address/bc1q8pfc63383al0wxzzx65p5uczw4k9t2f7xuj2hr
https://www.blockchain.com/explorer/transactions/btc/f5cd50ab5e42286966e2f6cd46b84d54109c5a491a4aa5aeab2afa229c4e14e6

Address bc1q8pfc63383al0wxzzx65p5uczw4k9t2f7xuj2hr

Total received	2,526.81677720 BTC
Total sent	2,501.58185016 BTC
Balance	25.23492704 BTC $704,610

50 of 279 transactions

f5cd50ab5e42286966e2f6cd46b84d54109c5a491a4aa5aeab2afa229c4e14e6 2023-10-09 12:25

3D9Y3uoP1vXjB4uSCu1cJdj4bZXYPm4tCo Multisig 2 of 3	2.31000000 BTC	bc1q27v9ecyf46vsjxj3y42kgjjw2qa… xq0m6sj8	1.25965295 BTC
3D9Y3uoP1vXjB4uSCu1cJdj4bZXYPm4tCo Multisig 2 of 3	3.60000000 BTC	bc1q8pfc63383al0wxzzx65p5uczw4k… f7xuj2hr	25.20000000 BTC

✝ **Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)**

mempool.space

- Bitcoin address
  - Legacy address
  - SegWit address : bc1xxx

✝ **Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)**

Hash and SigningKey

```
>>> import hashlib
>>> hashlib.sha256("Hello".encode()).hexdigest()

>>> from ecdsa import SigningKey

>>> priv = SigningKey.generate()
>>> priv.to_string().hex()
>>> pub = priv.get_verifying_key()
>>> pub.to_string().hex()

>>> sig = priv.sign("Open seminar".encode())
>>> sig

>>> sig.hex()

>>> pub.verify(sig, "Open seminar".encode())
True
```

Send Tx

```
>>> c = Bitcoin()
>>> inputs = c.unspent('1A6LBamAnquzSZdiETrE7XJcakTNFrffEC')
>>> inputs
[{'output': '103d2fb7f4b9bbe62530ea464627237fa6a2983cc42f4d2b4b94ab0b23b13535:0', 'value': 28600}]
>>> balance = sum(list(map(lambda o: o['value'], inputs)))
>>> fee = 1500
>>> outs = [{'value': balance - fee, 'address': '1LLQccvbR37ixxwx7uC5NQyAAf871ATKHG'}]
>>> tx = c.mktx(inputs, outs)
>>> tx
{'locktime': 0, 'version': 1, 'ins': [{'script': '', 'sequence': 4294967295, 'outpoint': {'hash':
  '2dc4276f4b88134b7253286ea9bcbfb9451fdc7bc6ac8f51835e23bd5eee3814', 'index': 1}, 'amount': 30000}],
  'outs': [{'script': '76a914d4160157d591f11dd2b5f6322b3f5c5dc041e0fb88ac', 'value': 28600}]}
>>>
>>> priv = 'your priv'
>>> signed = c.signall(tx, priv)
>>> signed
'01000000011438ee5ebd235e83518facc67bdc1f45b9bfbca96e2853724b13884b6f27c42d010000008a47304402203401acc754
  df15545ad9afa1f640eb7b01a6f0c7ad8cac7a3d5f058f19aef14602207e942e4641821d42deea1b2b4e460b8990d01c073f4f3
  1c1bad27bdac9f10146014104e8d75e253e1c223ede954d6c503f0c321070a8adcaab728163c0806b354a5eeb961318e7bf6e24
  25493345b55433184bcc6adef57c384577227070c3bf5028f3ffffffff01b86f0000000000001976a914d4160157d591f11dd2b
  5f6322b3f5c5dc041e0fb88ac00000000'
>>> c.pushtx(signed)
{'status': 'success', 'data': {'txid': '103d2fb7f4b9bbe62530ea464627237fa6a2983cc42f4d2b4b94ab0b23b13535',
  'network': 'BTC'}}
>>>
```

✝ **Have reverence for God, and obey his commands, because this is all that man was created for (Ecclesiastes 12:13)**

```
from cryptos import *

c = Bitcoin()
from_address = 'address1'                              // Any bitcoin address from blockchain.com
to_address = '1Mfy6uiX3cZc15RmPkLR5hpGR32taCjY4g'     // Created above

inputs = c.unspent(from_address)
balance =  sum([i['value'] for i in inputs])
fee = 1500
to = 'address2'
outs = [{'value': balance - fee, 'address': to_address}]
tx = c.mktx(inputs, outs)

priv = sha256('dkkim')
signed_tx = c.signall(tx, priv)
signed_tx
# send tx to bitcoin node
c.pushtx(signed_tx)
```

Ubuntu-22.04.3

```
$ bitcoin-core.cli getbestblockhash
$ ls -l blocks/blk00000.dat
$ ls -l blocks/blk*

$ bitcoin-core.cli getblockchaininfo
$ cat currentblk
  #!/bin/bash
  cnt=`bitcoin-core.cli getblockcount`
  hash=`bitcoin-core.cli getblockhash ${cnt}`
  timeline=`bitcoin-core.cli getblock $hash | grep '"time"'`
  ltrimtime=${timeline#*time\" : }
  newest=${ltrimtime%%,*}
  echo $((`date +%s`-$newest))
  echo $(((`date +%s`-$newest)/60))

$ bitcoin-core.cli getblockcount

$ cnt=`bitcoin-core.cli getblockcount`
$ hash=`bitcoin-core.cli getblockhash ${cnt}`
$ timeline=`bitcoin-core.cli getblock $hash | grep '"time"'`
$ ltrimtime=${timeline#*time\" : }
$ newest=${ltrimtime%%,*}
```

bitcoin-core.cli or bitcoin-cli

```
$ bitcoin-cli getblockhash 0

  000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

$ bitcoin-cli getblock 000000000019d...26f 2

Where the first argument is the block hash,
and 2 is the verbosity (i.e. for json object with transaction data)
```

bitcoin-core

Bitcoin

- https://bitnodes.io/#join-the-network
- https://bitcoincore.org/en/download/

- https://bitcoin.org/
- https://github.com/bitcoin/bitcoin
- https://bitcoincore.org/en/releases/
- https://bitcoincore.org/en/download/

- Bitcoin source code

```
$ sudo apt-get install git
$ git clone https://github.com/bitcoin/bitcoin.git
$ git tag –l

* git checkout tags/version_tag
$ git checkout tags/v0.21.0
```

bitcoin.org

- Official Bitcoin Website
  - It is associated with the creator of Bitcoin or the Bitcoin Core development team and provides official information regarding Bitcoin updates, releases, and documentation.
- Download Bitcoin Client Software
  - "bitcoin.org" provides download links for Bitcoin Core and other Bitcoin client software.
  - Users can download Bitcoin client software from this site and check for updated versions.

bitcoincore.org

- Bitcoin Core Development Team Website
  - "bitcoincore.org" is the website of the Bitcoin Core development team.
  - This website focuses on providing information about the development, updates, documentation, and related projects of Bitcoin Core.

Bitcoin-core

- Bitcoin Core
  - is the official client software that implements the Bitcoin protocol, allowing users to interact with the Bitcoin network and manage their wallets
  - It serves as the official client software for Bitcoin and is responsible for interacting with the Bitcoin network and maintaining the blockchain
  - Bitcoin Core is a node software that maintains a replica of the Bitcoin network and verifies new transactions
  - It also includes wallet software, allowing individual users to hold and transact with Bitcoin
- Differences between bitcoin and bitcoin-core
  - Bitcoin refers to the cryptocurrency itself, encompassing the protocol, technology, and ecosystem
  - Bitcoin Core is the official client software that implements the Bitcoin protocol, allowing users to interact with the Bitcoin network and manage their wallets

Bitcoin-core

- Bitcoin Core, while serving as a full node software,
  - is often referred to as "client software" because it allows individual users to act as clients on the Bitcoin network
- It provides the essential functions for users
  - to create and manage Bitcoin wallets,
  - send and receive transactions,
  - and interact with the broader Bitcoin network
- It also functions as a full node, contributing to the network's security and validation of transactions

- Bitcoin Core software includes mining functionality
  - Bitcoin Core is a full node software that serves several key roles within the Bitcoin network, and one of those roles is mining
- How Bitcoin Core's mining functionality works
  - Full Node
    - Bitcoin Core functions as a full node, which means it maintains a complete replica of the Bitcoin network, including the blockchain. This is crucial for verifying new transactions and blocks, ensuring the security and trustworthiness of the network.
  - Wallet
    - Bitcoin Core also operates as a wallet software, allowing users to send and receive Bitcoin. Users can manage their Bitcoin holdings and conduct transactions using Bitcoin Core's wallet features.
  - Mining
    - Bitcoin Core includes mining capabilities. Miners can use Bitcoin Core to mine new blocks and include new transactions in those blocks. This process is how new Bitcoin is issued, and miners earn rewards for mining blocks as well as collecting transaction fees.

Build bitcoin source

- **`git clone https://github.com/bitcoin/bitcoin.git -b branch_name --single-branch`**
- **`git clone https://github.com/bitcoin/bitcoin.git -b release/v0.21.0 --single-branch`**

- `cd bitcoin`
- `./autogen.sh`
- `./configure`
- `make`
- `sudo make install`
- `./src/bitcoind`

Install bitcoin

- Method 1

```
$ curl https://bitnodes.io/install-full-node.sh
```

- Method 2

```
$ sudo apt update
$ sudo apt install build-essential autoconf automake libtool libevent-dev libboost-all-dev libminiupnpc-dev libssl-dev libzmq3-dev libqt5gui5 libqt5core5a libqt5dbus5 qttools5-dev qttools5-dev-tools libqrencode-dev
$ wget https://bitcoin.org/bin/bitcoin-core-x.y.z/bitcoin-x.y.z-x86_64-linux-gnu.tar.gz
$ sha256sum bitcoin-x.y.z-x86_64-linux-gnu.tar.gz
$ tar -xvf bitcoin-x.y.z-x86_64-linux-gnu.tar.gz
$ sudo mv bitcoin-x.y.z /usr/local/bin/bitcoin
$ bitcoin
```

https://bitcoin.org/en/bitcoin-core/

- https://bitcoin.org/en/bitcoin-core/
- https://bitcoin.org/en/download