# Bitcoin:
# Programming the Future of Money

Topics in Computer Science - ITCS 4010/5010, Spring 2025

Dr. Christian Kümmerle

Lecture 15

## Bitcoin Transactions

Some figures are taken from:

- “Mastering Bitcoin: Programming the Open Blockchain”, (Andreas Antonopoulos, David Harding), 3rd Edition, O’Reilly, 2023.

# Transactions

# THE STRUCTURE OF A BITCOIN TRANSACTION

Structure of “Legacy” transactions (without any P2WPKH/P2WSH/P2TR inputs or outputs, standard before 2016):

- Version (4 bytes, little-endian)
- Inputs (variable), see breakdown here
- Outputs (variable), see breakdown here
- Locktime (4 bytes, little-endian integer):
  If 499,999,999 or below: Indicates block height after which transaction can be mined
  If 500,000,000 or above: Indicates Linux time after which transaction can be mined

# THE STRUCTURE OF A BITCOIN TRANSACTION

## Structure of “SegWit” transactions (with P2WPKH/P2WSH/P2TR inputs, standard after 2016):

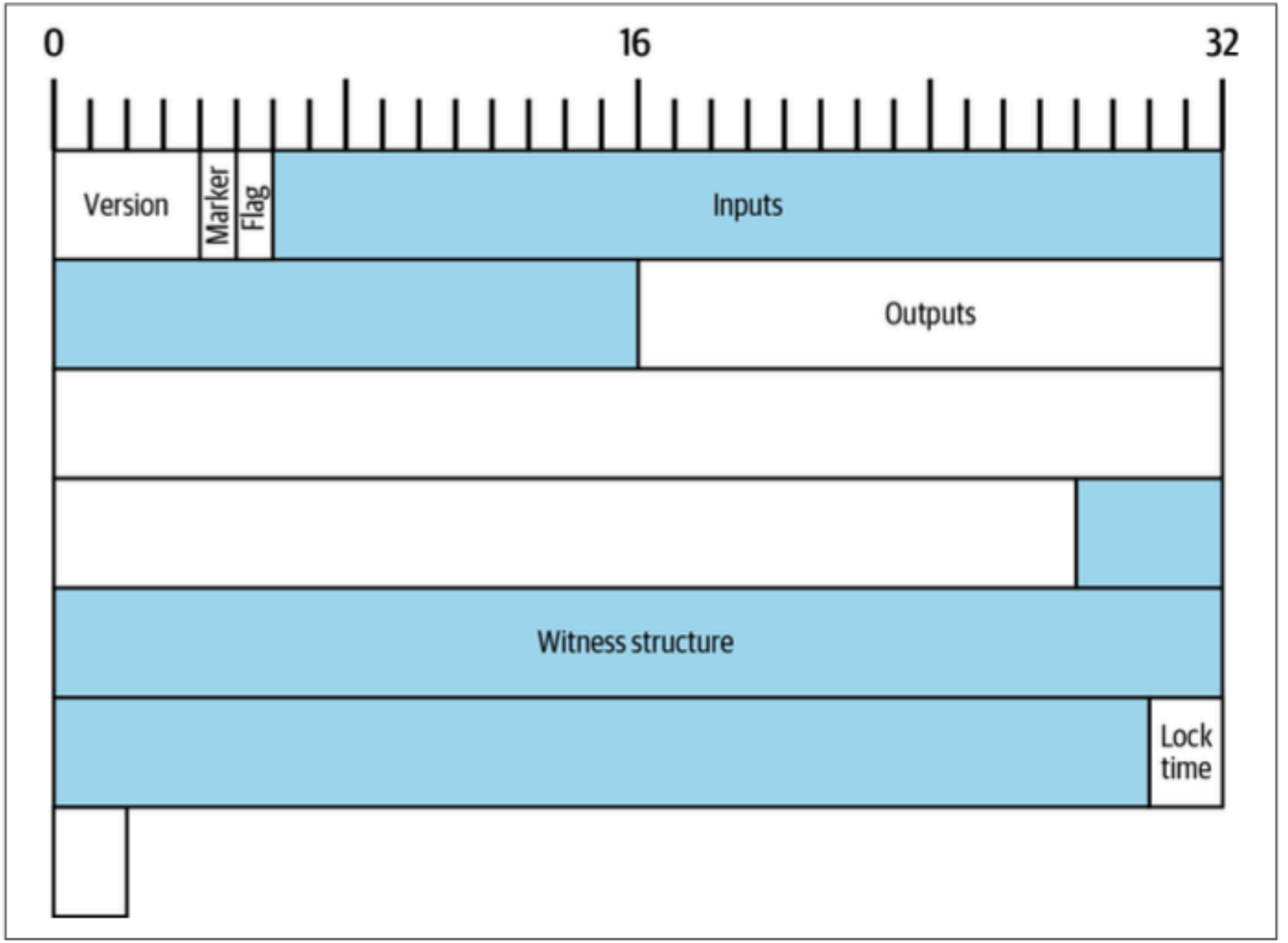
- Version (4 bytes, little-endian)
- Marker (1 byte), is 00 if SegWit transaction
- Flag (1 byte), is 01 for current version of transactions
- Inputs (variable), see breakdown here (without ScriptSig if SegWit transaction)
- Outputs (variable), see breakdown here
- Witness (variable, repeats for each input): Contains signature information
- Locktime (4 bytes, little-endian integer):
  If 499,999,999 or below: Indicates block height after which transaction can be mined
  If 500,000,000 or above: Indicates Linux time after which transaction can be mined

# THE STRUCTURE OF A BITCOIN TRANSACTION

| Field | Example | Size | Format | Description |
|---|---|---|---|---|
| Version ↘ | `02000000` | 4 bytes | Little-Endian | The version number for the transaction. Used to enable new features. |
| Marker ↘ | `00` | 1 byte | | Used to indicate a segwit transaction. Must be `00`. |
| Flag ↘ | `01` | 1 byte | | Used to indicate a segwit transaction. Must be `01` or greater. |
| Input Count ↘ | `01` | *variable* | Compact Size | Indicates the number of inputs. |
| Input(s) ↘ | *This structure repeats for every input.* See below for details. | | | |
| Output Count ↘ | `02` | *variable* | Compact Size | Indicates the number of outputs. |
| Output(s) ↘ | *This structure repeats for every output.* See below for details. | | | |
| Witness ↘ | *This structure repeats for every input.* | | | |
| Locktime ↘ | `00000000` | 4 bytes | Little-Endian | Set a time or height after which the transaction can be mined. |

**Note:** Rows in highlighted in blue are fields used in segwit transactions.

# THE STRUCTURE OF A BITCOIN TRANSACTION



# THE VERSION FIELD

## Determines whether certain new transaction features are enabled.

- Size: 4 bytes
- Type: Integer
- Format: Little-endian

- Version 1 (0x**01000000** in little-endian hex representation):
  Basic transactions
- Version 2 (0x**02000000** in little-endian hex representation) or higher:
  Enables new meaning of Sequence field (encoding relative lock-time of a transaction), see BIP 68 and BIP 112 (BIP = Bitcoin Improvement Proposal)

# THE MARKER FIELD

- Size: 1 bytes

Set to 0x**00** if segregated witness (SegWit) transaction.
Otherwise, byte after version field would indicate the count of inputs.

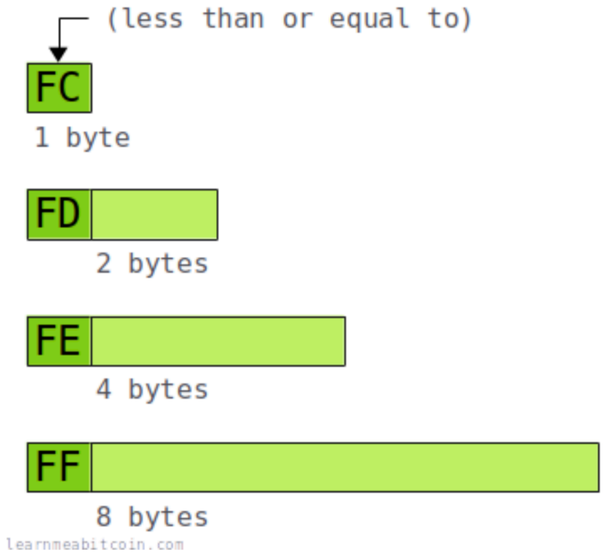
# THE INPUT COUNT FIELD

- Size: Variable (1 or 3 bytes)
- Type: Integer
- Format: Compact Size

Indicates the upcoming number of inputs in the transaction.

# COMPACT SIZE

A compact size field is a variable-length byte structure.




- Leading byte indicates
  - byte length of structure (if encoding integer larger than 252) or
  - integer itself (if smaller or equal than 252)
- Following 2, 4 or 6 bytes encode (in little-endian) larger integer (if larger than 252).

| Leading Byte | Number | Range | Field Size | Example |
|---|---|---|---|---|
| `FC` (and below) | Current byte | 0 - 252 | 1 byte | `64` (100) |
| `FD` | Next 2 bytes | 253 - 65535 | 3 bytes | `FDE803` (1,000) |
| `FE` | Next 4 bytes | 65536 - 4294967295 | 5 bytes | `FEA0860100` (100,000) |
| `FF` | Next 8 bytes | 4294967296 - 18446744073709551615 | 9 bytes | `FF00E40B5402000000` (10,000,000,000) |

# THE INPUTS FIELDS

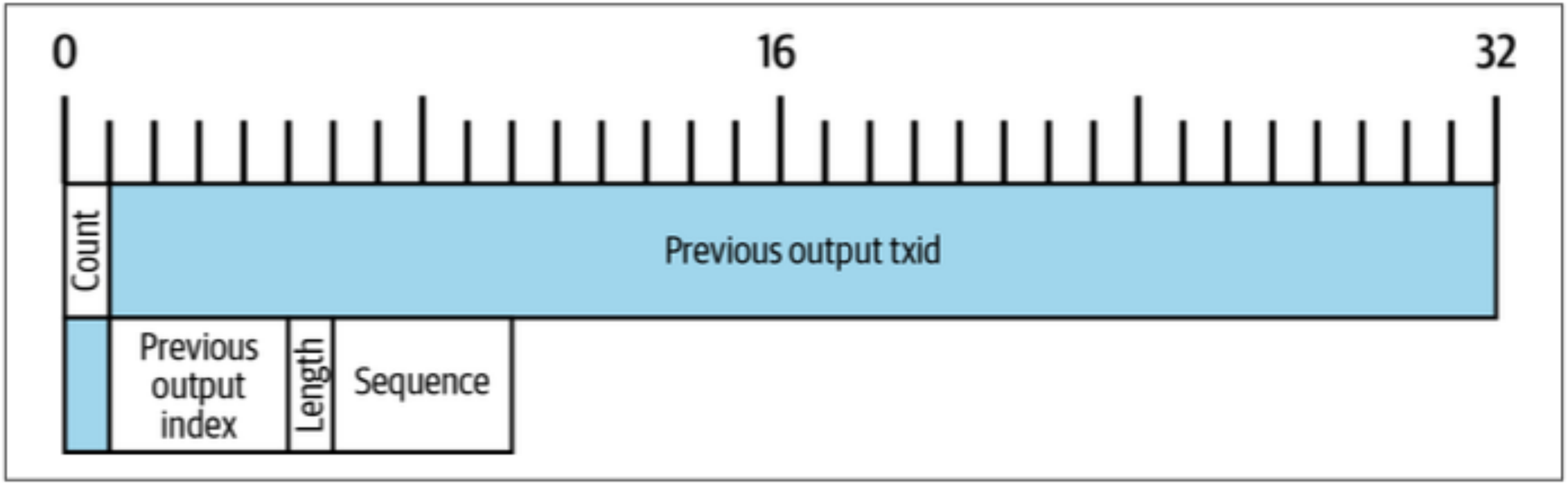
The following field blocks are present for each input of the transaction:

| Field | Example | Size | Format | Description |
|---|---|---|---|---|
| TXID | [TXID] | 32 bytes | Natural Byte Order | The TXID of the transaction containing the output you want to spend. |
| VOUT | 01000000 | 4 bytes | Little-Endian | The index number of the output you want to spend. |
| ScriptSig Size | 6b | *variable* | Compact Size | The size in bytes of the upcoming ScriptSig. |
| ScriptSig | [script] | *variable* | Script | The unlocking code for the output you want to spend. |
| Sequence | fdffffff | 4 bytes | Little-Endian | Set whether the transaction can be replaced or when it can be mined. |

- ScriptSig is empty for SegWit inputs (as corresponding information is relegated to Witness Field)
- Thus, ScriptSig Size = 0x**00** for SegWit inputs

# THE INPUTS FIELDS

Structure of an input (SegWit, thus, with empty ScriptSig)




Previous output index = VOUT
Length = ScriptSig Size

# THE INPUTS FIELDS: SEQUENCE

## 4 bytes, little-endian serialization.

Determines whether

- opt-in replace by fee (BIP 125) is signaled and/or whether
- absolute and/or relative locktime mining restrictions are imposed (BIP 68).

**Semantics:**

- <=0xfffffffe — **Locktime.** (Note: In serialization of Tx, would appear as e.g., “feffffff” (little-endian)
  This setting enables the transaction's locktime field to be used.
- <=0xfffffffd — **Replace By Fee (RBF)**.
  This setting enables the RBF feature, which allows you to replace a transaction with a higher-fee one if it's still in the mempool.
- <=0xefffffff — **Relative Locktime**.
  This setting allows you to set a locktime on the transaction relative to when the output being spent was mined.
  - 0x00000000 to 0x0000FFFF — **Blocks.** Set the relative locktime as a number of blocks.
  - 0x00400000 to 0x0040FFFF — **Time.** Set the relative locktime as a number of seconds.

# THE OUTPUT COUNT FIELD

- Size: Variable (1, 3 or 5 bytes)
- Type: Integer
- Format: Compact Size

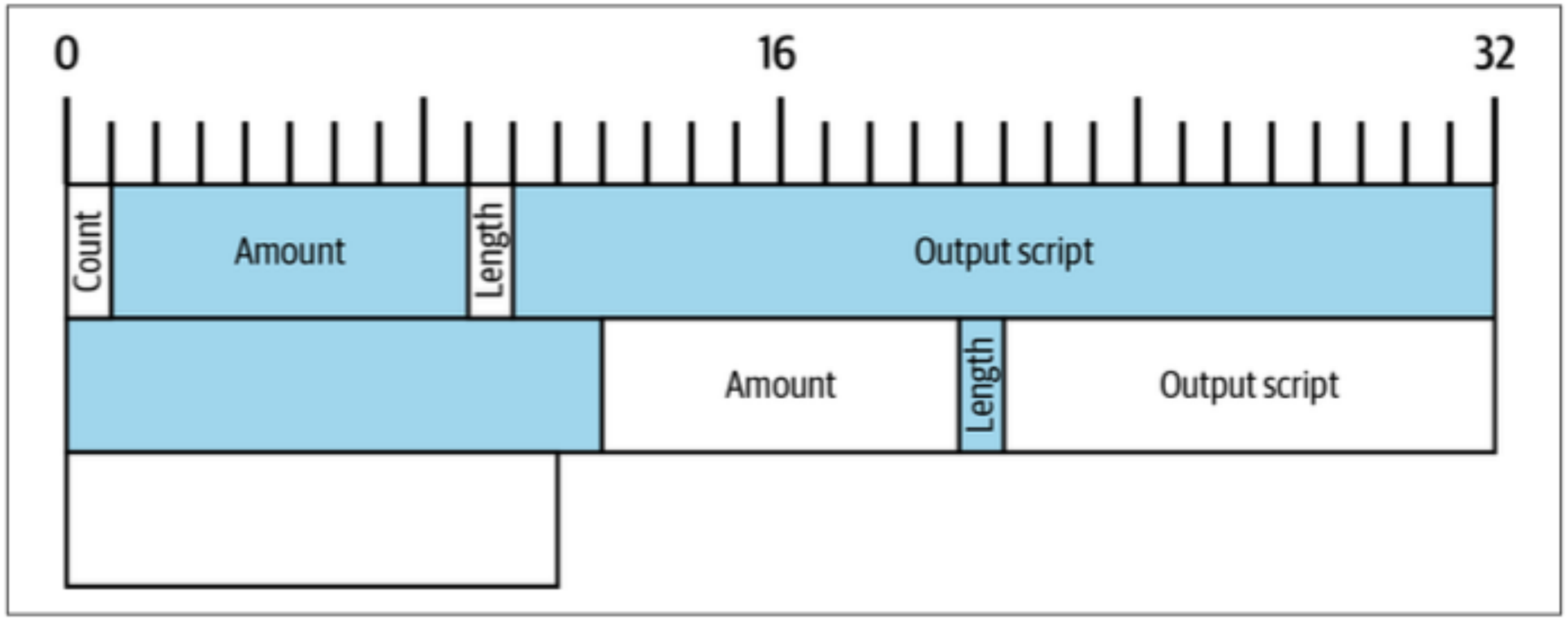
Indicates the upcoming number of outputs of the transaction.

# THE OUTPUTS FIELD

| Field | Example | Size | Format | Description |
|---|---|---|---|---|
| Amount | `e99e060000000000` | 8 bytes | Little-Endian | The value of the output in satoshis. |
| ScriptPubKey Size | `19` | *variable* | Compact Size | The size in bytes of the upcoming ScriptPubKey. |
| ScriptPubKey | `[script]` | *variable* | Script | The locking code for this output. |

- **Amount**: Indicates the amount of bitcoin spent towards the output address (1 BTC = 100,000,000 sathoshis)
- **ScriptPubKey/ Output Script**: Typically contains a particular address (single signature address such as P2PKH). Exact form to be discussed later when discussing Bitcoin Script

# THE OUTPUTS FIELD




Example with two outputs

# THE WITNESS FIELD

**Witness Field**

| Field | Example | Size | Format | Description |
|---|---|---|---|---|
| Stack Items | `02` | *variable* | Compact Size | The number of items to be pushed on to the stack as part of the unlocking code. |

| Field | Example | Size | Format | Description |
|---|---|---|---|---|
| Size | `47` | *variable* | Compact Size | The size of the upcoming stack item. |
| Item | `[public key] or [signature]` | *variable* | Bytes | The data to be pushed on to the stack. |

*This structure repeats for every stack item.*

Contains information used to unlock SegWit inputs (P2WPKH, P2WSH and P2TR), for example (e.g., signature information that is in the ScriptSig for pre-SegWit inputs)
We will understand the structure better after discussion Bitcoin Script.

# THE LOCKTIME FIELD

- Size: 4 bytes
- Type: Integer
- Format: Little-endian

Can set a specific block height or absolute lock time (in Unix time) of indicating the earliest block / time this transaction can be mined.

- If <=499999999: Transaction cannot be mined until after a specific **block height.**
- If >=500000000: Transaction cannot be mined until after this (**absolute) lock time.**

# THE STRUCTURE OF A BITCOIN TRANSACTION

```
0100000001 10ddd830599b17cc690535f7df28a84466eaca3c22f0d55b79023b6570f4fbc5 010000008b4830
45022100e6186d6f344ce4df46b2e15d87093d34edbf5b50462b6b45f9bd499a6a62fbc4022055f56a1c4a24
ea6be61564593c4196b47478a25cf596c1baf59f5a9a229b637c014104a41e997b6656bc4f5dd1f9b9df3b48
84cbec254d3b71d928587695b0df0a80417432f4ca6276bc620b1f04308e82e70015a40f597d8260912f801e
4b62ab089effffffff 02 00e9c8290100000001976a9146f34d3811aded1df870359f311c2a11a015e945388ac
00e40b5402000000 1976a91470d6734de69c1ac8913892f2df9be0e738d26c2d88ac 00000000
```

Example of Legacy transaction with 1 input (P2PKH) and 2 outputs (P2PKH) (see here for further breakdown)

```
0100000000010 1 3c735f81c1a0115af2e735554fb271ace18c32a3faf443f9db40cb9a11ca631 10000000000000f
ffffffff 02 b113030000000000160014689a681c462536ad7d735b497511e527e9f59245cf 1200000000000016
00148859f1e9ef3ba438e2ec317f8524ed41f8f06c6a 02473044022042472d4ad659960d4f1b541fd853f7da
62e8cf505c2f16585dc7c8cf643fe9a02207fbc63b9cf317fc41402b2e7f6fdc1b01f1b43c5456cf9b547fe96
45a16dcb150121032533cb19cf37842556dd2168b1c7b6f3a70cff25a6ff4d4b76f2889d2c88a3f2 00000000
```

Example of Segwit transaction with 1 input (P2WPKH) and 2 outputs (P2WPKH) (see here for further breakdown)

# TRANSACTION FEES

- Calculated by difference between input amounts and output amounts

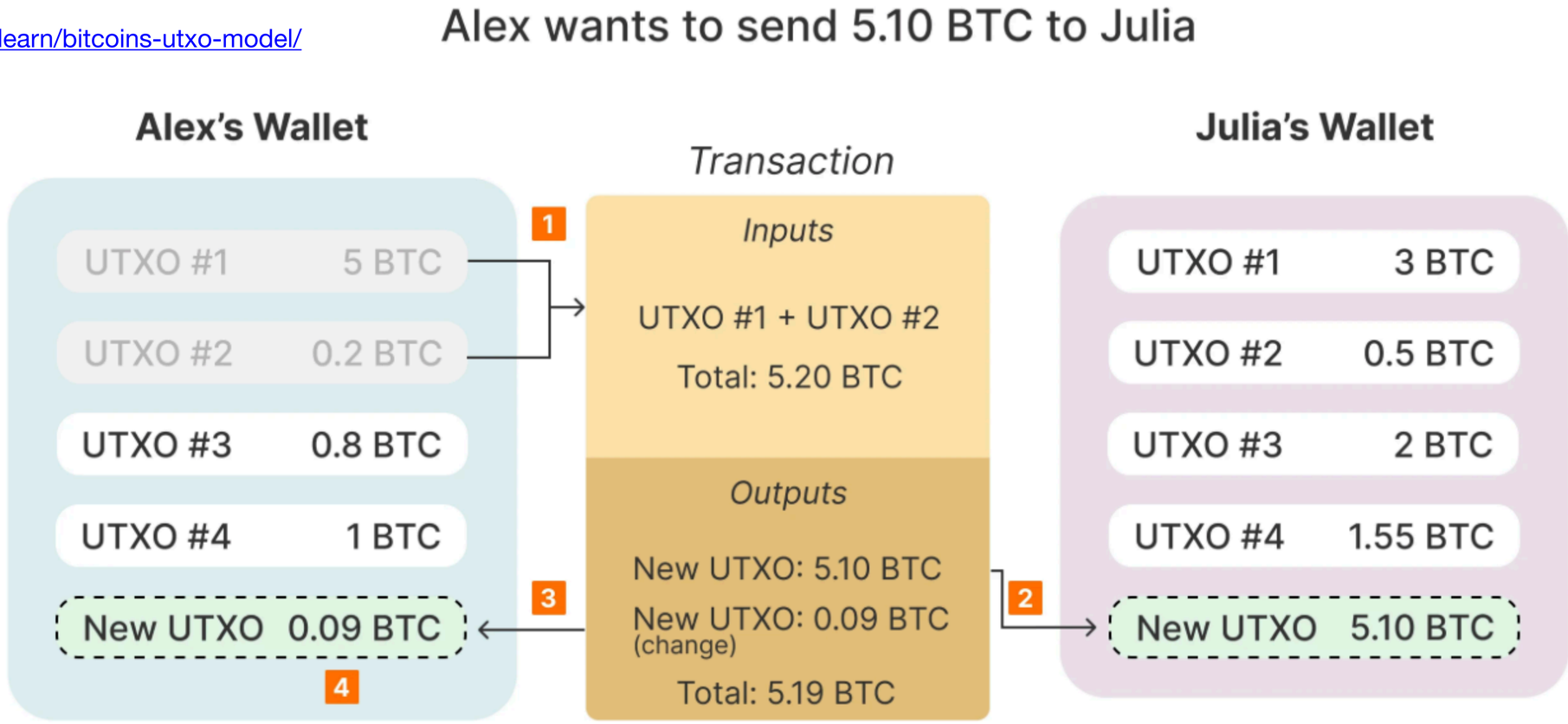
# UNSPENT TRANSACTION OUTPUTS

- A **UTXO** is an “unspent transaction output”:
  An amount of bitcoin units that are an output of some transaction, but have not yet been spent. (no input of any transaction)
- The **UTXO set** is the collection of all UTXOs in the network:
  - Convenient to consider in Bitcoin node software to “audit” the amount of coins in circulation
  - Can be used to easily detect a possible “double-spent”

- **Atomicity** of UTXOs: The “bitcoin balance” that you can have access to spend with your private key is better be understood as a collection of several UTXOs.
- UTXOs can be **traced back separately** -> Privacy implications

# EXAMPLE: BALANCES AND UTXOS IN A TRANSACTION

Source: https://river.com/learn/bitcoins-utxo-model/



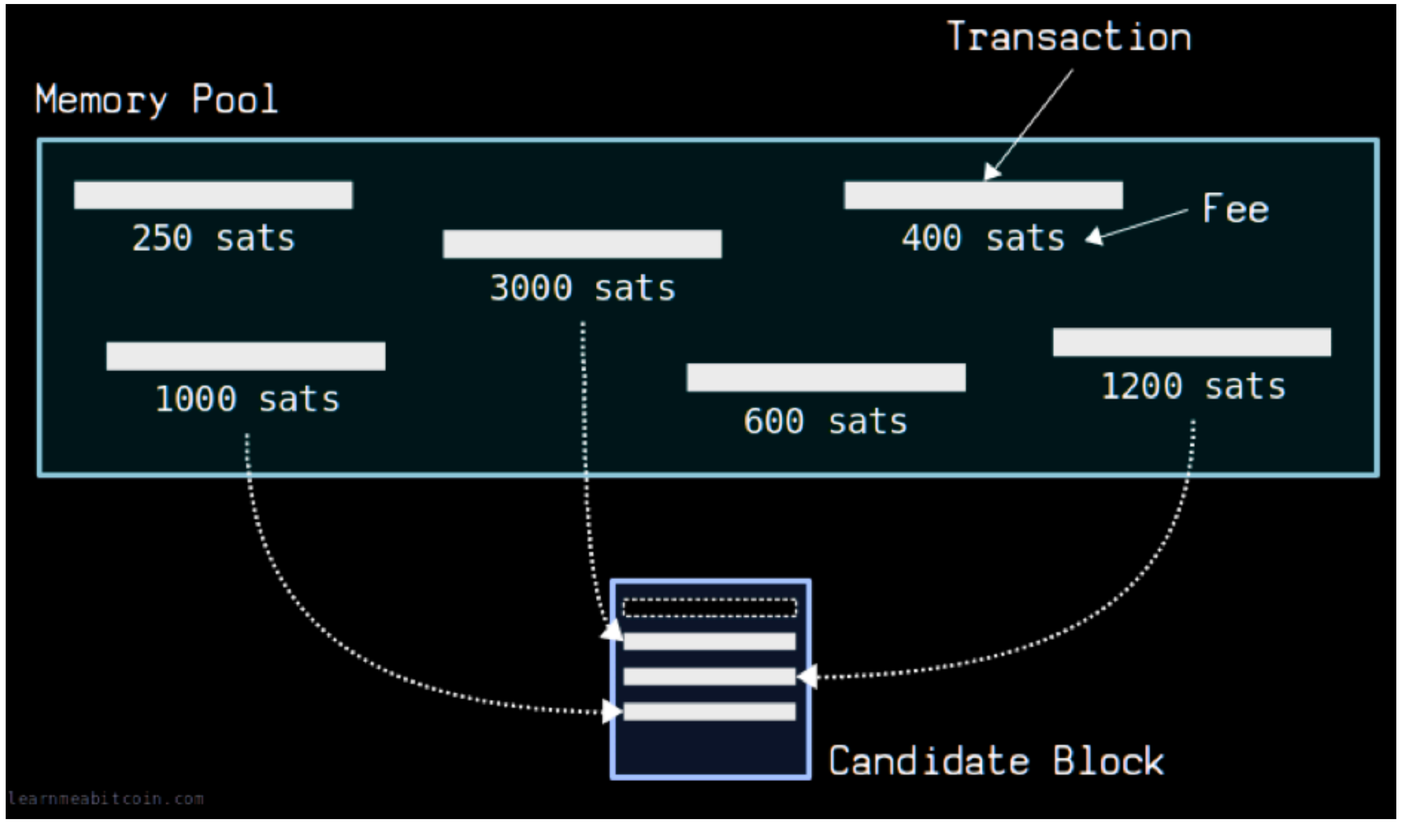

1. Alex's wallet selects the best UTXOs to get to the transaction amount or greater. Since UTXOs are indivisible, the wallet will select two UTXOs.
2. The wallet creates a new UTXO for Julia in the amount of the transaction.
3. The wallet creates a new UTXO for Alex, which is the "change."
4. The transaction fee is not paid to the miner as an output of the transaction. It is inferred by the difference between the value of the inputs and the value of the outputs.

RIVER

# Fees

# TRANSACTION FEES IN BITCOIN

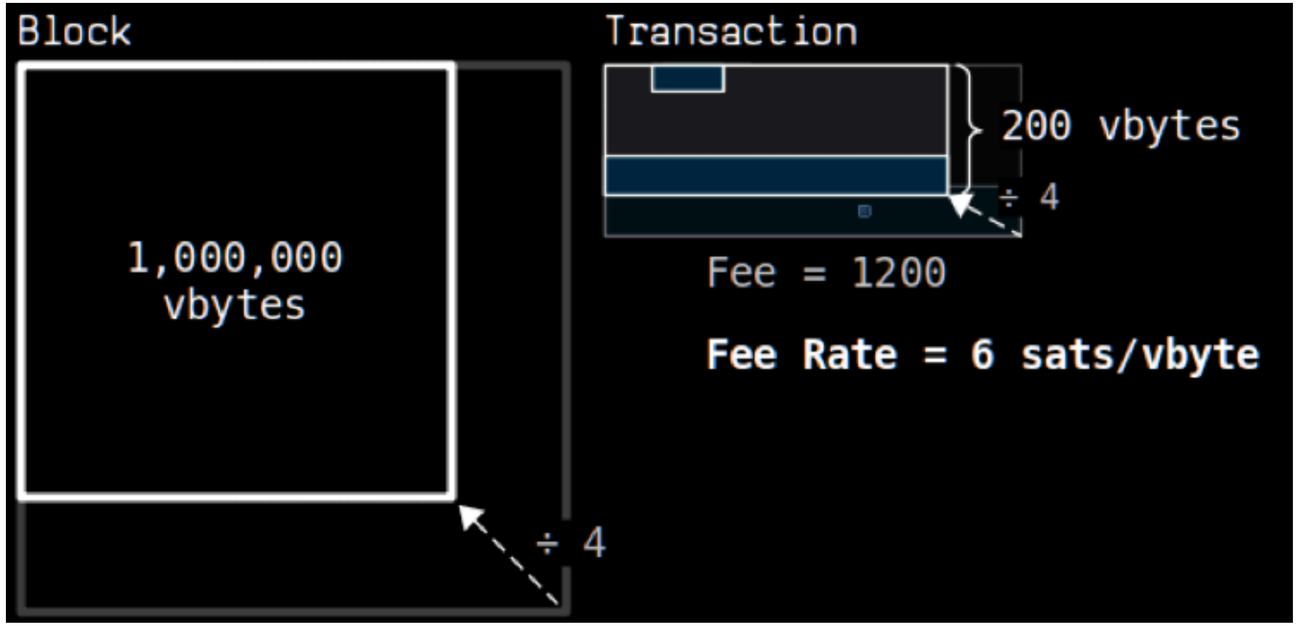
- Transactions in mempool are all eligible to be included in the next block by any miner.



# FEES AND FEE RATES (CALCULATION BEFORE 2017 SEGWIT SOFT FORK)

- Sender of transaction can choose a “fee rate” that is measured in “sats/byte” (1 BTC = 100,000,000 satoshis (sats))
- Transaction fee is computed such that

$$\text{Fee} = (\text{Bytes of transaction}) \cdot (\text{fee rate in sats/byte})$$

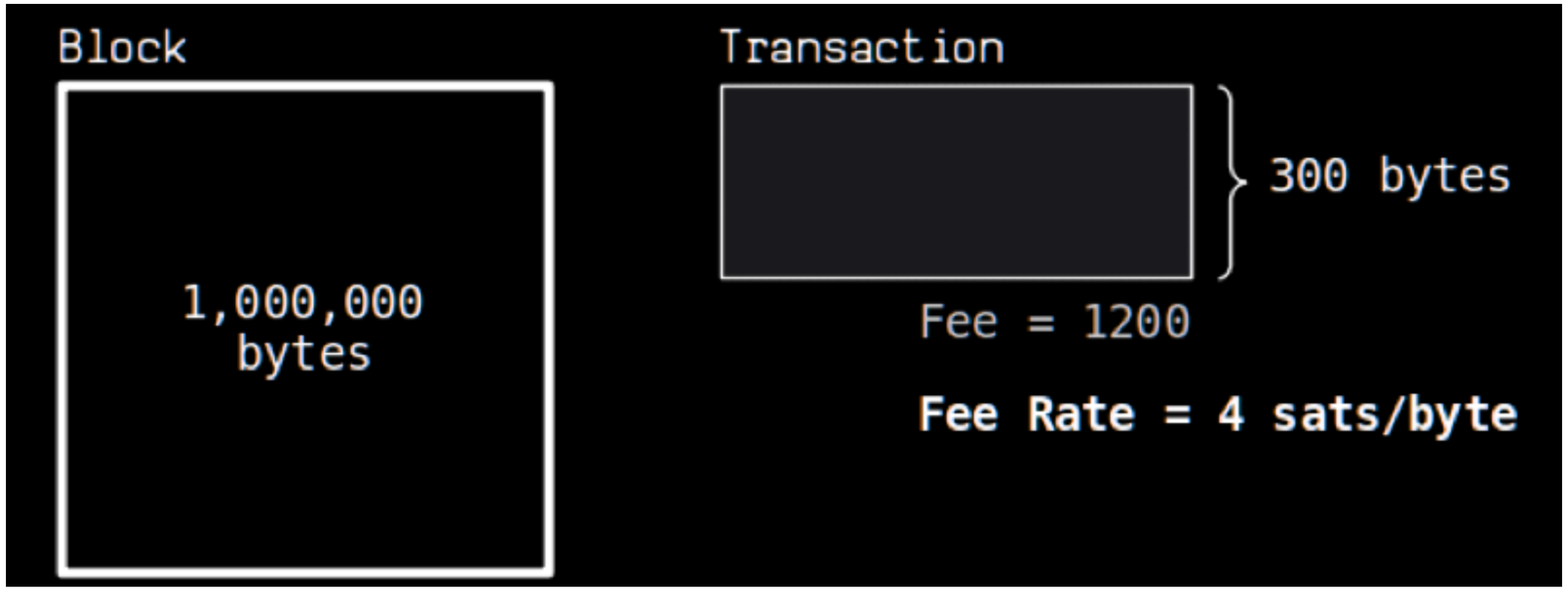



- Size of a block is upper bounded by 1 MB (or 1,000,000 bytes)

# FEES AND FEE RATES (CALCULATION AFTER 2017 SEGWIT SOFT FORK)

- Sender of transaction can choose a “fee rate” that is measured in “sats/**vbyte**” where “vbyte” stands for a “virtual byte” unit.
- Transaction fee is computed such that

$$\text{Fee} = (\text{VBytes of transaction}) \cdot (\text{fee rate in sats/vbyte})$$

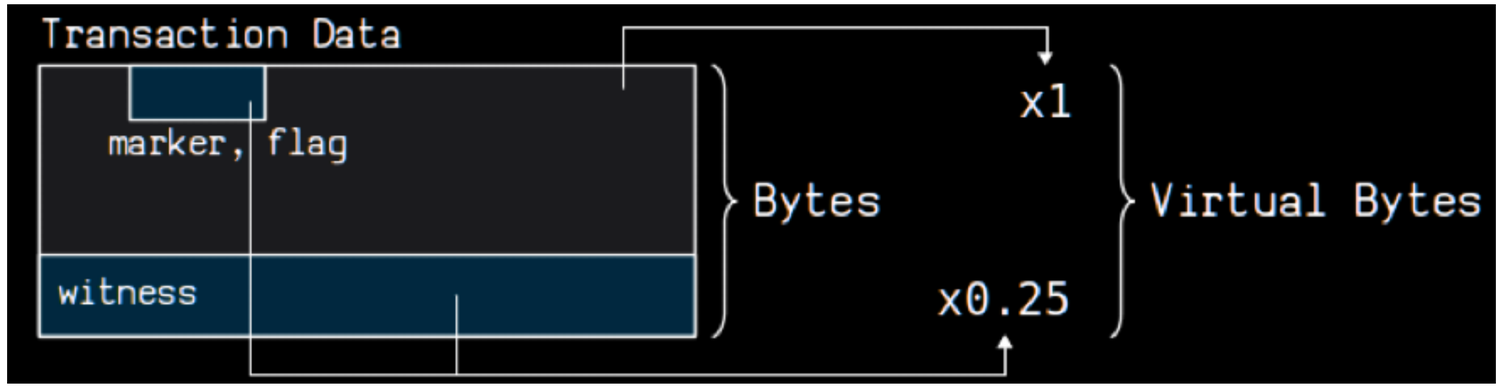



- Size of a block is upper bounded by 1,000,000 “vbytes” (can be more than 1 MB)

# WHAT IS A VIRTUAL BYTE (VBYTE)?

- A vbyte unit corresponds to a byte in most transaction fields, but to one quarter of a byte for the “Marker”, “Flag” and “Witness” fields.

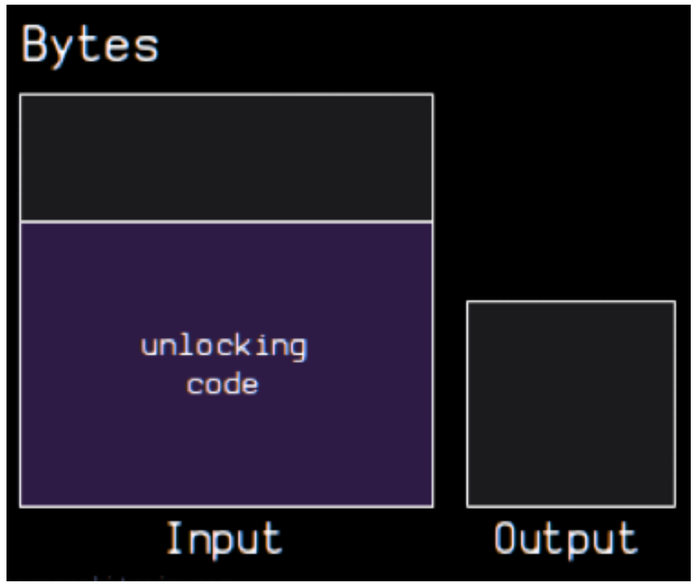
| Field | Multiplier |
|---|---|
| version | x1 |
| marker | x0.25 |
| flag | x0.25 |
| input | x1 |
| output | x1 |
| witness | x0.25 |
| locktime | x1 |



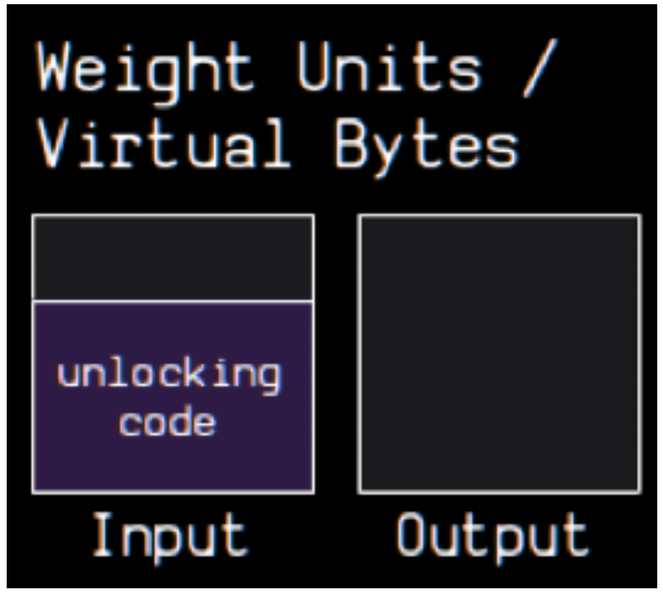

- Size of a block is upper bounded by 1,000,000 “vbytes” (can be more than 1 MB)

# PURPOSE OF SATS/VBYTE FEE RATES IN SEGWIT SOFT FORK

- Implicit way of slightly increasing the block size limit of 1 MB (higher amount of transactions / block supported)
- Fairer distribution of fees between spender & recipient




Byte sizes before SegWit




VByte sizes after SegWit