Mechanics of Bitcoin (1)

Blocks & Transactions

Prof. James Won-Ki Hong

Distributed Processing and Network Management (DPNM) Lab.
Dept. of Computer Science and Engineering
POSTECH
Pohang, Korea

http://dpnm.postech.ac.kr
jwkhong@postech.ac.kr

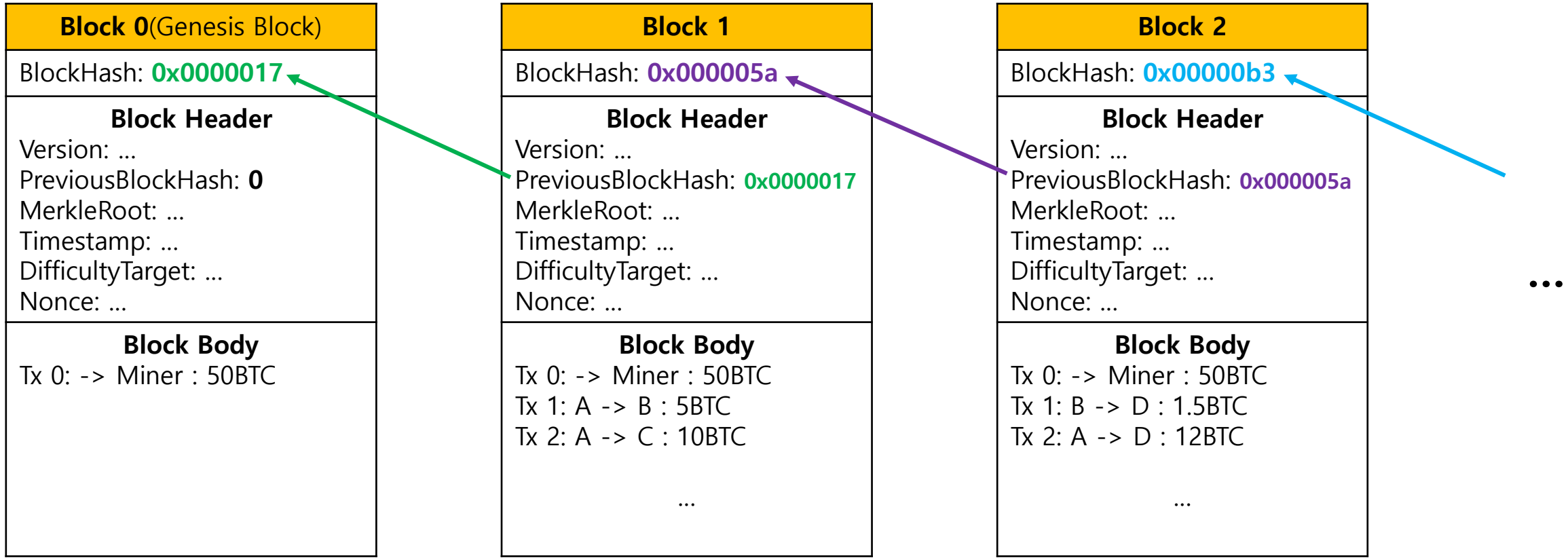
Table of Contents

Origin of Bitcoin

- In 2008, an anonymous developer or development group named "Satoshi Nakamoto" proposed the first cryptocurrency-based digital payment system called Bitcoin
- Used Distributed Ledger Technology (= Blockchain)
  - No centralized management
  - Open transaction history
  - Immutability of transaction data
  - Strong security (counterfeiting is impossible)
- Total coins limited to 21 Million BTC (Bitcoins)
  - Used for electronic payment
  - Transaction fees & incentives from Mining
  - Still the most valuable cryptocurrency….

Blockchain in Bitcoin

Blockchain in Bitcoin



Transactions → Blocks → Blockchain

Elements in Block (1/7)

Structure of a Block

- Block = minimum unit to be saved in the blockchain
- A container data structure

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields from the block header
1-9 bytes (VarInt)	Transaction Counter	How many transactions follow
variable	Transactions	The transactions recorded in this block

The structure of a block

Elements in Block (2/7)

- **Block Header**
  - Consist of several block metadata (Three sets of block metadata)

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the Merkle-Tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block(seconds from Unix Epoch)
4 bytes	Difficulty Target	The proof-of-work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the proof-of-work algorithm

The structure of a block header

Elements in Block (3/7)

- **Block Identifiers**
  - **Block Hash** (Block Header Hash)
    - Cryptographic hash of a block (block header)
    - Block Hash = SHA256(SHA256(block header))
    - 32 byte hash

      e.g., 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
    - Identify a block uniquely and explicitly
    - Play a role of digital fingerprint
  - **Block Height**
    - Block's **position** in the blockchain
      - The first block is at block height 0
      - The height of current block = The height of previous block + 1
    - Not a unique identifier

Elements in Block (4/7)

- **Genesis Block**
  - The first block in the blockchian
  - The common ancestor of all the blocks in the blockchain
  - Statically encoded within the bitcoin client software
    - Every node always starts with a blockchain of at least one block
  - Cannot be altered

⇒ You can look into the genesis block using several websites of block explorer

https://blockchain.info/block/

000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

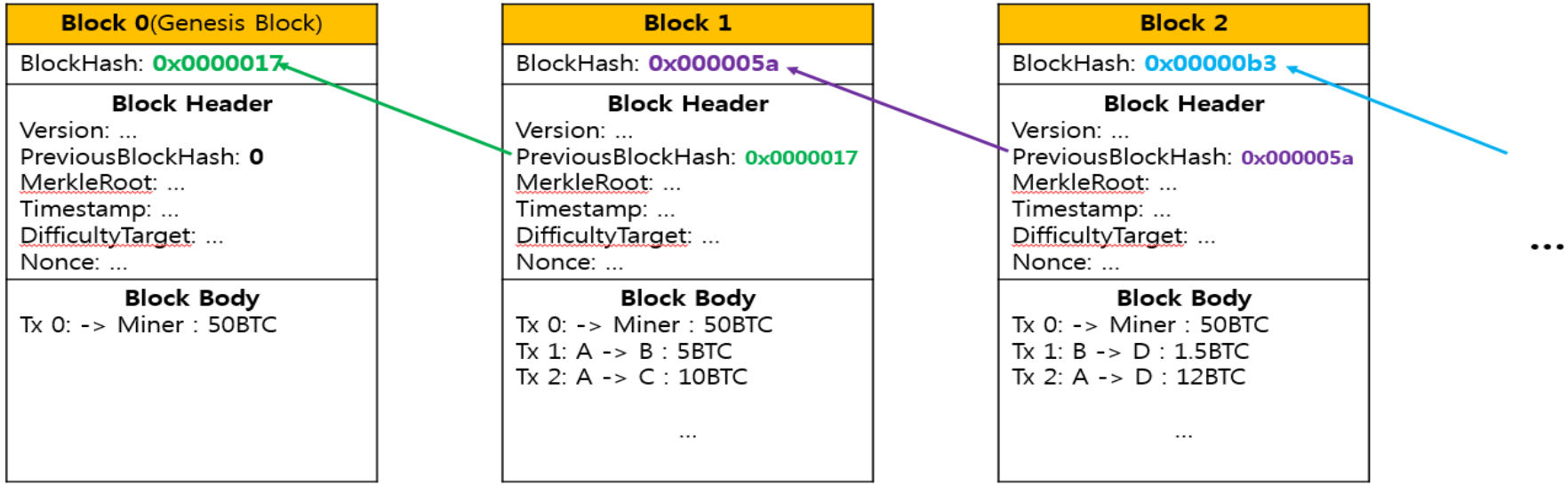
Elements in Block (5/7)

- **Linking Blocks in the Blockchain**
  - Bitcoin nodes maintain a local copy of the blockchain starting from genesis block
    - constantly updated as new blocks are created
  - Steps

    1) A node receives an incoming block from the network

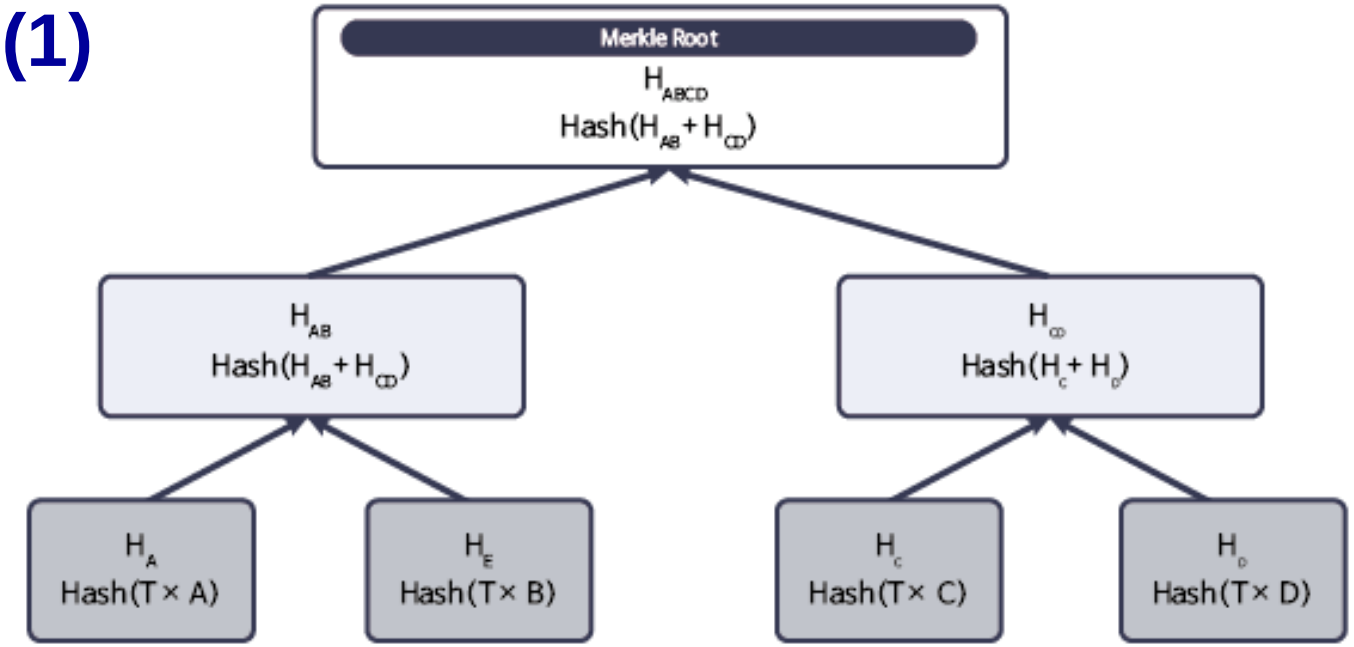
    2) It validates this block

    3) If the validation is passed, it links this block to the existing blockchain



Elements in Block (6/7)

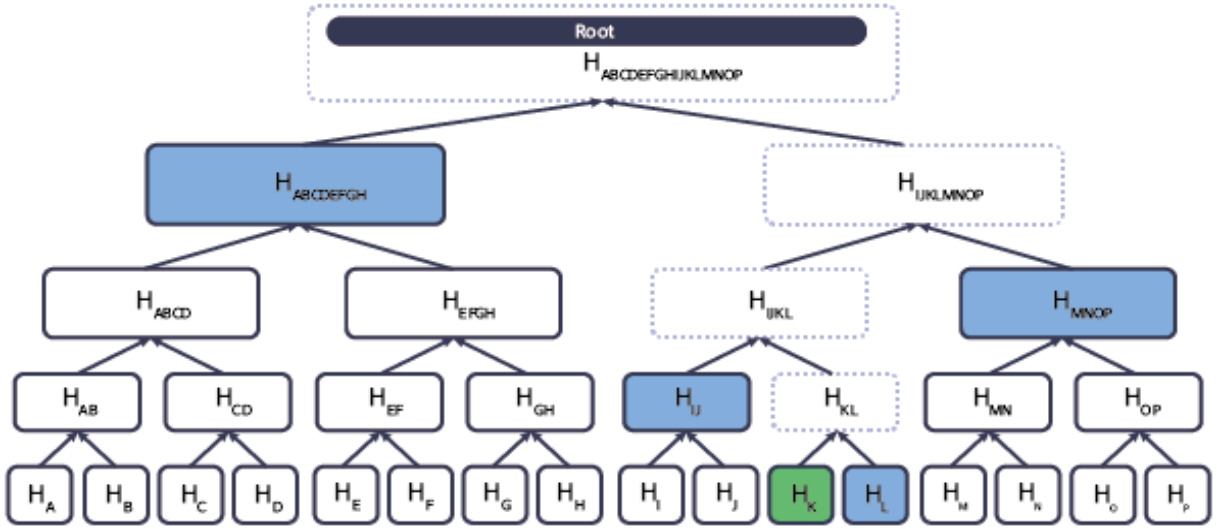
- **Merkle Tree (1)**



- Binary Hash Tree and Balanced Tree
  - Recursively hashing pairs of nodes until there is only one hash (**Merkle Root)**
- Used to summarize all the transactions in a block
- Uses the cryptographic hash algorithm and bottom-up method
  - H_A = SHA256(SHA256(Transaction A))
  - H_{AB} =SHA256(SHA256(H_A + H_B))

Elements in Block (7/7)

- **Merkle Tree (2)**
  - Main features
    - Produce an overall digital fingerprint of the entire set of transactions
    - Provide a very efficient process to verify if a transaction is included in a block
  - To prove that a specific transaction is included in a block
    - Merkle path
      - Connect the specific transaction to the root of the tree
      - log_2(N) 32-byte hashes



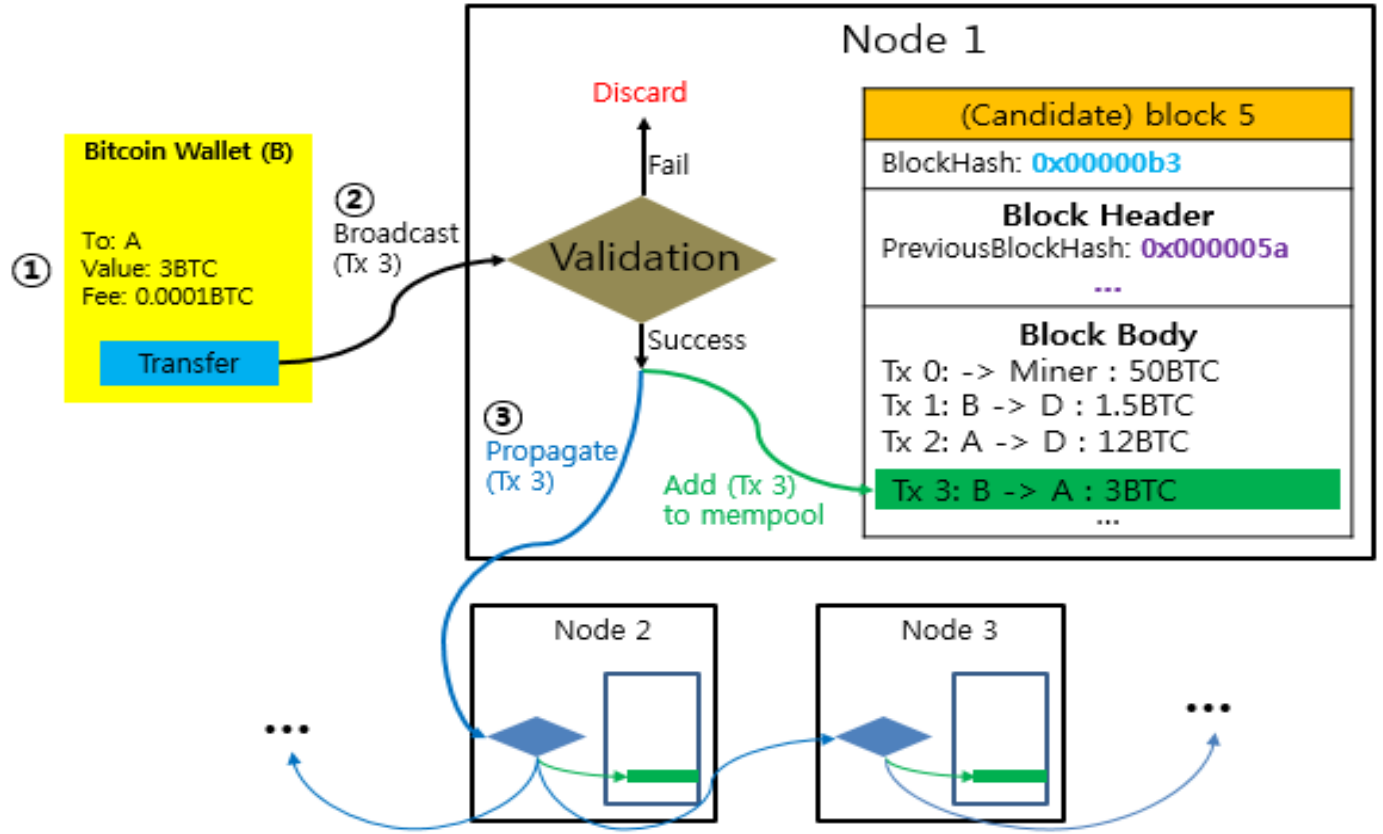
Number of Transactions	Approx. Size of Block	Path Size (Hashes)	Path Size (Bytes)
16 transactions	4 KB	4 hashes	128 bytes
512 transactions	128 KB	9 hashes	288 bytes
2048 transactions	512 KB	11 hashes	352 bytes
65,535 transactions	16 MB	16 hashes	512 bytes

Merkle Tree Efficiency

Elements in Transaction

- **Transaction in Bitcoin**
  - Data structure for the transfer of value between participants in Bitcoin
  - Each transaction is a public entry in the ledger of Bitcoin

- **Transaction Lifecycle**
  - ① Creating transactions
  - ② Broadcasting transactions to the bitcoin network
  - ③ Propagating transactions on the bitcoin network



Elements in Transaction

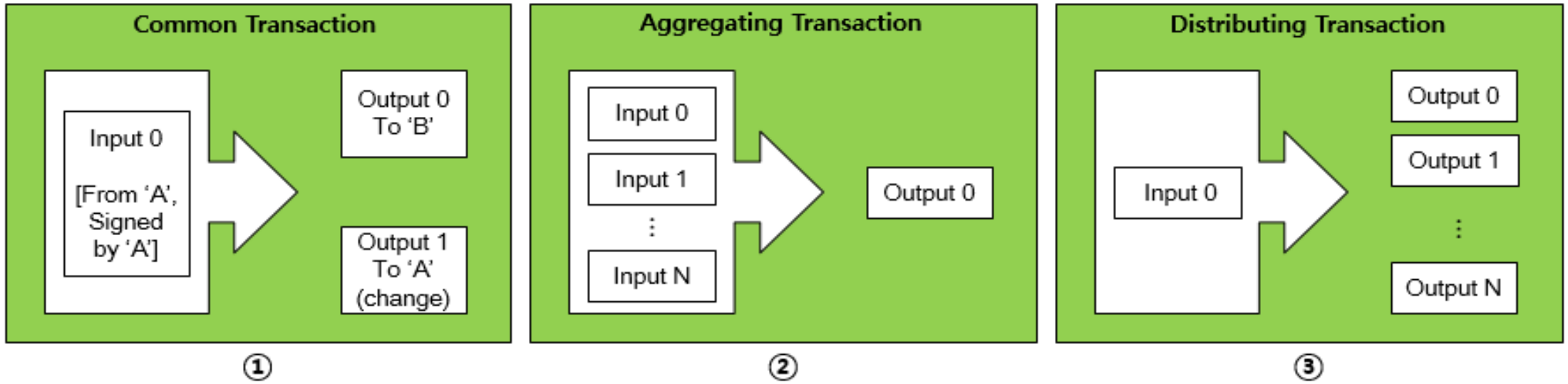
Transaction Structure

- A data structure that encodes a transfer of value from **an input** to **an input**
  - ➔ From a source of funds to a destination

Size	Field	Description
4 bytes	Version	Specifies which rules this transaction follows
1-9 bytes (VarInt)	Input Counter	How many inputs are included
Variable	Inputs	One or more Transactions Inputs
1-9 bytes (VarInt)	Output Counter	How many outputs are included
Variable	Outputs	One or more Transactions Outputs
4 bytes	Locktime	A unix timestamp or block number

The structure of a transaction

Types of Transaction



Elements in Transaction

- **Unspent Transaction Output (UTXO)**
  - Locked to a specific owner
  - Recorded on the blockchain
  - Recognized as currency units by the entire network
  - Indivisible chunks of bitcoin currency

  ➔ User's bitcoin = UTXO (scattered amongst hundreds of blocks)

- **Relationship between *UTXO* and *Transaction Inputs/Outputs***
  - Transaction inputs = The UTXO consumed by a transaction
    - Transactions consume UTXO, unlocking it with the signature of the current owner
  - Transaction outputs = The UTXO created by a transaction
    - Transactions create UTXO, locking it to the bitcoin address of the new owner

Elements in Transaction

- **Transaction Outputs**
  - Every bitcoin transaction creates outputs
  - Sending someone bitcoin

  ➔ Creating an UTXO registered to their address and available for them to spend
  - Transaction outputs consist of two parts:
    - An amount of bitcoin
    - A locking script that locks this amount by specifying the conditions that must be met to spend the output

Size	Field	Description
8 bytes	Amount	Bitcoin Value in Satoshi (10^{-8} bitcoin)
1-9 bytes (VarInt)	Locking-Script Size	Locking-Script length in bytes, to follow
variable	Locking Script	A script defining the conditions needed to spend the output

The structure of a transaction output

Elements in Transaction

Transaction Inputs

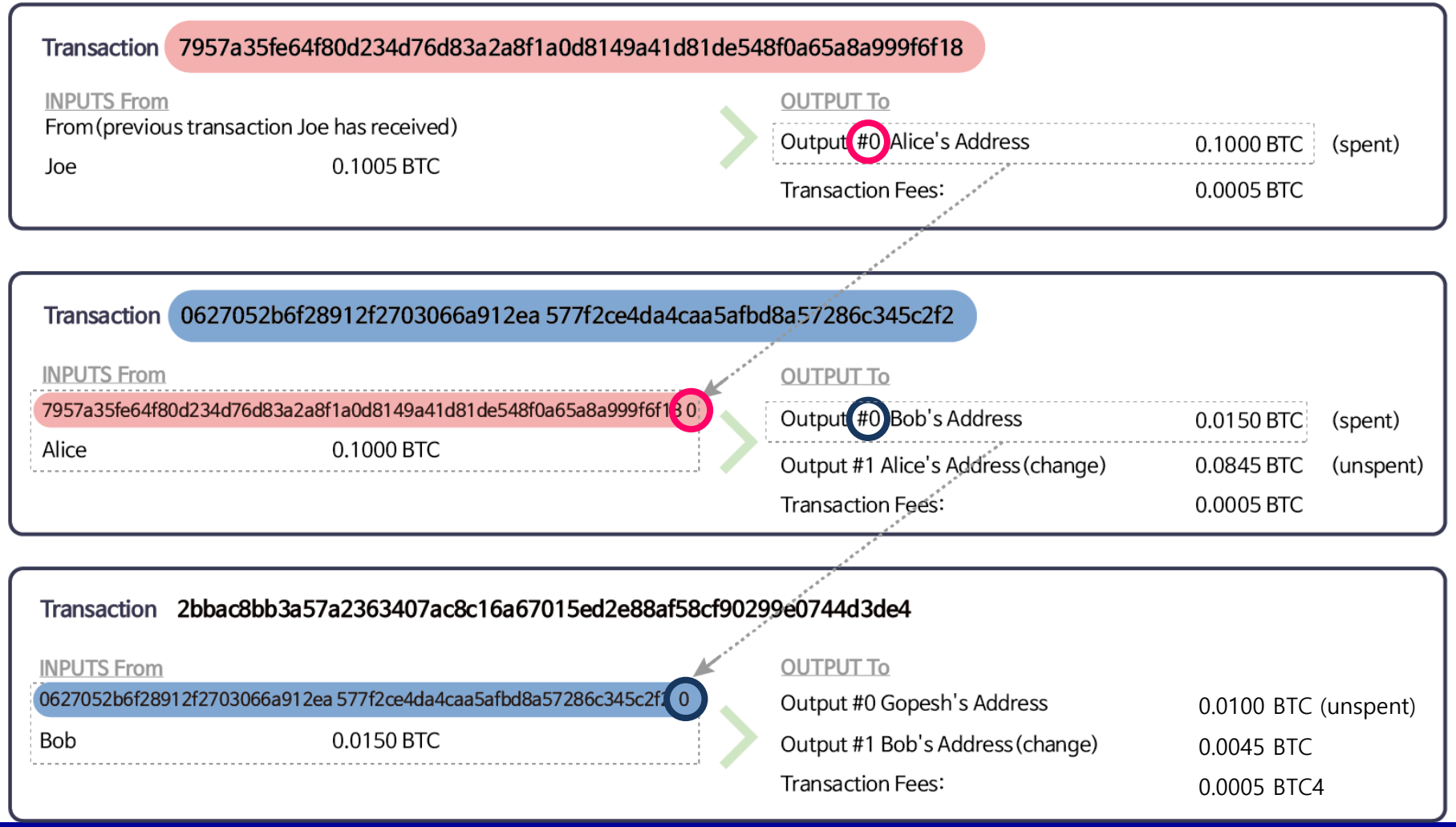
- Point to a specific UTXO
  - By reference to the **transaction hash** and **sequence number**
- Include **unlocking-scripts**
  - These satisfy the spending conditions set by the UTXO
  - Unlocking script = a signature

Size	Field	Description
32 bytes	Transaction hash	Pointer to the transaction containing the UTXO to be spent
4 bytes	Output Index	The index number of the UTXO to be spent, first one is 0
1-9 bytes (VarInt)	Unlocking-Script Size	Unlocking-Script length in bytes, to follow
Variable	Unlocking-Script	A script that fulfills the conditions of the UTXO locking-script
4 bytes	Sequence Number	Currently-disabled Tx-replacement feature, set to 0XFFFFFFFF

The structure of a transaction input

Elements in Transaction

- **A Chain of Transactions**
  - The output of one transaction is the input of the next transaction



Elements in Transaction

- **Fee**
  - Serve as an **incentive**
    - To mine a transaction into the next block
    - The miner who mines the block collects fees
  - Serve as a disincentive against spam
  - Fees = Sum(Inputs) – Sum(Outputs)
  - Calculated based on the size of the transaction in kilobytes
    - Not the value of the transaction in Bitcoin
  - Used as one of criteria of priority to be mined
  - Not mandatory

- **Coinbase**
  - First added transaction in a block = Generation transaction
    - Does not consume UTXO as inputs
    - Has one input, which creates Bitcoin from nothing (Coinbase)
    - Has one output, payable to the miner's own bitcoin address
  - **Reward for the mining effort**

Elements in Transaction

- **Bitcoin Script**
  - **Locking script**
    - An encumbrance placed on an output
    - Specify the conditions that must be met to spend the output in the future
    - Called a **scriptPubKey**
      - Because it usually contains a public key or bitcoin address
  - **Unlocking script**
    - Solve the conditions placed on an output by a locking script
      - To allow the output to be spent
    - Part of every transaction input
    - Mostly contain a digital signature
    - Called a **scriptSig**
      - Because it usually contained a digital signature

Elements in Transaction

Scripting Language

- Stack-based execution language
  - A stack allows two operations: **push** and **pop**.
- Script is a very simple, lightweight language
  - Designed to be limited in scope and executable on a range of hardware
- Execute the script by processing each item **from left to right**

Turing Incompleteness

- No loops or complex flow control capabilities
  - ⇒ Ensure that this script language is not Turing Complete
  - ⇒ Ensure that the language cannot be used to create **infinite loop** or **logic bomb**
- Prevent the transaction validation mechanism from being used as a vulnerability

Stateless Verification

- The bitcoin transaction script language is **stateless**
- A script
  - Be predictable
  - Execute the same way on any system

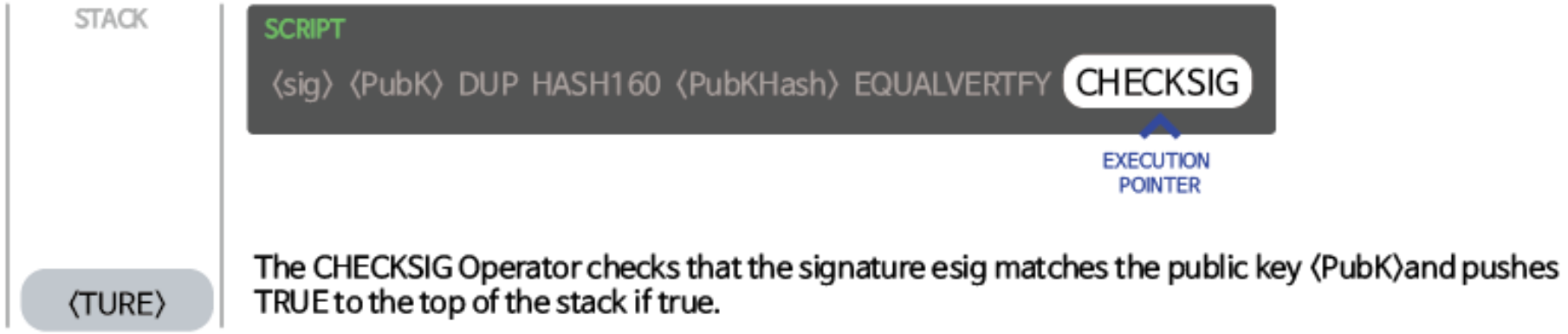
Elements in Transaction

Standard Transactions (1)

- Bitcoin developers introduced some restrictions in the types of scripts
  - Five standard types
- The five standard types of transaction scripts:
  1. **Pay-to-Public-Key-Hash (P2PKH)**
  2. **Pay-to-Public-Key**
  3. **Multi-Signature**
  4. **Pay-to-Script-Hash (P2SH)**
  5. **Data Output (OP_RETURN)**

Elements in Transaction

- **Standard Transactions (2)**
  - Evaluating a script for a Pay-to-Public-Key-Hash transaction



Summary

- **Block**
  - Block Structure
  - Block Identifier
  - Genesis Block and how to connect to blockcahin
  - Merkle Tree

- **Transaction**
  - Transaction Structure and Types
  - UTXO
  - Transaction Inputs/Outputs
  - Bitcoin Script
  - Standard Transactions

References

- Andreas M. Antonopoulos, **Mastering Bitcoin,** O'Reilly, 2014
- http://homoefficio.github.io/2016/01/23/BlockChain-%EA%B8%B0%EC%B4%88-%EA%B0%9C%EB%85%90/