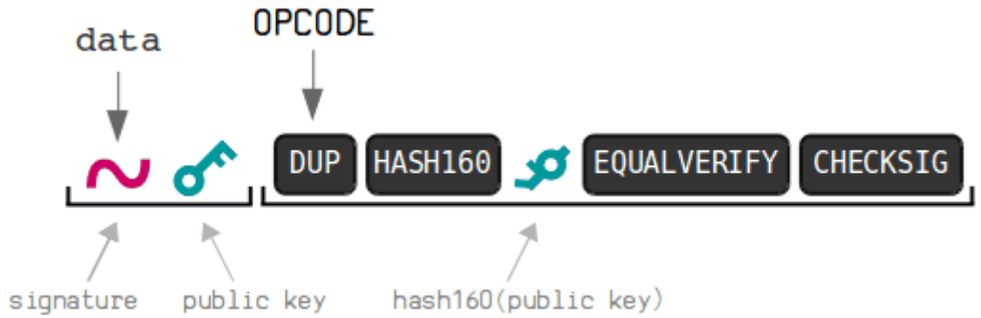
Decentralized Systems

Bitcoin (cnt)

Bitcoin Script

(a language with no name)



Valid payments

- To ensure the **correctness of a transaction**, each node verifies
  - its input Bitcoins have not been spent yet (UTXO)
  - the sender’s signature

- To **automatically control** the transaction correctness, each node carries out the **transaction script**

Bitcoin Script

- Specialized **programming language** used in Bitcoin to define the rules for spending and validating transaction
- **Automatically processed** by peers (wallets, miners, and other nodes behind the scene)
- **Stack-based**: every instruction is executed exactly once

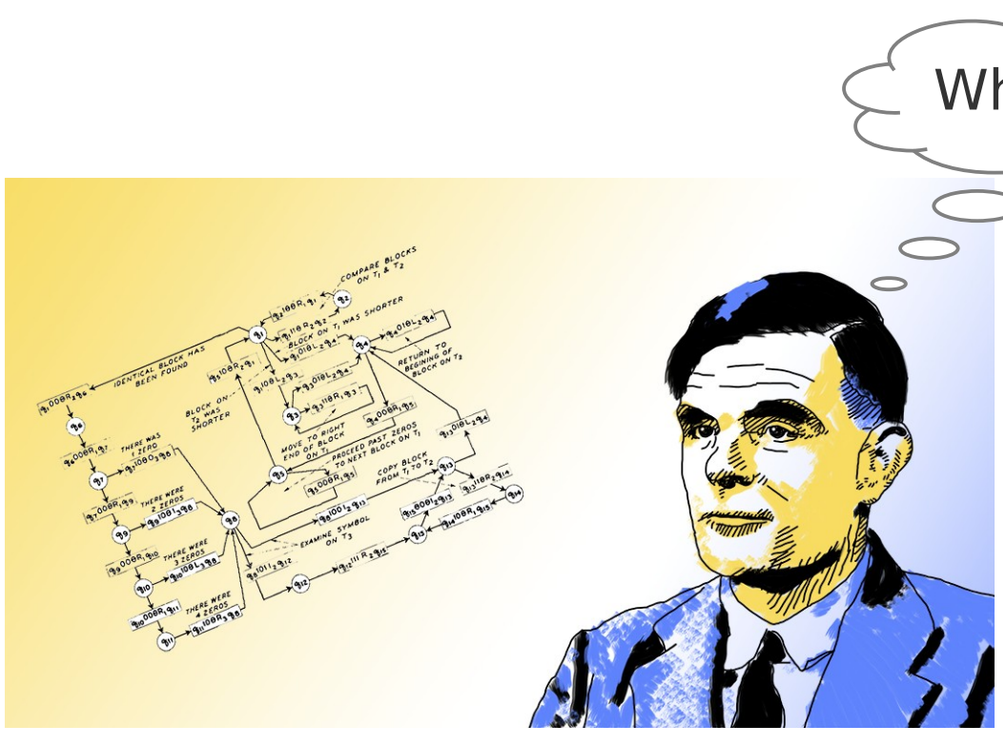
Bitcoin Script

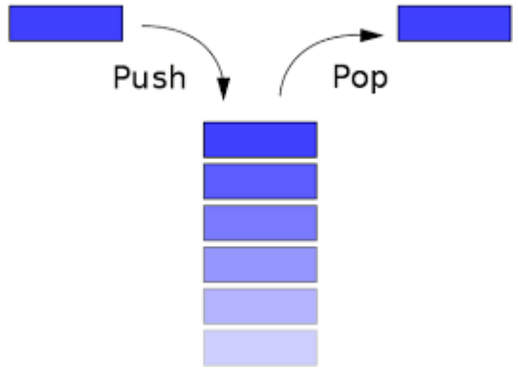
- Used to **define the conditions that must be met** in order to spend a particular UTXO
  - requires the **recipient** to provide a **valid signature** with the private key corresponding to the public key associated with the output
  - implements a variety of features, such as **multi-signature** wallets, **escrow** payments, and **time-locked** transactions

https://en.bitcoin.it/wiki/Script

Bitcoin Script

- No Turing complete (no infinite loops or complex logic)

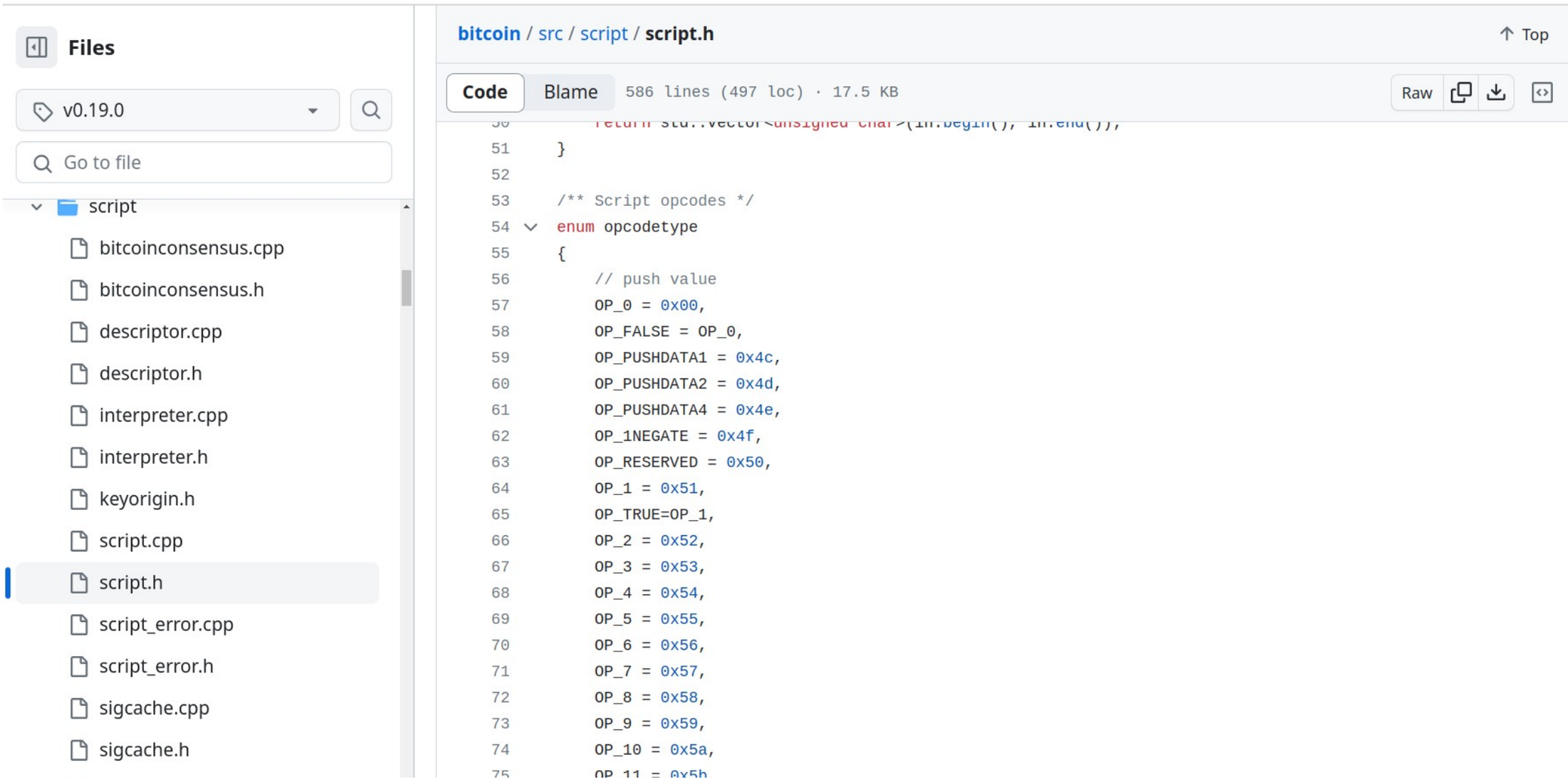




Bitcoin Script

- Design goals
  - simple, compact, support for cryptography
- Composed of
  - **Data**: for example public keys and signatures
  - **Opcodes**: instructions (there is room only for 256 opcodes)
    - basic arithmetic (+, -, *, ...)
    - basic logic, returning earlier, etc...
    - special-purpose instructions to compute and verify signatures

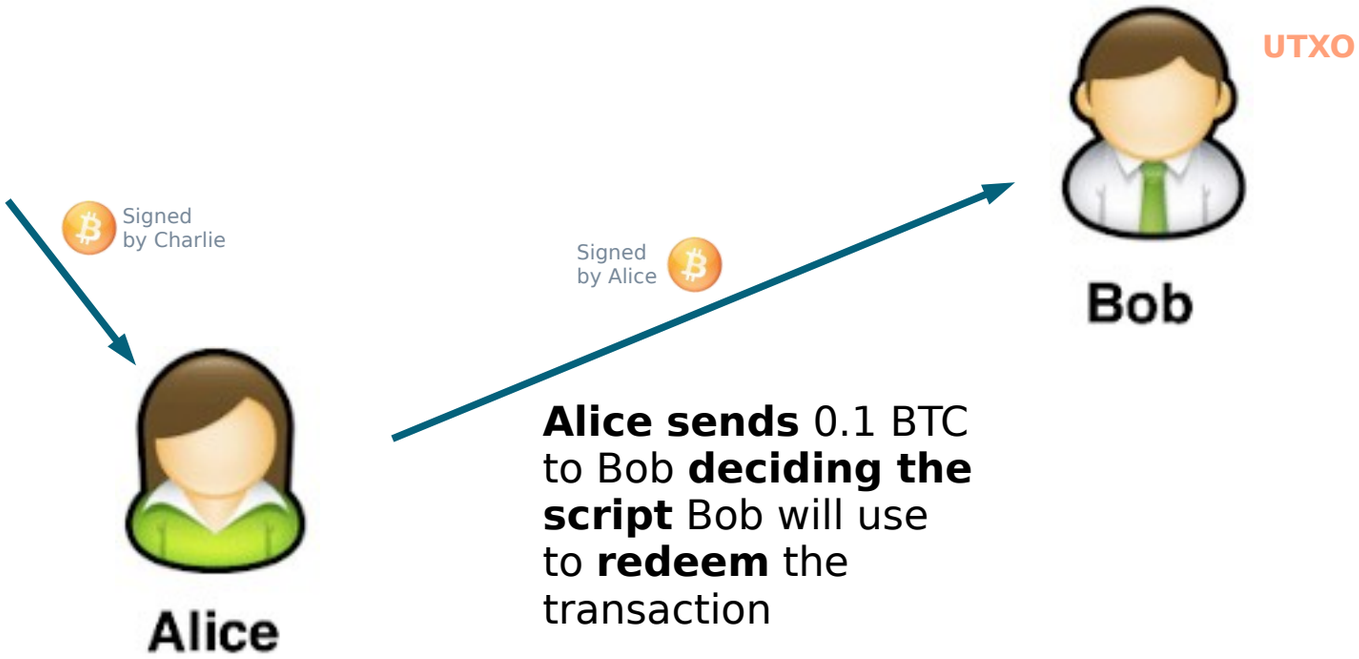
Bitcoin Script



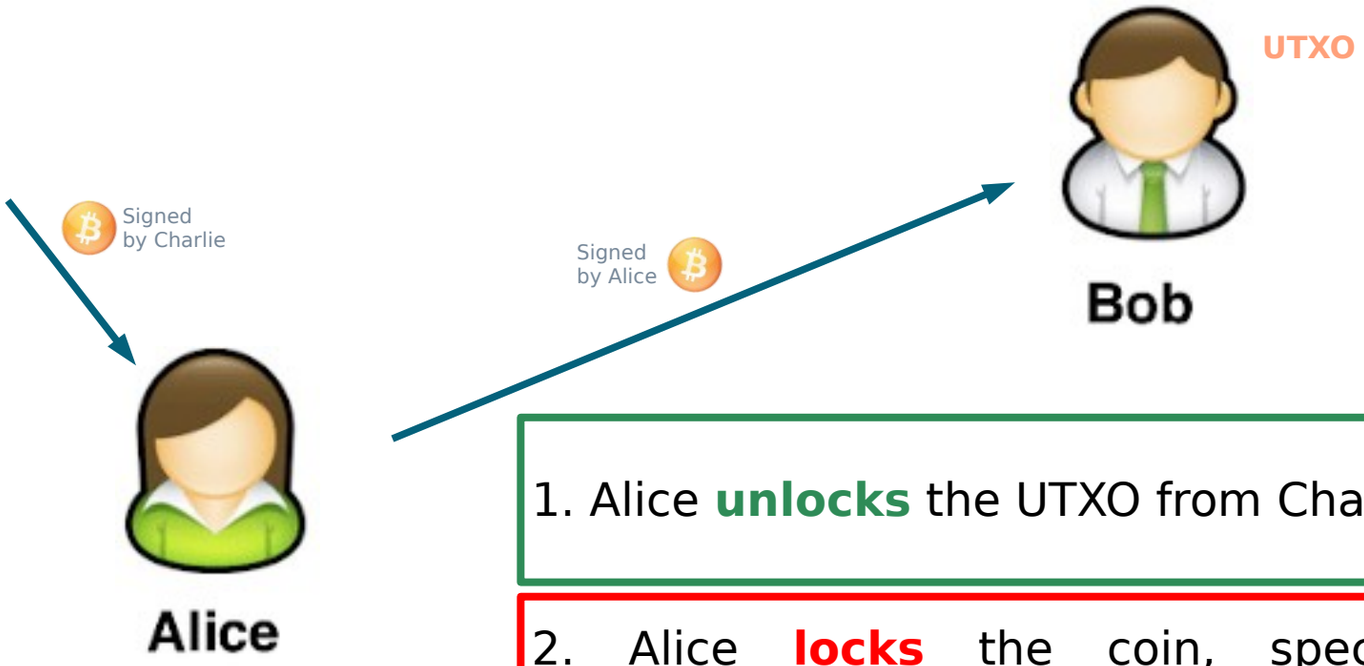
https://github.com/bitcoin/bitcoin/blob/v0.19.0/src/script/script.h
https://en.bitcoin.it/wiki/Script

Bitcoin Script

Bitcoin Script



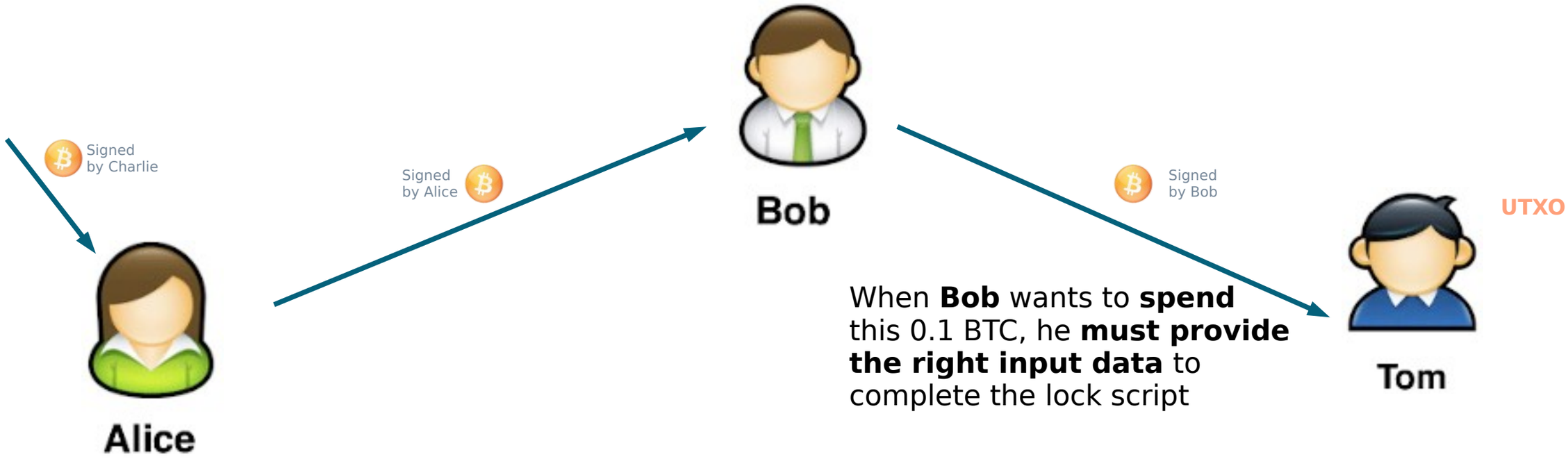
Bitcoin Script



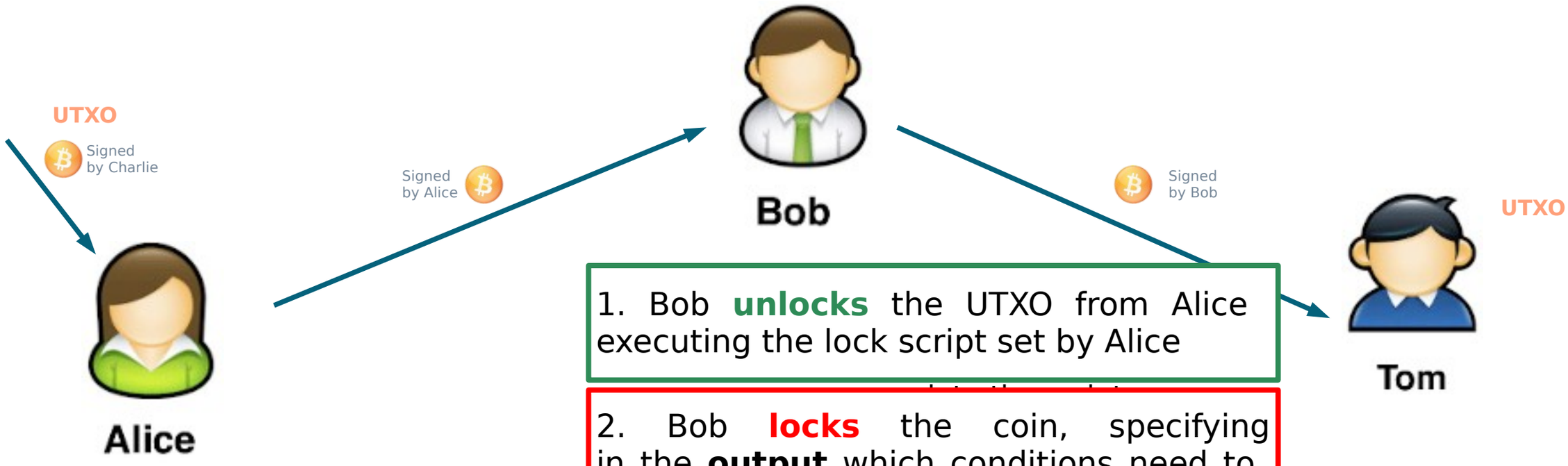
1. Alice **unlocks** the UTXO from Charlie

2. Alice **locks** the coin, specifying in the **output** which conditions need to be met to spend the same coin in the future

Bitcoin Script



Bitcoin Script



Bitcoin Script

- Every **output** is given a **locking script**
- **Data** must be provided in the **input** for executing the **unlocking script** to reedem and spend coins
- The two scripts **are concatenated** and, if the **resulting script runs with success**, the **transaction is valid**

Bitcoin Script

- Most transactions use the same **small set of simple instructions**
- Bitcoin nodes have a **list of standard scripts** and they refuse to accept scripts that are not in the list
- This makes the Bitcoin script language inherently **more secure** due to the limited number of operations that it can perform

Pay-to-PubKey-Hash (P2PKH)

- P2PKH allows a Sender (Alice) to send coins to a Bitcoin address (one of the most used script before SegWit)
- To spend the coins (redeem) the Receiver (Bob) must show the ownership

Unlock script (ScriptSig)

3044022041c09a16ee9db2315d09df490d0b4ba
3b34f40b148fe39bd756b93dffe27b31802204c1
230b9415657c176369a7556283398b7552039f2
a5cd3bfb17a22be0a77b0f01

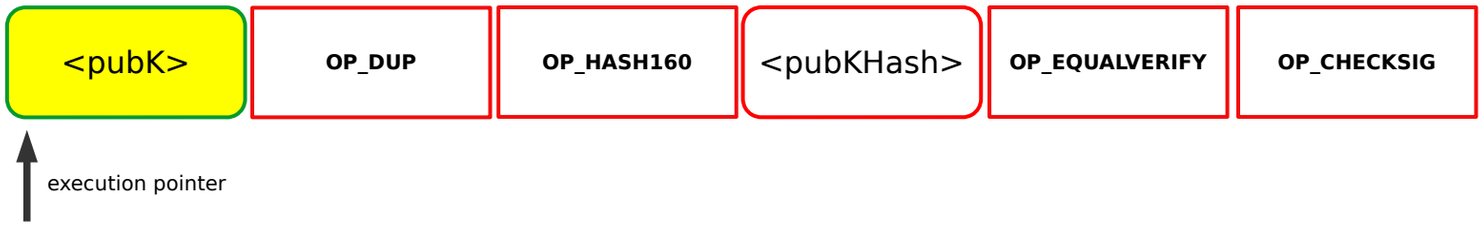
02f5e548d2ab03cb235fc979c8cce56620fd0911
4362f926f0cc31c7e317f36e91

Lock script (ScriptPubKey)

OP_DUP
OP_HASH160
7b134de21ecf69a197ae05305e5264a691b58753
OP_EQUALVERIFY
OP_CHECKSIG

Pay-to-PubKey-Hash (P2PKH)

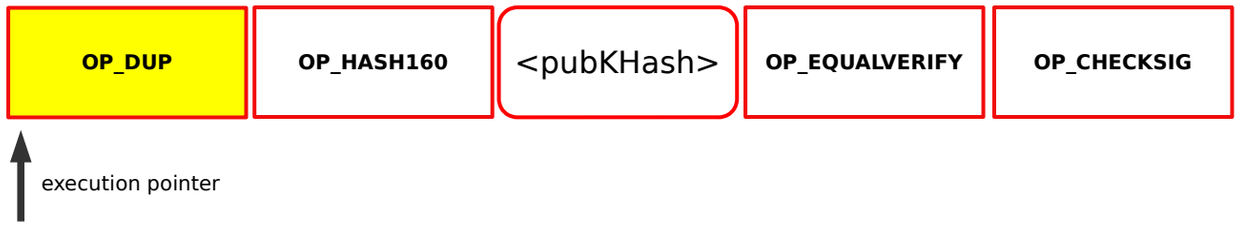
Pay-to-PubKey-Hash (P2PKH)



3044022041c09a16ee9db2315d09df490d0b4ba3b34f40b148fe39bd756b93dffe27b31
802204c1230b9415657c176369a7556283398b7552039f2a5cd3bfb17a22be0a77b0f01

Provided by Bob

Pay-to-PubKey-Hash (P2PKH)

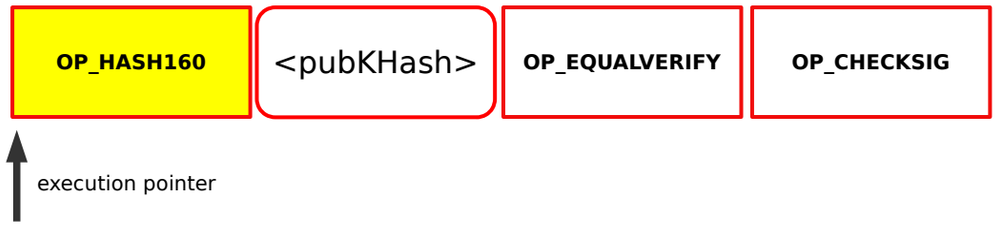


02f5e548d2ab03cb235fc979c8cce56620fd09114362f926f0cc31c7e317f36e91

3044022041c09a16ee9db2315d09df490d0b4ba3b34f40b148fe39bd756b93dffe27b31
802204c1230b9415657c176369a7556283398b7552039f2a5cd3bfb17a22be0a77b0f01

Provided by Bob

Pay-to-PubKey-Hash (P2PKH)



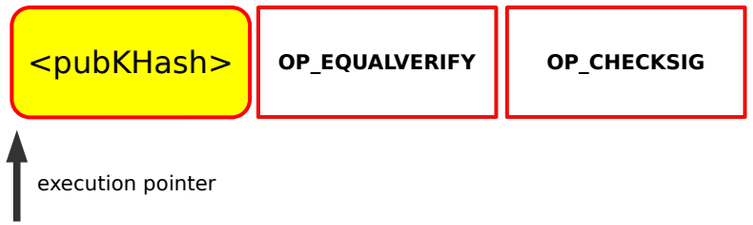
02f5e548d2ab03cb235fc979c8cce56620fd09114362f926f0cc31c7e317f36e91

Result of OP_DUP

02f5e548d2ab03cb235fc979c8cce56620fd09114362f926f0cc31c7e317f36e91

3044022041c09a16ee9db2315d09df490d0b4ba3b34f40b148fe39bd756b93dffe27b31
802204c1230b9415657c176369a7556283398b7552039f2a5cd3bfb17a22be0a77b0f01

Pay-to-PubKey-Hash (P2PKH)



The input is hashed twice: first with SHA-256 and then with RIPEMD-160, e.g., Bitcoin address

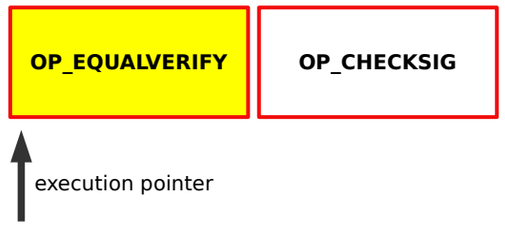
7b134de21ecf69a197ae05305e5264a691b58753

Result of OP_HASH160

02f5e548d2ab03cb235fc979c8cce56620fd09114362f926f0cc31c7e317f36e91

3044022041c09a16ee9db2315d09df490d0b4ba3b34f40b148fe39bd756b93dffe27b31
802204c1230b9415657c176369a7556283398b7552039f2a5cd3bfb17a22be0a77b0f01

Pay-to-PubKey-Hash (P2PKH)



7b134de21ecf69a197ae05305e5264a691b58753

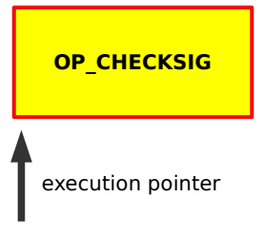
7b134de21ecf69a197ae05305e5264a691b58753

02f5e548d2ab03cb235fc979c8cce56620fd09114362f926f0cc31c7e317f36e91

3044022041c09a16ee9db2315d09df490d0b4ba3b34f40b148fe39bd756b93dffe27b31
802204c1230b9415657c176369a7556283398b7552039f2a5cd3bfb17a22be0a77b0f01

Provided by Alice in the lock script when sending coins to Bob’s address

Pay-to-PubKey-Hash (P2PKH)



Result of
OP_EQUALVERIFY

02f5e548d2ab03cb235fc979c8cce56620fd09114362f926f0cc31c7e317f36e91

3044022041c09a16ee9db2315d09df490d0b4ba3b34f40b148fe39bd756b93dffe27b31
802204c1230b9415657c176369a7556283398b7552039f2a5cd3bfb17a22be0a77b0f01

Pay-to-PubKey-Hash (P2PKH)

A script is valid if the top and only element left on the stack is a 1

1

Result of OP_CHECKSIG

Pay-to-PubKey-Hash (P2PKH)

A script is invalid if

- The final stack is empty
- The top element on the stack is 0
- There is more than one element left on the stack at the end of execution
- The script exits prematurely (e.g. OP_RETURN)

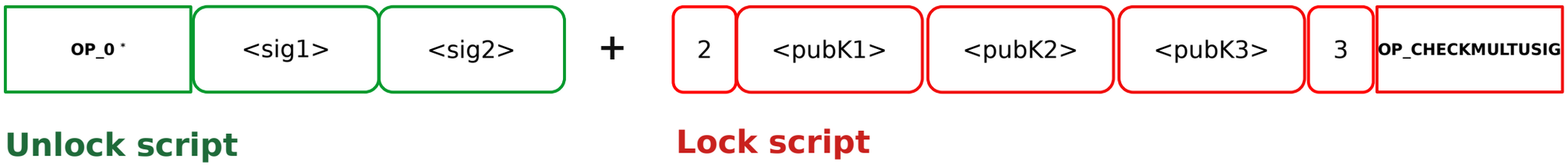
0 Result of OP_CHECKSIG

Pay-to-Pubkey (P2PK)

- P2PK (Pay To Pubkey) is a script pattern that locks an output to a **public key** (which is longer than a Bitcoin address)
- P2PK can be found in Coinbase transactions in the earlier blocks in the blockchain
- Other scripts can be found in the Coinbase, if OP_1 appears in the output script, it is likely associated with SegWit outputs...

Pay-to-Multisig (P2MS)

- Third party (**escrow**/arbiter) may optionally be involved in transactions
- To unlock a P2MS script, a **required number of signatures** (2) need to be provided among a set of specified public keys (3)
- Few locking scripts are P2MS, so they can be hard to find



* OP_0 is required because of a bug in the original implementation

Pay-to-Multisig (P2MS)

- Third party (**escrow**/arbiter) may optionally be involved in transactions

OP_CHECKMULTISIG	174	0xae	x sig1 sig2 ... <number of signatures> pub1 pub2 <number of public keys>	True / False	Compares the first signature against each public key until it finds an ECDSA match. Starting with the subsequent public key, it compares the second signature against each remaining public key until it finds an ECDSA match. The process is repeated until all signatures have been checked or not enough public keys remain to produce a successful result. All signatures need to match a public key. Because public keys are not checked again if they fail any signature comparison, signatures must be placed in the scriptSig using the same order as their corresponding public keys were placed in the scriptPubKey or redeemScript. If all signatures are valid, 1 is returned, 0 otherwise. Due to a bug, one extra unused value is removed from the stack.

Unlock script **Lock script**

* OP_0 is required because of a bug in the original implementation

NULL DATA (OP_RETURN)

- NULL DATA is a standard **locking script** that can be used to **store data** on the blockchain
- The instruction OP_RETURN
  - returns immediately an error
  - instructions after OP_RETURN are not processed
- The data after OP_RETURN could be at maximum of 80 bytes

NULL DATA (OP_RETURN)

- Any output with a OP_RETURN on it is **unspendable**
- 80 bytes are sufficient for a hash function output (32 bytes for SHA-256)
- The NULL DATA script is used to **timestamp public and immutable data** in the blockchain

Proof-of-burns

- All Bitcoin transactions with the OP_RETURN instruction in their outputs can not be redeemed
- That is equivalent to **burning** coins!
- But it is possible to take advantage of the ash and store data permanently on the blockchain

https://en.bitcoin.it/wiki/OP_RETURN

Script types today

Output Type Volume (90 Days)	?
Pubkey Hash	33.11%
Script Hash	9.69%
SegWit v0 Pubkey Hash	43.64%
SegWit v0 Script Hash	12.68%
Taproot	0.94%

Output Type Counts (90 Days)	?
Bare Multisig	341,730
Non-standard	5
Nulldata (OP_RETURN)	33,218,115
Pubkey	27
Pubkey Hash	9,466,300
Script Hash	9,280,866
SegWit v0 Pubkey Hash	58,596,281
SegWit v0 Script Hash	1,894,231
SegWit Unknown Version	7
Taproot	23,141,129

https://bitcoin.clarkmoody.com/dashboard/

Do you remember RFC?

Bitcoin Improvement Proposals

- In Bitcoin we have BIP (https://en.bitcoin.it/wiki/BIP_0001)
- A BIP is a **design document** providing **information** to the Bitcoin community, or describing a **new feature** for Bitcoin or its **processes or environment**. The BIP should provide a concise technical specification of the feature and a rationale for the feature
  - A **Standards Track BIP** describes any change that affects most or all Bitcoin implementations [...]

Hard fork

- Changes in the Bitcoin protocol can lead to
  - **Hard fork**
  - **Soft fork**
- A **hard fork** is a **radical change** to the protocol that makes previously valid blocks/transactions invalid (or vice-versa)
- A hard fork requires all nodes or users to upgrade to the latest version of the protocol software

Hard fork

- Nodes of the **newest version of a blockchain no longer accept the older version(s)** of the blockchain and this creates a **permanent divergence** from the previous version of the blockchain
- Generally, after a short time, those on the old chain will realize that their version of the blockchain is outdated and quickly upgrade to the latest version

Soft fork

- A **soft fork** is a change to the software protocol which is **backward compatible**
- Only a majority of the nodes upgrade to enforce the new rules, as opposed to a hard fork that requires all nodes to upgrade and agree on the new version
- The **blockchain accepts the new rules** and, therefore, accepts both the updated blocks and the old blocks of transactions at the same time

New scripts to solve some issues

The issue of scalability

- Visa more than 1700 transactions per sec
- Bitcoin 4-7 transactions per sec (average 4.6)

https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44

Segregated witness (SegWit)

- **Protocol upgrade** (proposed in Dec 2015, BIP 141, adopted in Aug 2017) intended to **increase block capacity**
- It addresses also **malleability** of transactions

Malleability

- Bitcoin code allows **digital signatures to be altered** while a transaction is **still waiting to be confirmed**
- The **signature alteration** can be done in such a way that, by running a mathematical check, the **signature is still valid** for the network, but…
- … its **hash is different**!
- And hash values identify transactions in the network

Malleability

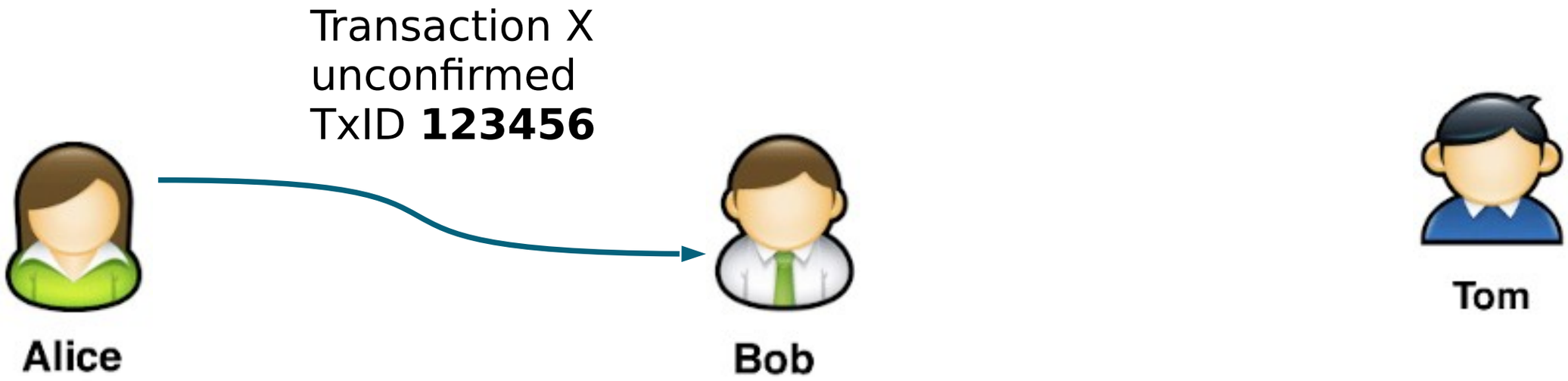
Signature

Mathematical value

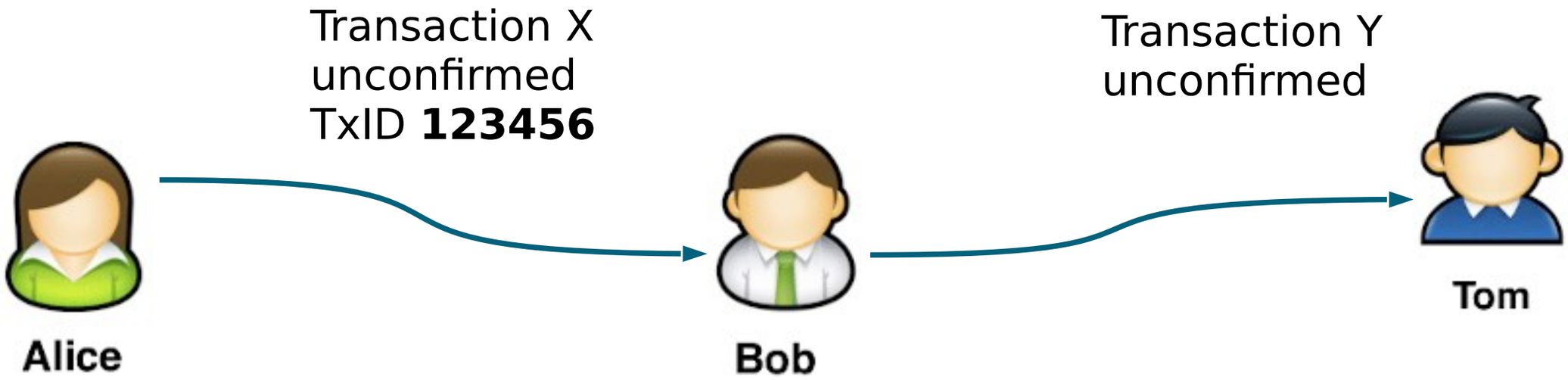
Hash value (new TxID!)

Original Tx	Altered Tx
3	7-4
3	3
1121cfccd5913f0 a63fec40a6ffd44 ea64f9dc135c66 634ba001d10bcf 4302a2	48fd8983f9799e 20d3ff8813dfe4 8e38ed1bd20dc 0514894357cc2 bdf141a0da

Malleability

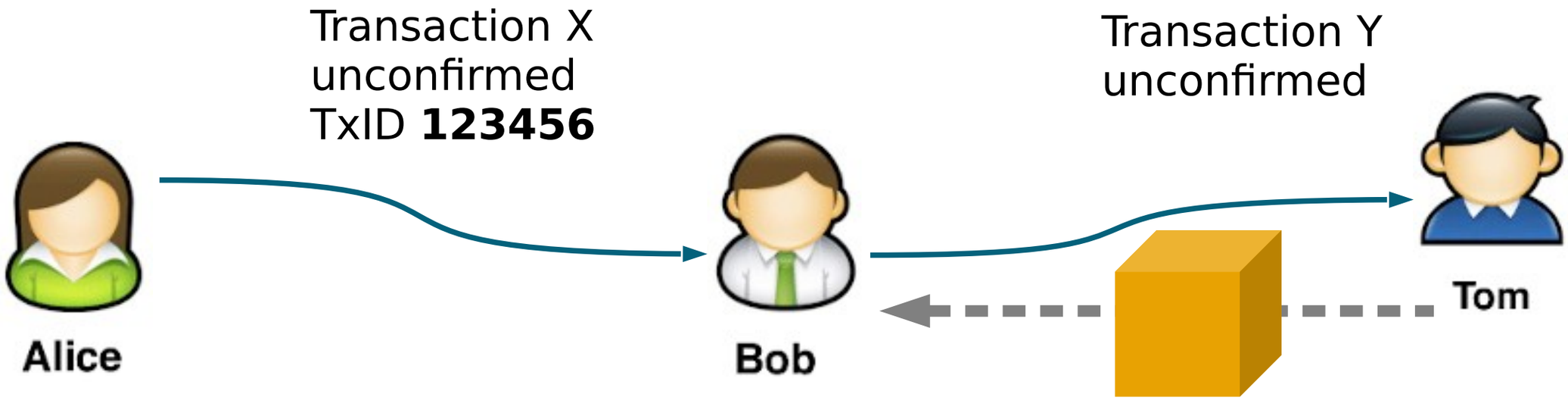


Malleability

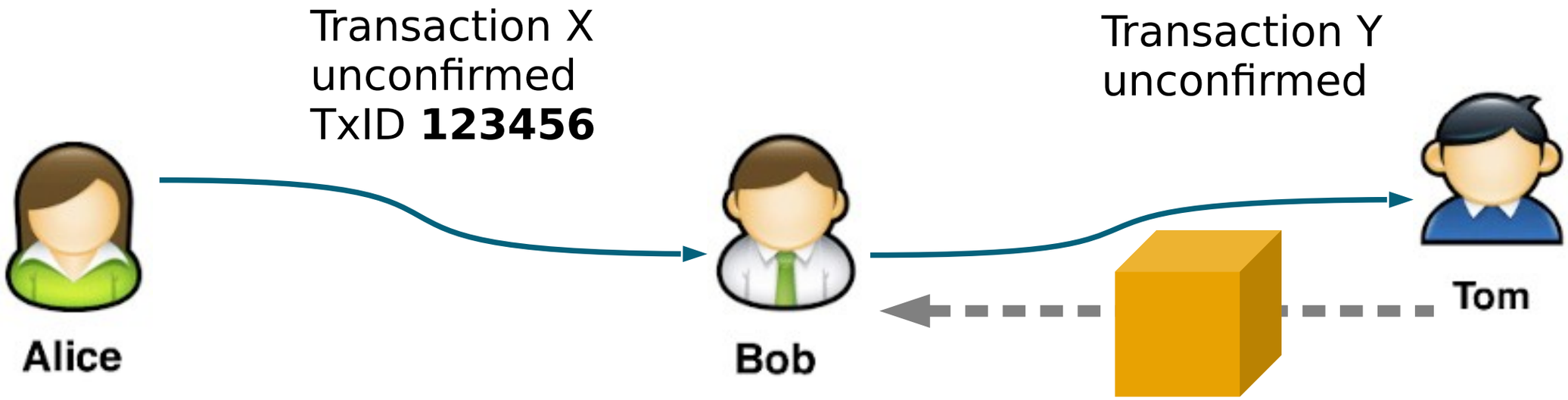


Bob **buys a product** from Tom with the same **unconfirmed transaction**

Malleability

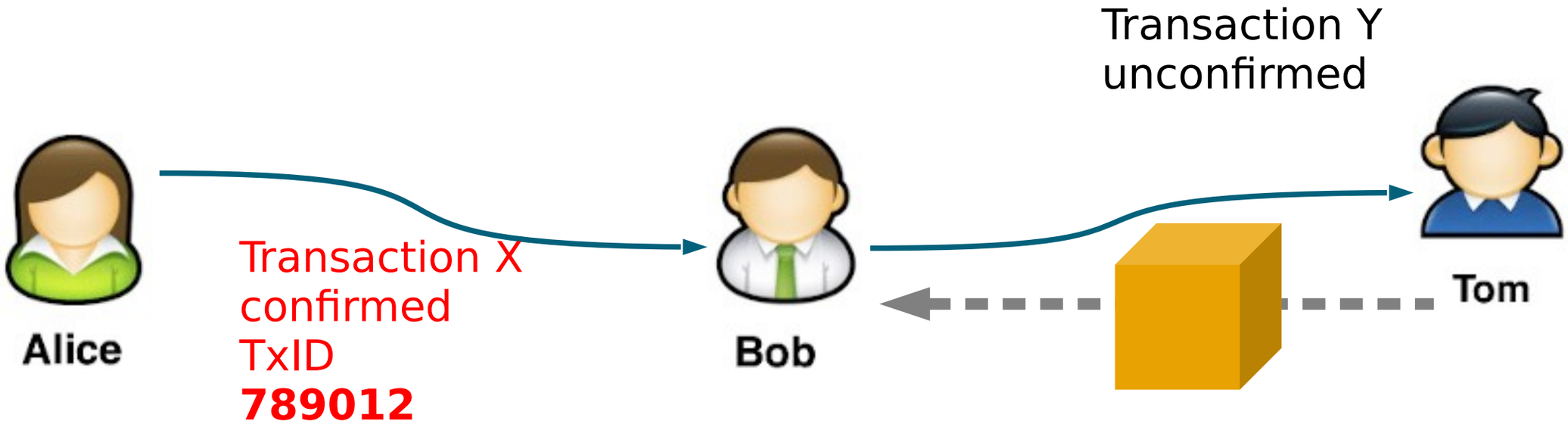


Malleability

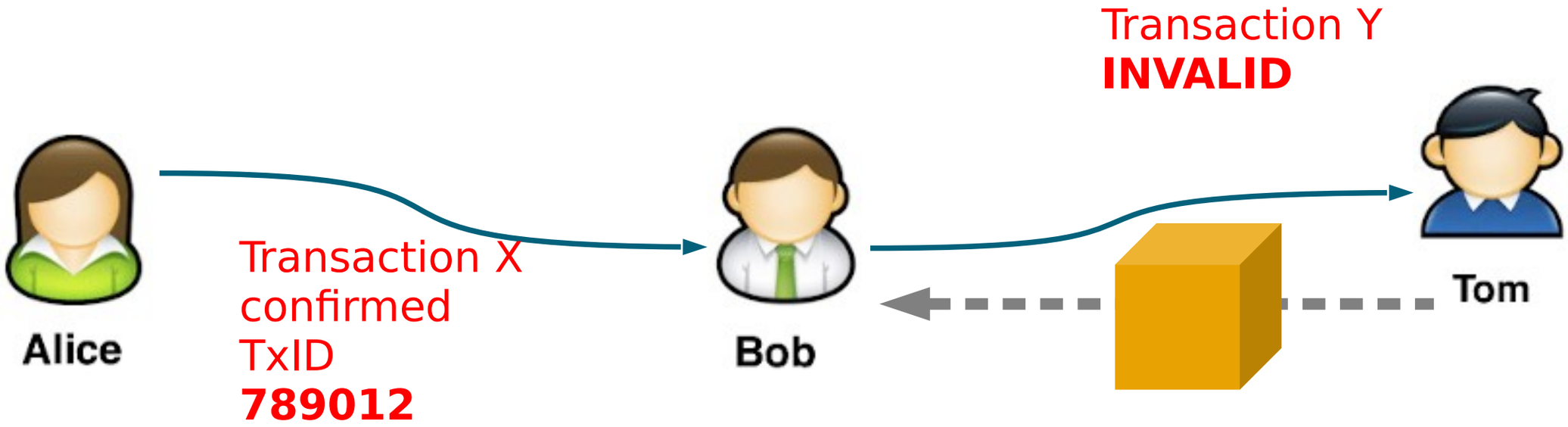


After receiving the product, Bob maliciously changes (**malleates**) **Alice's payment** so that her transaction gets confirmed but with a **different TxID**

Malleability



Malleability

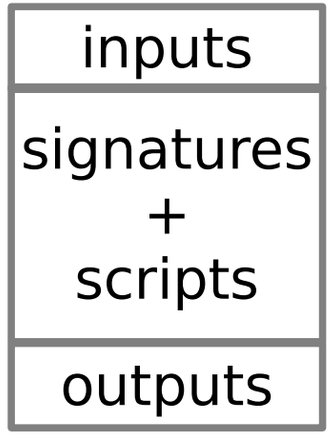


Transaction Y is now **invalid**, since it relies on TxID 123456 which no longer exists and **Tom will not get paid**

https://en.wikipedia.org/wiki/Transaction_malleability_problem

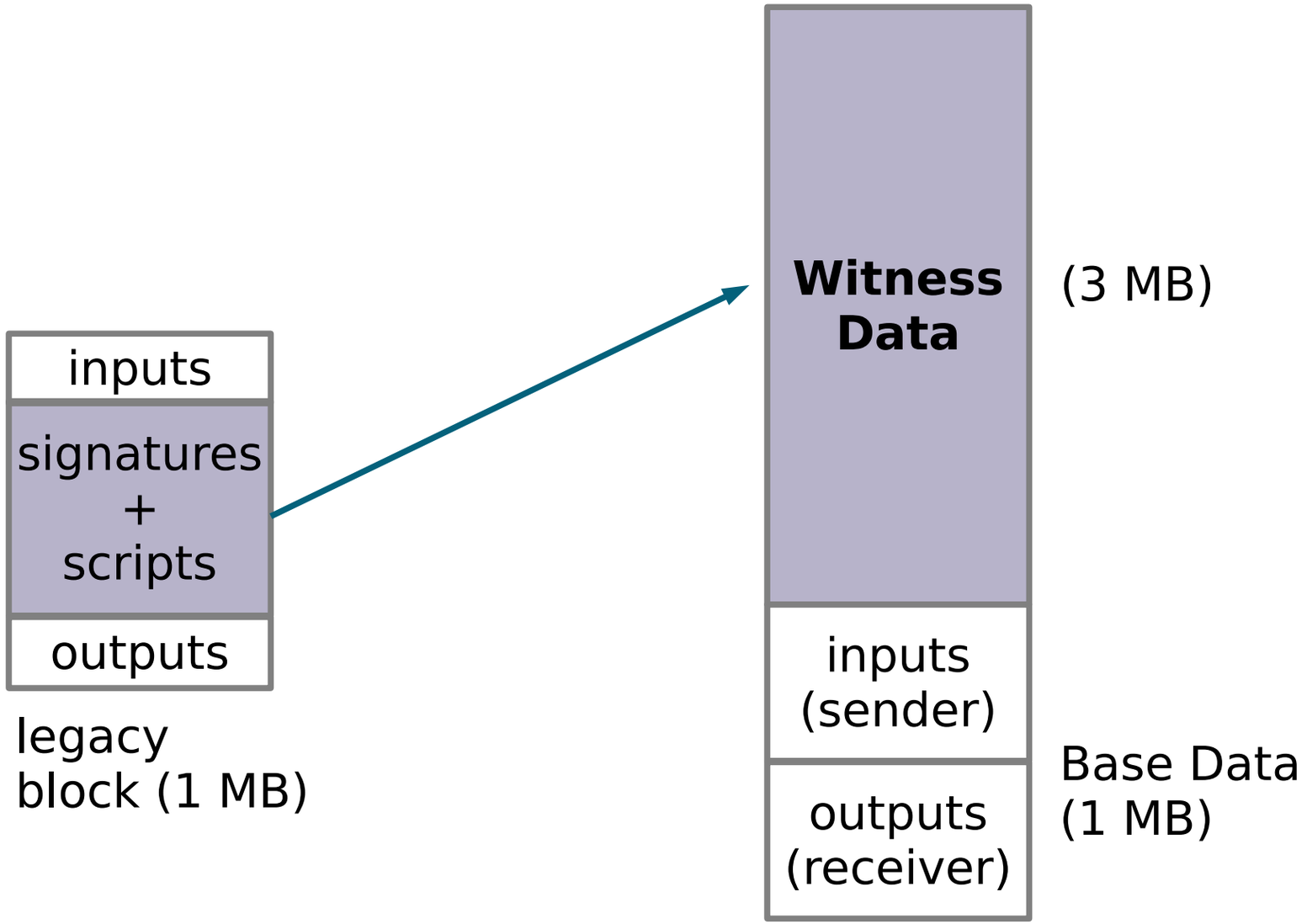
Segregated Witness (SegWit)

- Legacy blocks 1 MB, but
  - 90% of the space is used by transactions
  - 65% of the space of the transactions is used by signatures

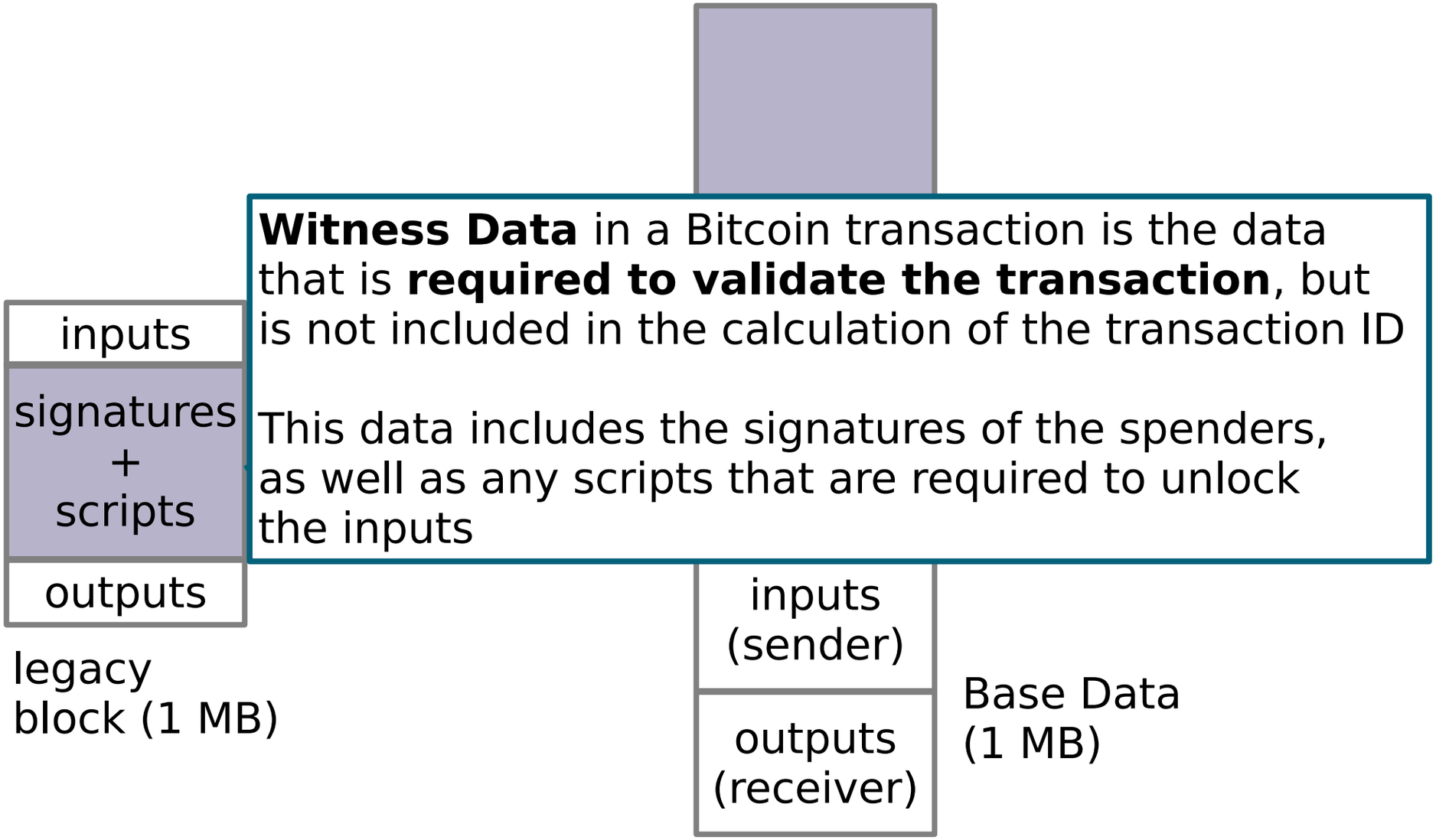


- What about separating (**segregating**) signatures (**witness data**) from the rest of the transaction data?

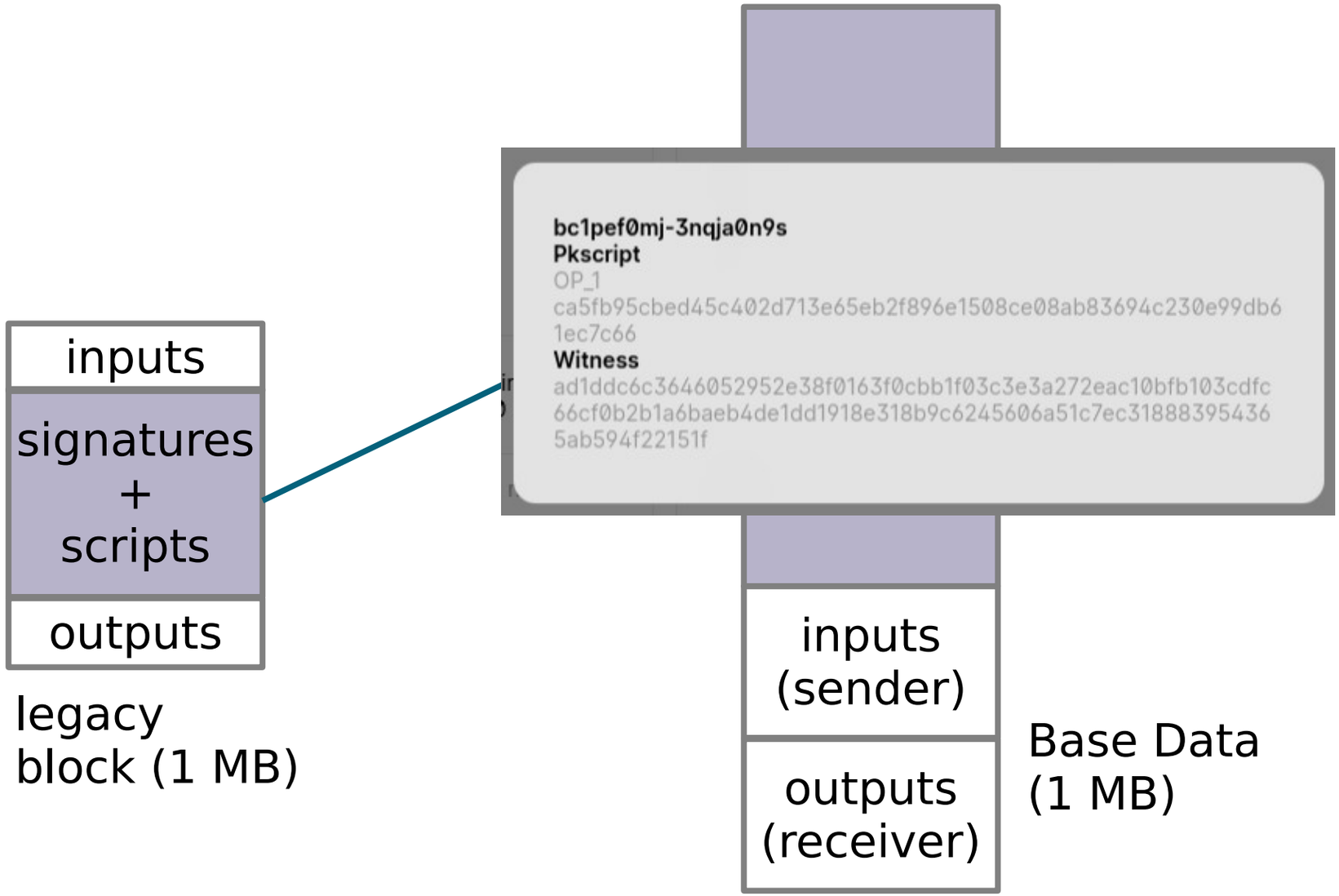
Legacy vs SegWit



Legacy vs SegWit



Legacy vs SegWit



SegWit

- Increases block capacity and it is **compatible** with the original Bitcoin protocol, e.g., is a **soft fork**
- The network accepts both SegWit and Non SegWit blocks
  https://transactionfee.info/charts/transactions-spending-segwit/

- Pay-to-Witness-Pubkey-Hash (P2WPKH) and Pay-to-Witness-Script-Hash (P2WSH)

SegWit

- **Pros**
  - more transactions can fit in a 1 MB block, **more scalable**
  - addresses also **malleability**, by moving the signature outside the base transaction block, changes in the transaction signature will not affect TxID
  - SegWit enables **second layer scaling solutions**
- **Cons**
  - Legacy wallets do not support SegWit (addresses start with bc1, legacy addresses start with 1)

Measuring blocks

Legacy blocks	SegWit blocks
Measured in **size**	Measured in **weight**

Measuring blocks

Legacy blocks

Measured in **size**

Instead of simply increasing the block size to accommodate more transactions, Bitcoin developers introduced the concept of **block weight** to ensure backward compatibility with the existing 1 MB block size limit while allowing for more efficient data usage

Measuring blocks

Block Weight = (3 * Base Size) + Total Size

Legacy transaction

No ability to strip witness data
the weight is always:
4 * Total Size

SegWit transaction

Can strip witness data, the
weight is always:
less than 4 * Total Size

Miners favor SegWit
transactions

Measuring blocks

Block Weight = (3 * Base Size) + Total Size

Legacy transactio

No ability to strip witness
the weight is always:
4 * Total Size

With the introduction of block weight, the block size limit (1 MB) was replaced by a **block weight limit of 4,000,000** weight units (WU)

A block is still restricted to around 1 MB of non-witness data (Base Size), to preserve compatibility with older, non-SegWit nodes

Miners favor SegWit transactions

Taproot

- **Protocol upgrade** activated in Nov 2021 (BIPs 340, 341, 342)
- Improves privacy, scalability and security of the network thanks to
  - **Schnorr signature**, that is smaller, faster, and more efficient than previous ECDSA signature used in Bitcoin
  - **MAST** (Merkelized Abstract Syntax Trees), that allows multiple signatures to be combined into a single signature, without revealing whether the signature is a single signature or a multisignature

Taproot

- Collection of **new opcodes** which extends Bitcoin Script
- Taproot, like SegWit, is implemented as a **soft fork**

- **Pay-to-Taproot** (P2TR) is the name of the scripts which are compatible with this upgrade
  - they are smaller and therefore cheaper in term of space occupied on the blockchain

Reduce consumption?

CLIMATE, ENERGY

Cryptocurrency's Dirty Secret: Energy Consumption

BY JEREMY HINSDALE | MAY 4, 2022 Comments

Though skeptics may characterize cryptocurrency as "fake money," "worse than tulip bulbs," or a "greater fool" scheme, it is a very real business. The market capitalization of the almost 19,000 cryptocurrencies in circulation is currently around $1.75 trillion — about the same as the gross domestic product of Italy, the world's eighth largest economy. Even though you might not be able to buy a loaf of bread with Bitcoin at the corner store, many investors are putting a lot of legal tender money into cryptocurrencies.

But crypto has a dirty little secret that is very relevant to the real world: it uses *a lot* of energy. How much energy? Bitcoin, the world's largest cryptocurrency, currently consumes an estimated 150 terawatt-hours of electricity annually — more than the entire country of Argentina, population 45 million. Producing that energy emits some 65 megatons of carbon dioxide into the atmosphere annually — comparable to the emissions of Greece — making crypto a significant contributor to global air pollution and climate change.

And crypto's thirst for energy is growing as mining companies race to build larger

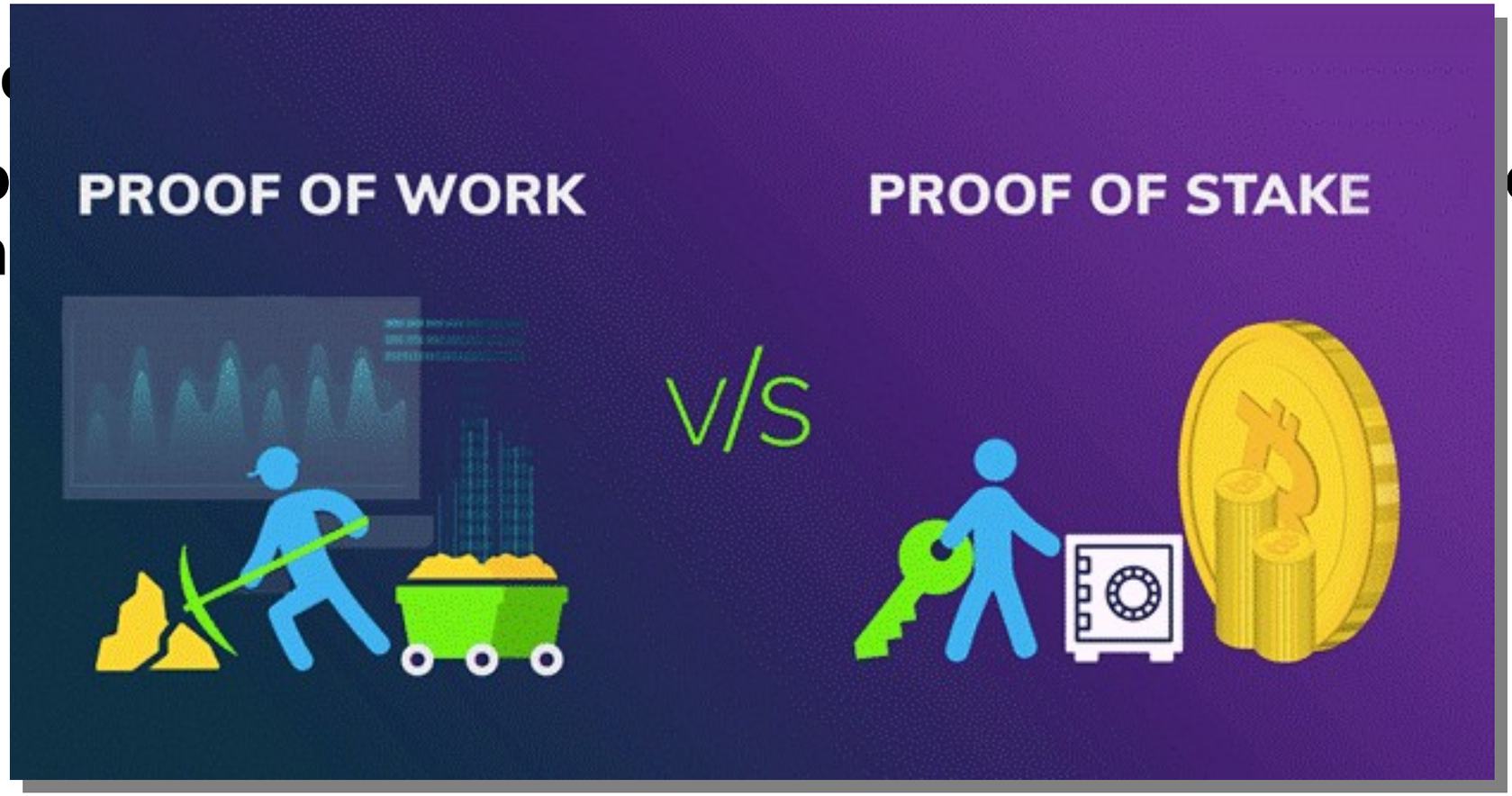
https://news.climate.columbia.edu/2022/05/04/cryptocurrency-energy/

Changing consensus

- Bitcoin Prof-of-Work is the **first consensus algorithm** used in blockchain
  - **Pros**: **anyone can join** the network
  - **Cons**: generally **slow**, requires a **huge amount of energy**

Changing consensus

- Bitcoin Prof-of-Work is the **first consensus algorithm** used in blockchain
  - **Pr**
  - **Co** **of** **en**



Proof-of-Stake

- Alternative consensus mechanism proposed for the firt time in 2012 with Peercoin (https://www.peercoin.net/)
- Blocks are **forged** or **minted** and not mined
- **Validators** not miners
- Two types of PoS
  - Chain-based
  - Consortium consensus

Chain-based PoS

- The leader (**validator)** adds the next block to the blockchain and the other peers agree on it
- **Leader election** is done by considering the **amount of investment** (**stake**) peers have committed to be part of the process **plus some randomness**
- Nodes have an **economic incentive to behave honestly** so as not to devalue the network and their stake in it
- We will see Ethereum