Unit 2 - Cryptocurrency Fundamentals – I

Text Book:

Mastering Blockchain

Unlocking the Power of Cryptocurrencies and Smart Contracts

Authors

Lorne Lantz & Daniel Cawrey

Public and Private Keys in Cryptocurrency Systems

- When someone signs up for a Bitcoin wallet, it generates a **public** key and a **private** key, and a public **address**
  - Public key can be shared with anyone
  - However, private key is kept secret and not shared
  - Public key is used to compute public address, which uniquely identifies the wallet
  - Public address can be shared with everyone and is used to send and receive funds or any data of digital value
- Public private key pair can be used to either **digitally sign a transaction** or to **send secure (confidential) messages**
- For sending a secure message from A to B,
  - The message at A is encrypted with the public key of B
  - This encrypted message is then sent to B
  - B uses its private key to decrypt the message and read the information

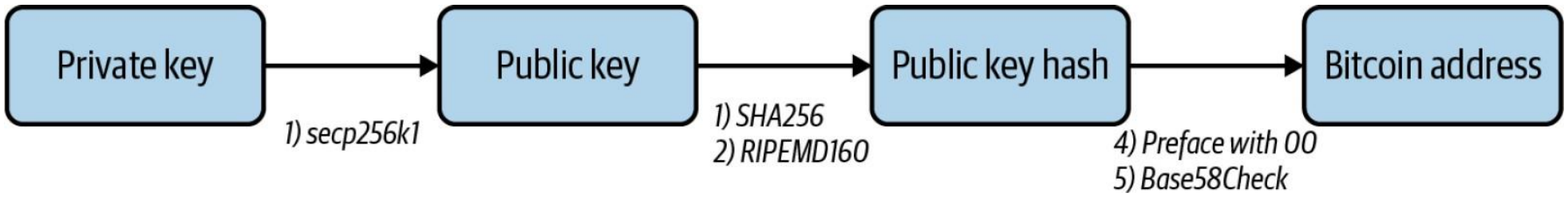
Public and Private Keys in Cryptocurrency Systems

Private key	Kyc9JCPPKNPrMUopkCc7ng9PU5Bp9SG
Public key	033b368bfccf5921f8a5a
Bitcoin address	19Paci

Example of how public key, private key, and address looks

Public and Private Keys in Cryptocurrency Systems

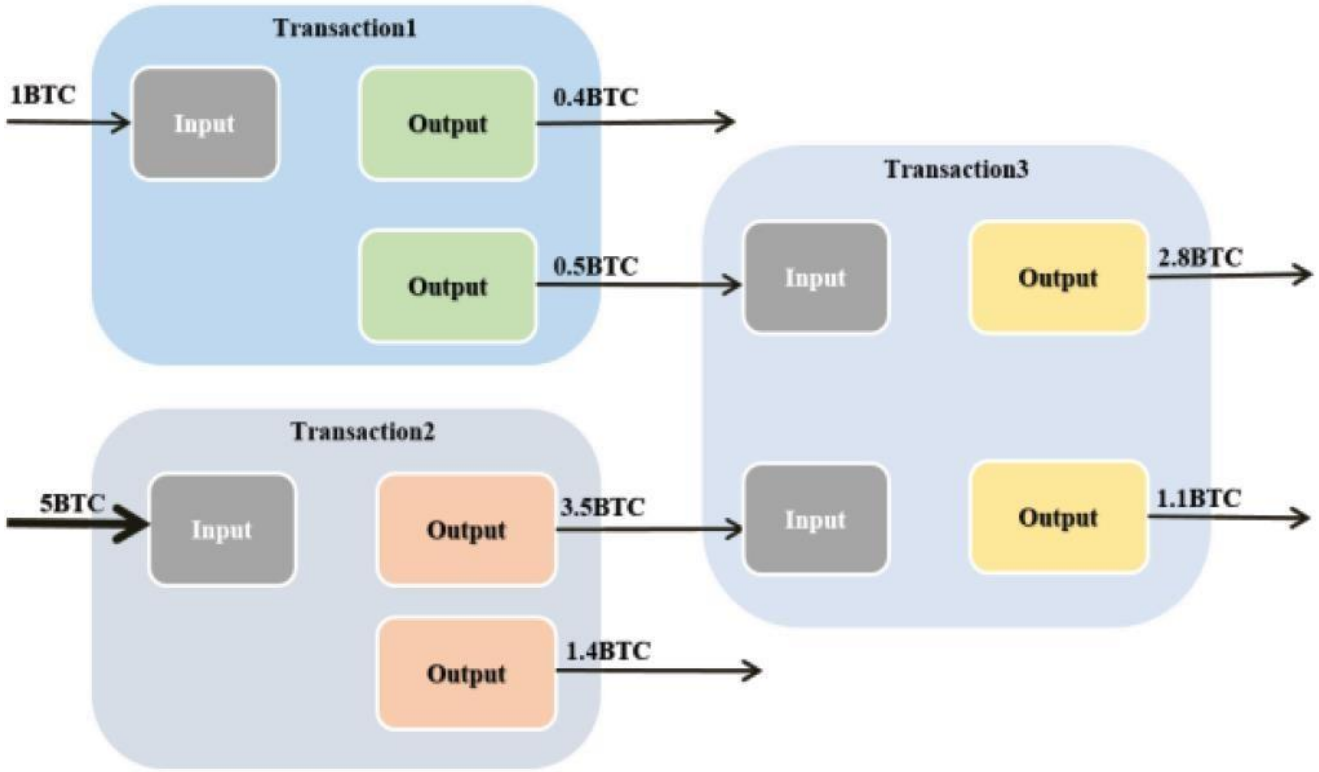
- A **private key** is usually 256-bit number and mostly shown in **hexadecimal** format
- Private key is chosen at random by using a function to generate a random number
- The corresponding **public key** can be generated by running the private key through an Elliptic Curve Digital Signature Algorithm (ECDSA) secp256k1 function
- The **double hash** of public key is then generated by running the public key through first SHA256 and then RIPEMD160 functions (output is 160 bits or 20 bytes)
- Prefix this hash with 00 and then run through Base58Check encoding
- The result is Base58Check Encoded Public Key Hash which is bitcoin address



The UTXO Model

The UTXO Model

- UTXO stands for Unspent Transaction Output
- It is unique type of **accounting** followed by Bitcoin transactions
- Fundamentally, a Bitcoin transaction is essentially a list of **inputs** and a list of **outputs**
- An **input** identifies Bitcoin address of the user who has sent the funds, and the value of that unspent transaction
- An **output** represents the Bitcoin address receiving the funds and the amount that address receives
- The difference between the input and the output is the **transaction fee**, which will be earned by the bitcoin miner



Simple Figure Explaining UTXO Model

Source of Figure:
Xue, Z., Wang, M., Zhang, Q., Zhang, Y. and Liu, P., 2021. A regulatable blockchain transaction model with privacy protection. International Journal of Computational Intelligence Systems, 14(1), pp.1642-1652.

The UTXO Model

The following is a bitcoin transaction overview:

1. **Creating a Transaction**: When you send bitcoin, you create a transaction from your digital wallet. This transaction includes the sender's address (public key), the recipient's address (public key), the amount of Bitcoin to be sent, and a transaction fee that you're willing to pay to the miners.

2. **Digital Signatures**: To prove that you are the owner of the bitcoin you want to send, the transaction must be signed using your private key through a cryptographic process. This is known as a digital signature. It's essential to keep your private key secret because it's like your digital password.

The UTXO Model

The following is a bitcoin transaction overview:

3. Broadcasting and confirmations: Once signed, the transaction is broadcasted to the Bitcoin network and goes into the mempool, which is like a waiting room for transactions that are waiting to be confirmed. Miners can pick transactions from the mempool to form new blocks. The first miner to solve a difficult mathematical problem gets to make the next block. The winning miner broadcasts its new block, which gets confirmed by the rest of the network.

4. Transaction Finalization: Once confirmed, the new block is added to each network participant’s copy of the blockchain. The transactions in the new block are considered to be confirmed. However, it's common practice to wait for at least six confirmations (six more blocks to be added after the block containing your transaction) to consider the transaction final. This is to ensure that the transaction won't be reversed or double-spent in case of a temporary fork in the blockchain.

The UTXO Model

Tom

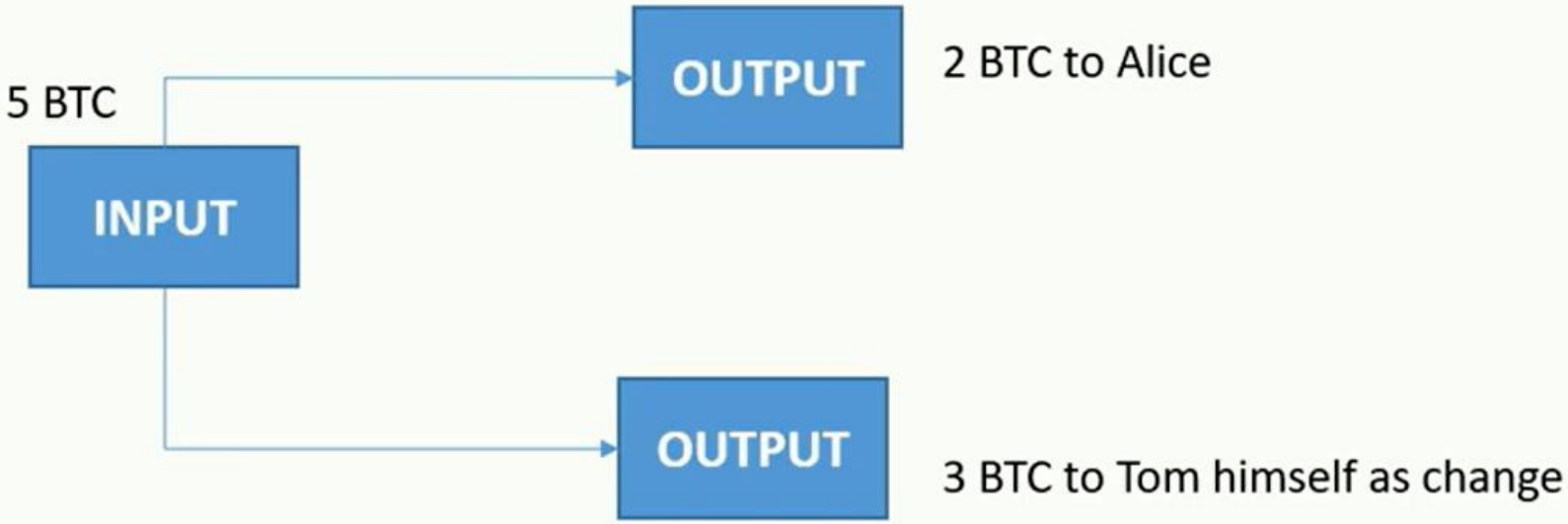
Address : 0x123
Available : 5 BTC

Alice

Address : 0x456
Available : 0 BTC

1. Tom sends 2 BTC to Alice

This becomes a transaction



The UTXO Model

Unspent Transaction output

Tom
Address : 0x123
Available : 5 BTC

How to calculate Tom's Balance
(Our example : 5 BTC)

This can be found from UTXO database which resides in each Bitcoin Wallet

- Tom's balance is calculated real time by looking up UTXO database
- If Tom initiates a transfer of 2 BTC to Alice then the above *inputs would be locked* and sent as part of transaction
  - 2 BTC → Alice
  - 3 BTC → to Himself

The UTXO Model

Unspent Transaction output --continued

Address : 0x123
Available : 5 BTC

Tom

Alice

Address : 0x456
Available : 0 BTC

5BTC to be transferred to Alice

2 BTC INPUT
1 BTC INPUT
1.5 BTC INPUT
0.5 BTC INPUT

Unlock the inputs

5 BTC

Lock the Output

2 BTC OUTPUT
1 BTC OUTPUT
1.5 BTC OUTPUT
0.5 BTC OUTPUT

3 BTC

Structure of a Transaction

VERSION	INPUT COUNTER	INPUTS	OUTPUT COUNTER	OUTPUTS	LOCK TIMESTAMP

The UTXO Model

Unspent Transaction output --continued

Address : 0x123
Available : 5 BTC

Tom

Alice

Address : 0x456
Available : 0 BTC

5BTC to be transferred to Alice

2 BTC INPUT
1 BTC INPUT
1.5 BTC INPUT
0.5 BTC INPUT

Unlock the inputs

5 BTC

Lock the Output

2 BTC OUTPUT
1 BTC OUTPUT
1.5 BTC OUTPUT
0.5 BTC OUTPUT

3 BTC

Structure of a Transaction

VERSION	INPUT COUNTER	INPUTS	OUTPUT COUNTER	OUTPUTS	LOCK TIMESTAMP

The UTXO Model

Example:
Mark wants to send 1 BTC to Jessica. To do this, he uses his private key to 'sign' a message with the transaction-specific details. This message, which must be broadcast to the network, will contain the following:

1) Inputs: This contains information about the bitcoin previously sent to Mark's address. For example, imagine Mark previously received 0.6 BTC from Alice and 0.6 BTC from Bob. Now, in order to send 1 BTC to Jessica, there might be two inputs: one input of 0.6 BTC previously from Alice and one input of 0.6 BTC previously from Bob.

2) Amount: In this case, the amount Mark wants to send is 1 BTC.

3) Outputs: There are two outputs. The first is 1 BTC to Jessica’s address. The second is 0.2 BTC returned as 'change' to Mark. This second output is calculated as the total of the inputs [0.6 + 0.6 = 1.2], minus the amount Mark wants to send [1 BTC].

The UTXO Model

Example:

Step 3: Broadcasting and confirmations:

- Here Mark (via his wallet software) will broadcast his proposed transaction to the Bitcoin network.

- A special group of participants in the network known as 'miners' verify that Mark's keys are able to access the inputs (i.e. the address(s)) from where he previously received the bitcoin he claims to control. Miners also gather together a list of other transactions that were broadcast to the network around the same time as Mark's and form them into a block.

- Any miner who has completed the 'Proof of Work' is permitted to propose a new block that will be added or 'attached' to the chain and by referencing the last block.

- That new block is then broadcast to the network.

The UTXO Model

Example:
Step 3: Broadcasting and confirmations:

- If other network participants (nodes) agree it's a valid block (ie. the transactions it contains follows all the rules of the protocol and it properly references the previous block), they will pass it along.

- Eventually, another miner will build on top of it by referencing it as the previous block when proposing the next block. Any transactions that were in the previous block will now have been 'confirmed' by the next miner. As blocks are added to the chain, the number of confirmations of Mark's transaction increases.

The UTXO Model

Example:
Step 3: Broadcasting and confirmations:

- Each block can only contain a certain number of transactions, and that number is determined largely by the space available in each block, or the 'block size,' which is 1MB.

- The limited space gives rise to the fee market, where miners, who collect fees, choose to include in the next block only those transactions which have included a high enough fee.

- Thus higher fees act as incentive for miners to prioritize your transactions.

The UTXO Model

Example:
Transaction fees:

- Fees for sending bitcoin could be anywhere from a few cents all the way up to $100. The reason for the big variation is that Bitcoin fees depend on both supply and demand (ie. how congested the network is at a given time) and the "size" of your transaction.

- Size is affected primarily by inputs, so if your transaction has many inputs, it will take up more block space, and demand a higher fee. For example, if you want to send 10 BTC, there's a good chance your transaction will require more inputs than if you want to send 1 BTC.

- The 10 BTC transaction might consist of 5+2+1+1+1 (so a total of 5 inputs) while the 1 BTC transaction might be just two inputs as in our Mark/Jessica example above.

The UTXO Model

Example:
Transaction fees:

- Many wallets, allow users to manually set transaction fees.
- This helps you to avoid overpaying.
- For example, if you're not in a rush, you can set the fee the lower such that it will be picked up by a miner when the network is less congested.
- You can also ensure your transactions are processed immediately by increasing your fee.

The UTXO Model

- Here, there are **four** inputs
- Two inputs (2 input of 0.0027867 BTC and 3 input of 0.0034977 BTC) come from the same address (1HXpg8D9AMGFVZ9FEU2tkZYvAZ8xBhVudo)
- The other two inputs are from the addresses 14yPyVmGhNCSM9JgaabRZ8C3cT2RWEGd71 and 1MXDLBc2Tq2hnQ2x5qXTEPUen5xq9hDA39
- Total of the inputs is 0.0128 BTC
- The two outputs of value 0.00865732 BTC and 0.0028 BTC are going to two addresses as shown in fig.
- The sum of two are 0.01145732 BTC
- The **difference** between the inputs and outputs is 0.00134268 BTC, is paid to the miner who added the block the transaction is in to the blockchain (known as “mining” a block)

Transactions & Transaction Fee in Blockchain

Transactions in Blockchain

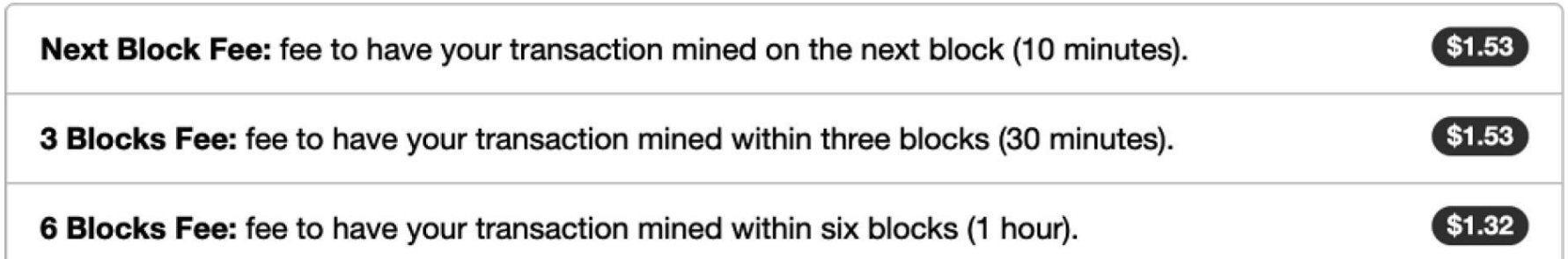
- A transaction in blockchain is a **unit of data** that represents a **movement or transfer of value** from one address to another
- A transaction which is successfully added to blockchain ledger (by mining a block containing that transaction) is called **confirmed transaction**
- The sender **digitally signs** a transaction using his/her **private** key and then sends it to the blockchain network
- The nodes in blockchain network uses sender's **public** key to **verify** the user and then validates the transaction
- The sender sends some additional value (other than the value to be transferred) which is called as **transaction fee**
- This transaction fee is **awarded** to the miner which successfully mines a new block containing that transaction

Transaction Fee

- At the beginning of a new mining cycle, miners pick transactions form the pool of unconfirmed transactions
- So, the miner tends to include transactions in the descending order of transaction fee associated with the transaction
  - In other words, miners pick the transactions having higher ‘transaction fee’ so as to maximize their rewards
  - Miners cannot include as much transactions as they wish in one single block
- There is limit to the maximum size of the block that any miner can create
  - Thus, only limited number of transactions can be included in one block
  - In Bitcoin the maximum size of a block can be 1 MB which allows approximately 3,500 transactions per block
- If a user wants to get a transaction confirmed early then higher transaction fee is required.

Bitcoin fee estimator (online)

Varying transaction fee based on how much a user can wait for getting a transaction confirmed!



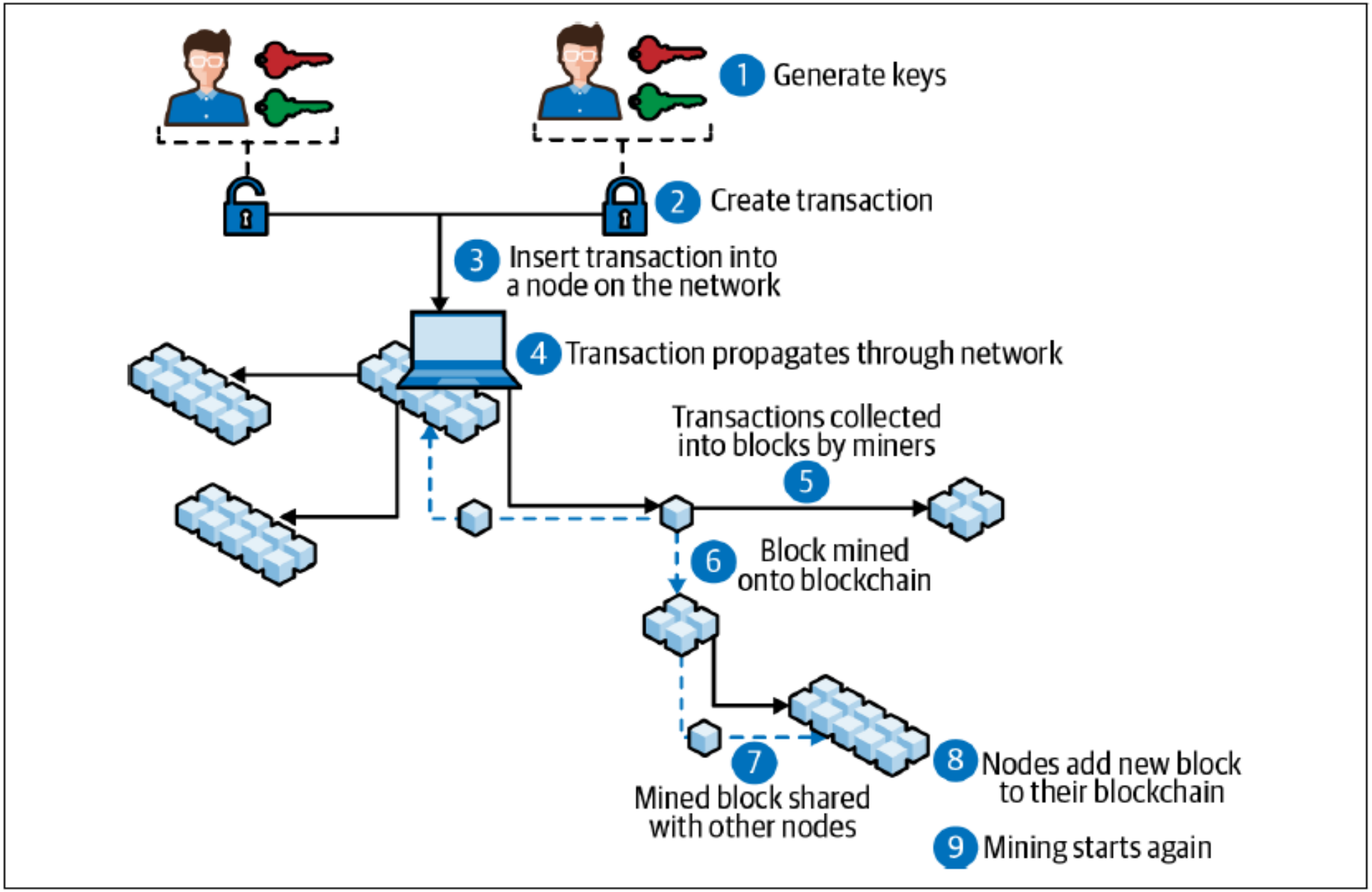


Figure 2-4. Series of events involved in executing a bitcoin transaction—"block mined onto blockchain" refers to miners adding a new block to be confirmed by the network

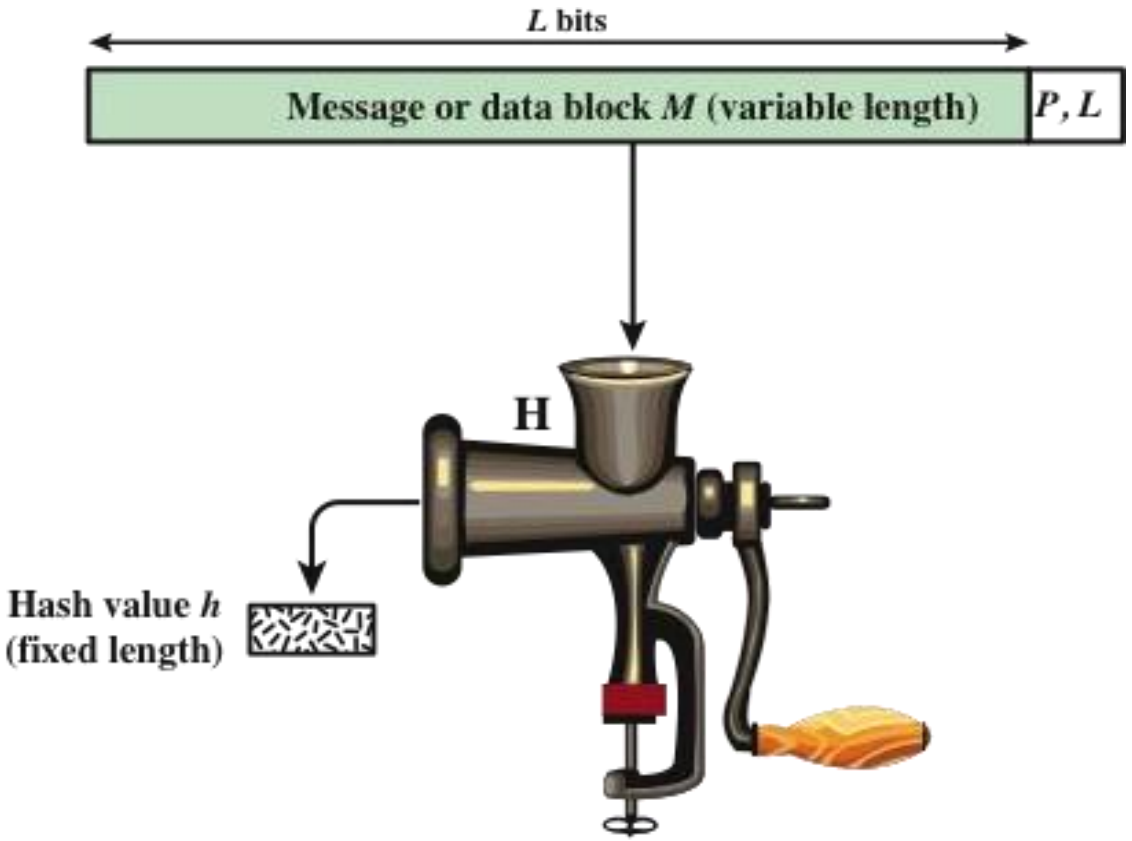
Hashing

Hashing

- Hashing is process that uses **hash function**
- Hash function **H** takes data of **variable-length as input** and produces a **fixed-length output** called as **hash value**

$$h = H(M)$$

- Also called as **one-way function**
  - Reverse engineering is not possible
- Small change in input results in complete change in hash value
- Hash is **deterministic** meaning that every time the same input data results in exactly same output
- **Collision resistant** - Very unlikely there will be same hash value for two different input data
- **SHA-256**, commonly used by **Bitcoin**, whereas, **Keccak-256**, commonly used by **Ethereum**



P, L = padding plus length field

Source: William Stallings, "Cryptography and Network Security, Principles and Practices", 5th Edition, Pearson, 2017.

Hashing

- A common use case for a hash is a secure website storing a hash of your password in its database.
- Let's say your password for the website *www.store.com* is *FNj`{:;`k#F43rQ*\.
- For extra protection the website's database will store not the password, but a hash of the password. If the website uses the hash function SHA-256, the resulting string stored in the database will be:

SHA-256("FNj`{:;`k#F43rQ\")
=6586BC035202DFF98A67B814ACA615E613CBBFAE8FFA8F4A475DA0FAEF079C9D

- Then, when you log in, the website only needs to verify the entered password by comparing the hash of the string you typed in to the hash stored in its database.
- This process makes the website more secure because if a hacker breaks into the database, they'll only get the hashes of customer passwords.

Digital Signature

Signing and validating transactions

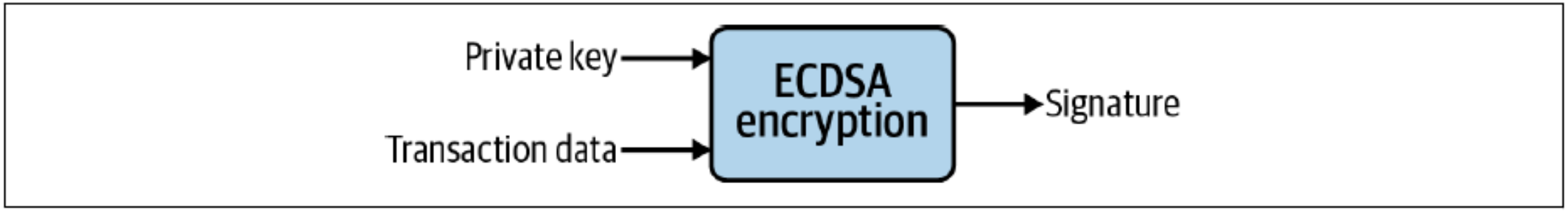


Figure 2-8. Encryption process to generate a transaction signature

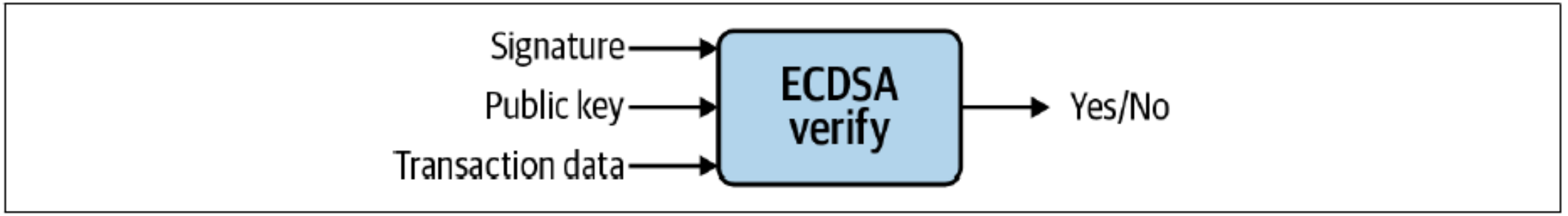
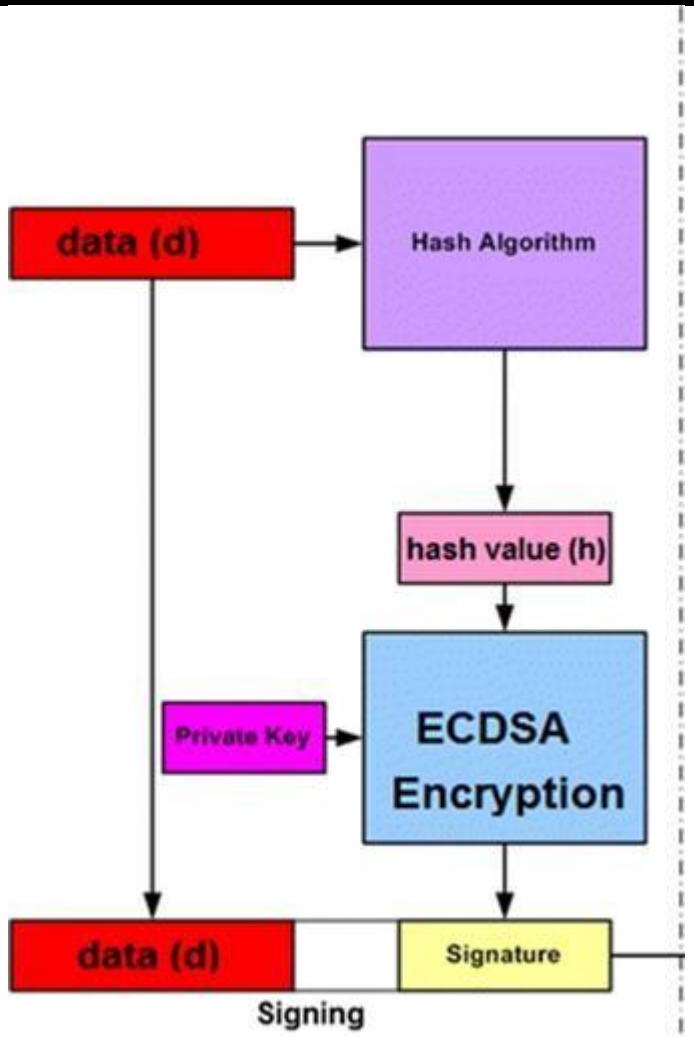


Figure 2-9. Verifying the signature on a transaction

ECDSA – Elliptic Curve Digital Signature Algorithm

Digital Signature

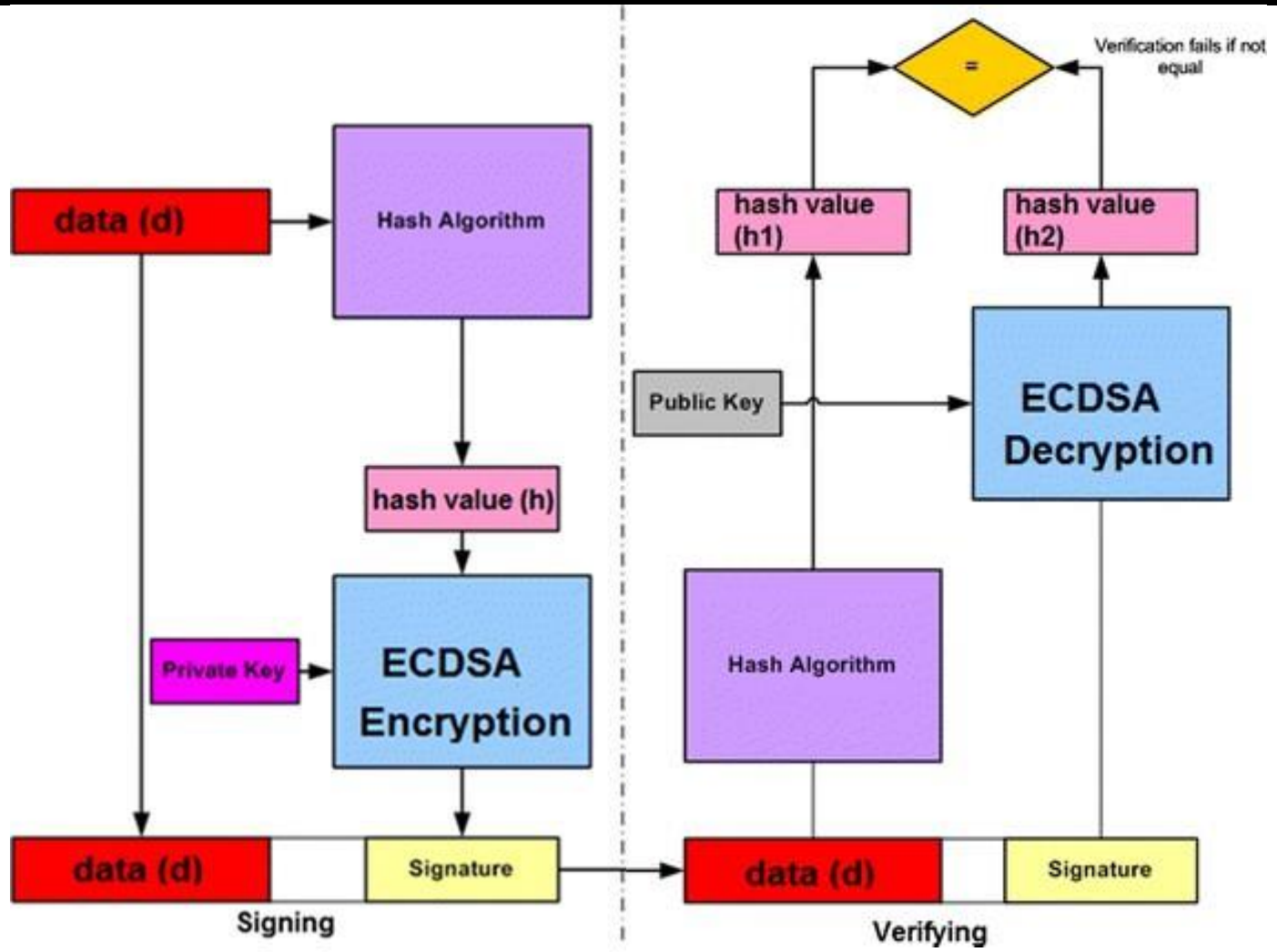
Digital signing by sender



Source: Younis, M.I. and Abdulkareem, M.H., 2017. ITPMAP: an improved three-pass mutual authentication protocol for secure RFID systems. Wireless Personal Communications, 96, pp.65-101.

Digital Signature

Digital signing by sender

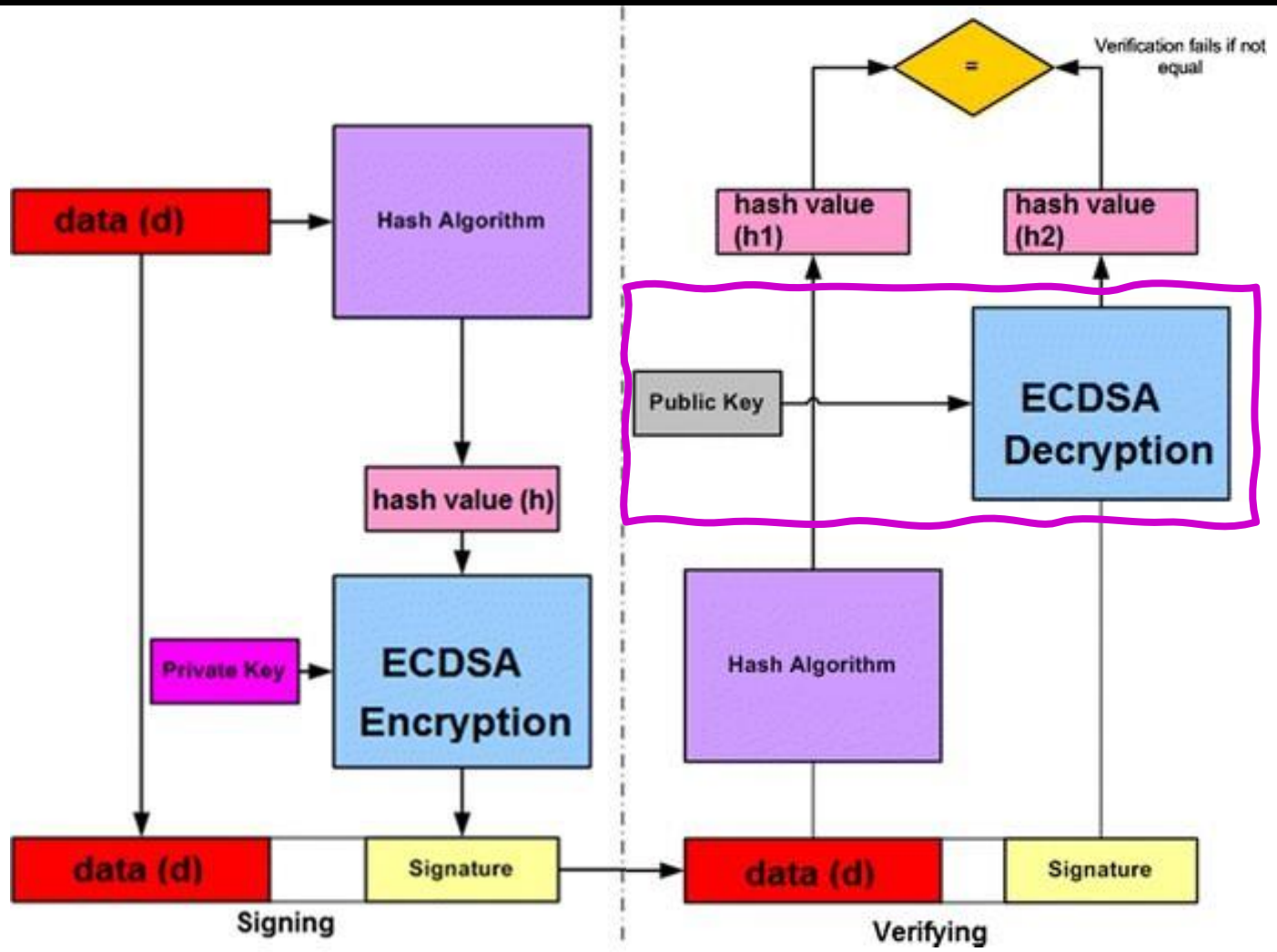


Verification of digital sign by recipient

Source: Younis, M.I. and Abdulkareem, M.H., 2017. ITPMAP: an improved three-pass mutual authentication protocol for secure RFID systems. Wireless Personal Communications, 96, pp.65-101.

Digital Signature

Digital signing by sender



Verification of digital sign by recipient

Source: Younis, M.I. and Abdulkareem, M.H., 2017. ITPMAP: an improved three-pass mutual authentication protocol for secure RFID systems. Wireless Personal Communications, 96, pp.65-101.

Merkle Tree & Merkle Root

Merkle Tree & Merkle Root

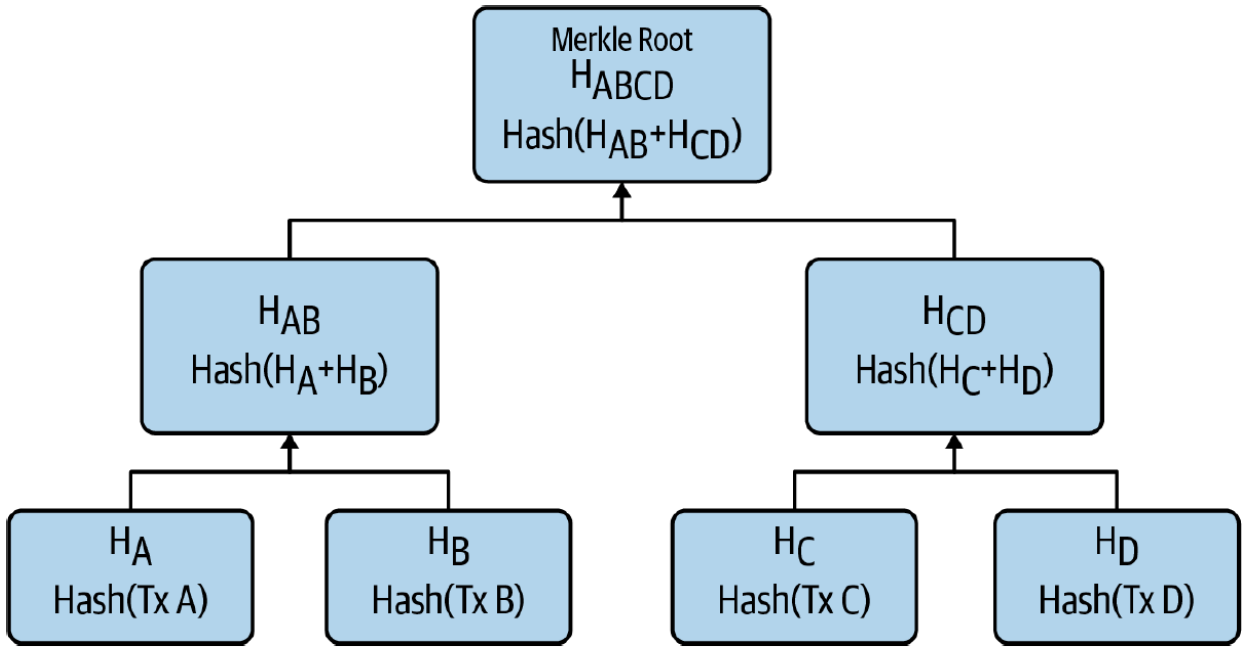


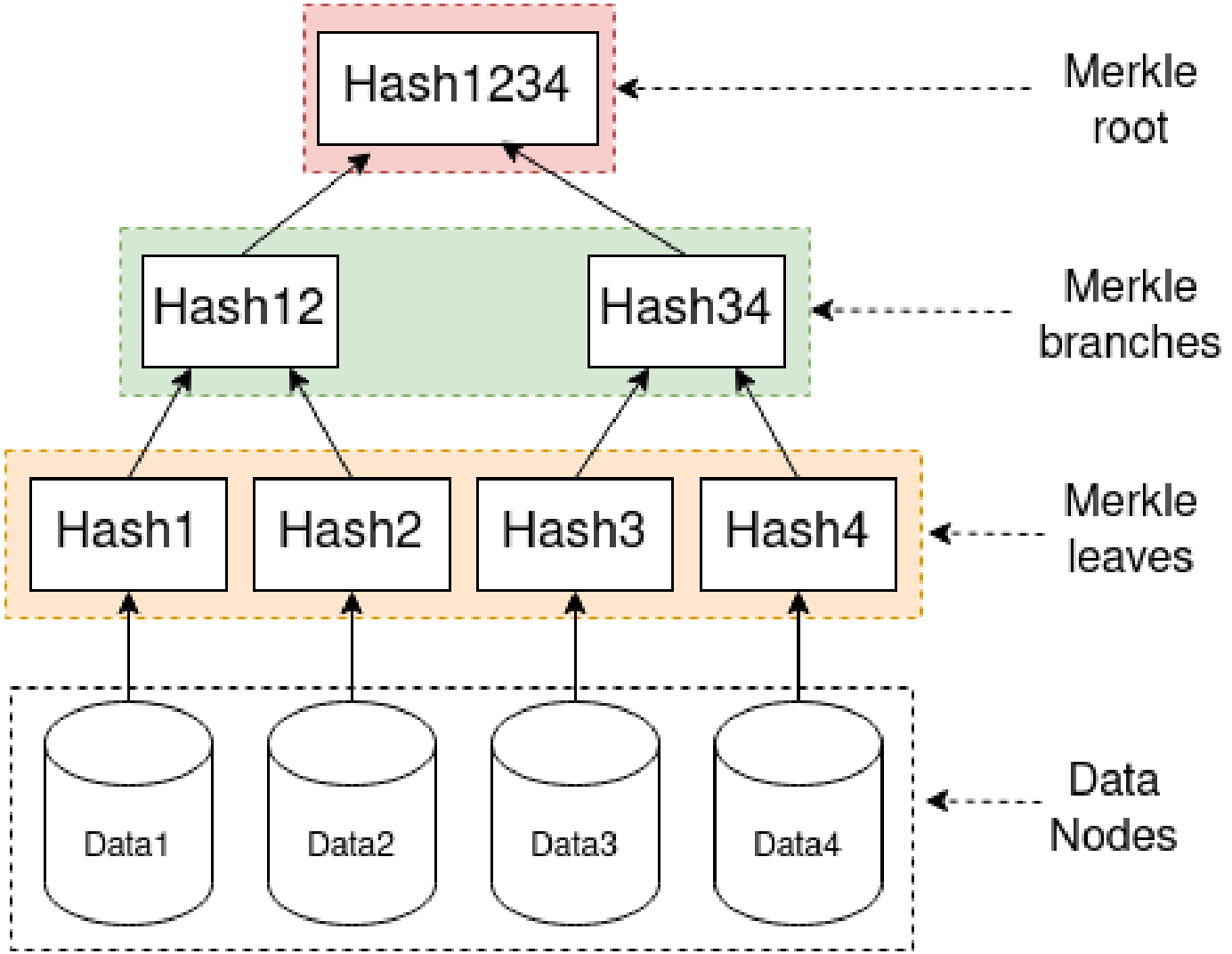
Figure 2-5. Flow chart of a sample Merkle tree

- Names comes from the inventor – **Ralph Markle**
- Starts with computation of hash values of individual transaction or **leaf node** is calculated.
- These hash values are paired in two and further hashed, as the figure shows.
- For odd transactions, the initial hash is duplicated and hashed together.
- The process continues till only one hash is left. This hash is known as the **Merkle root**.

Advantages

- This allows for **secure** and **compact** representation of large set of transactions. In other words, Merkel root is **256 bits digital finger print** of the state of all the transactions in a given block
- Just by comparing the Merkle root, any two nodes can be sure that they have exact list of transactions
- **Attacker** if makes a slight change in a transaction at **one node** then the Merkle root hash will **change** and will not match with other nodes → Thus **tampering** gets identified and discarded
- Helps in building light weight software for easy verification of transactions

Merkle Tree



Binary Inverted Tree of Hashes

Data i = Transaction i

Source: https://medium.com/coinmonks/merkle-trees-concepts-and-use-cases-5da873702318 by Teemu Kanstren

Bitcoin Blockchain Merkle Tree

Block 125552

Hash	00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d
Confirmations	518,890
Timestamp	2011-05-22 01:26
Height	125552
Miner	Unknown
Number of Transactions	4
Difficulty	244,112.49
Merkle root	2b12fcf1b09288fcaff797d71e950e71ae42b91e8bdb2304758dfcffc2b620e3

Figure 2-6. Overview of Bitcoin block #125552

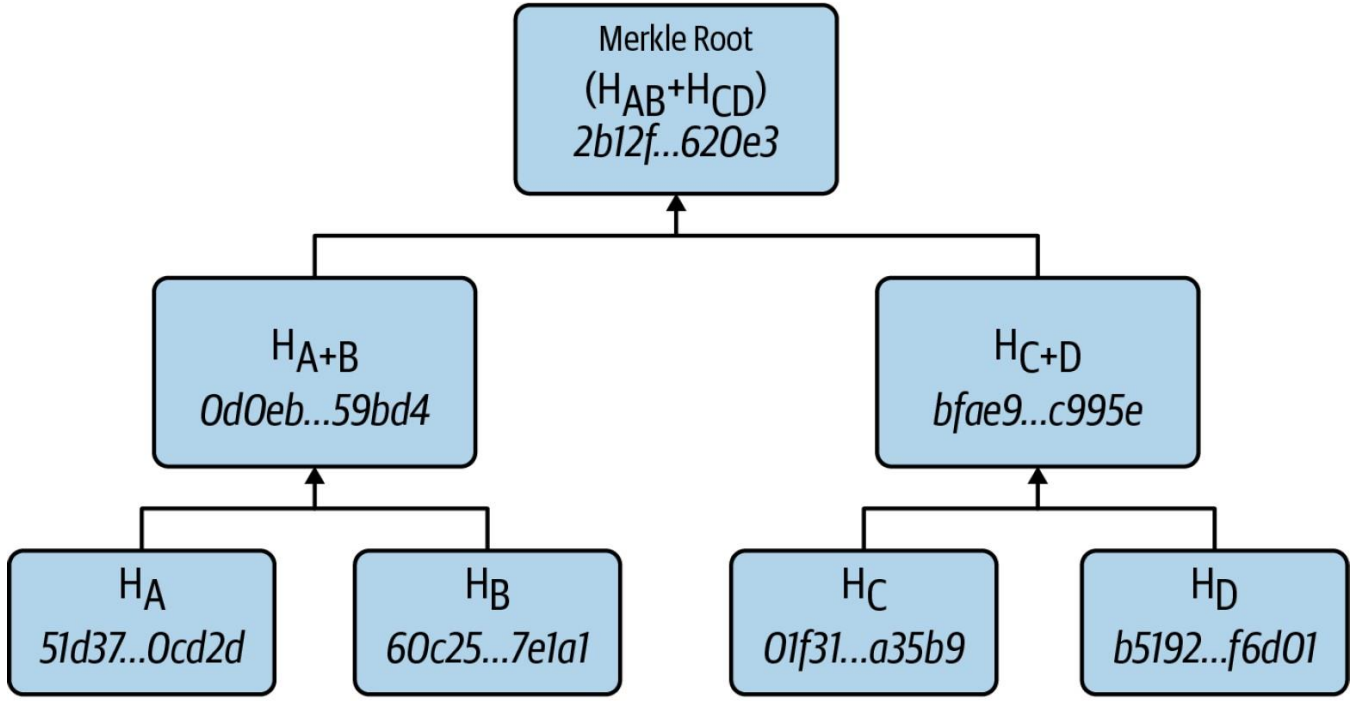


Figure 2-7. Flow chart of the example Merkle tree

H_A = 51d37bdd871c9e1f4d5541be67a6ab625e32028744d7d4609d0c37747b40cd2d

H_B = 60c25dda8d41f8d3d7d5c6240

H_C = 01f314cdd8566d3e5dbdd97de2d9fbfbfd6873e916a00d48758282cbb81a45b9

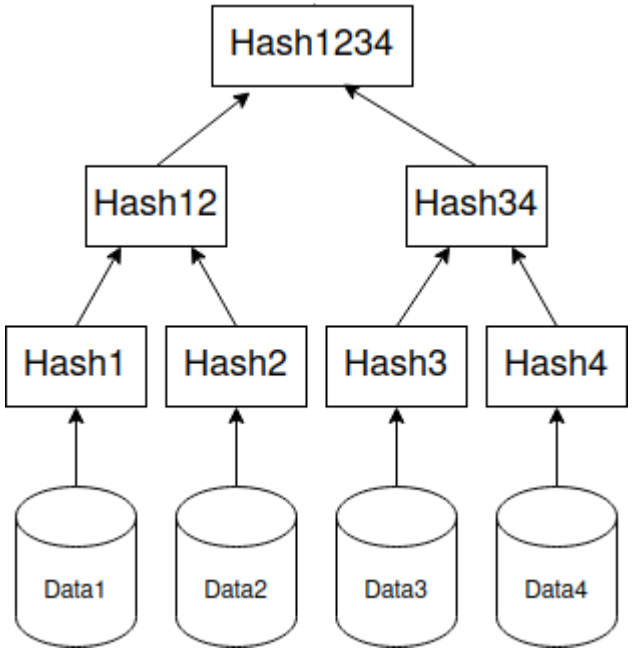
H_D = b519286a1040da6ad83c783eb2872659eaf57b1bec088e614776ffe7dc8f6d01

H_{A+B} = 0d0eb1b4c4b49fd27d100e9cce555d4110594661b1b8ac05a4b8879c84959bd4

H_{C+D} = bfae954bdb9653ceba3721e85a122fba3a585c5762b5ca5abe117b30c36c995e

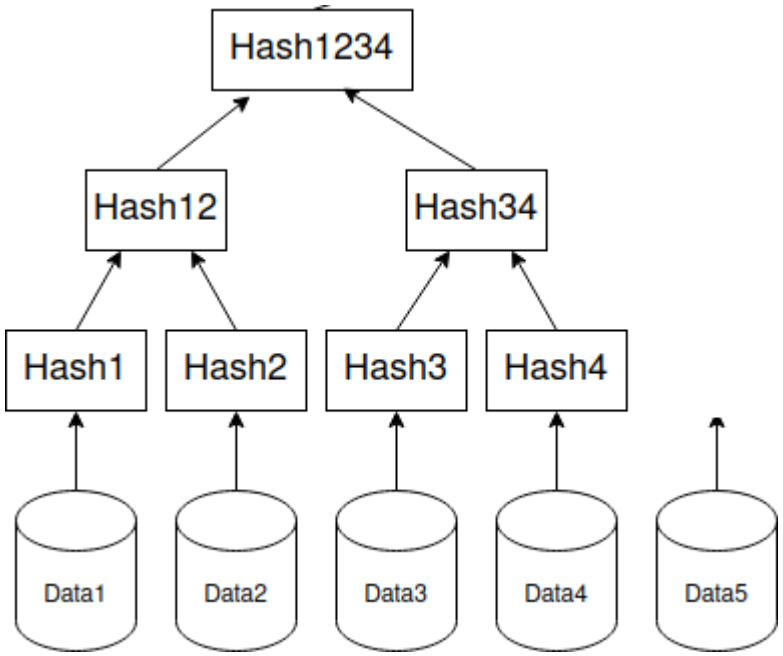
$H_{A+B} + H_{C+D}$ = Merkle root = 2b12fcf1b09288fcaff797d71e950e71ae42b91e8bdb2304758dfcffc2b620e3

Bitcoin Blockchain Merkle Tree



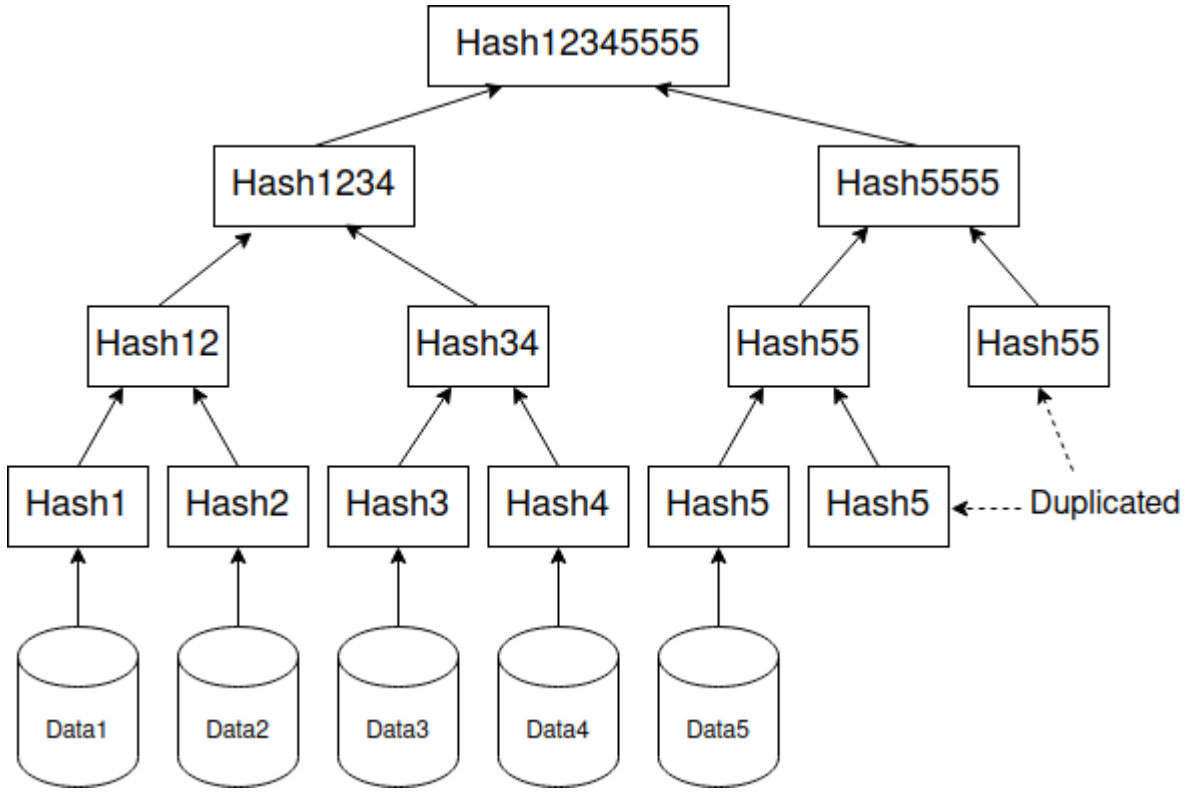
Source: https://medium.com/coinmonks/merkle-trees-concepts-and-use-cases-5da873702318 by Teemu Kanstren

Bitcoin Blockchain Merkle Tree



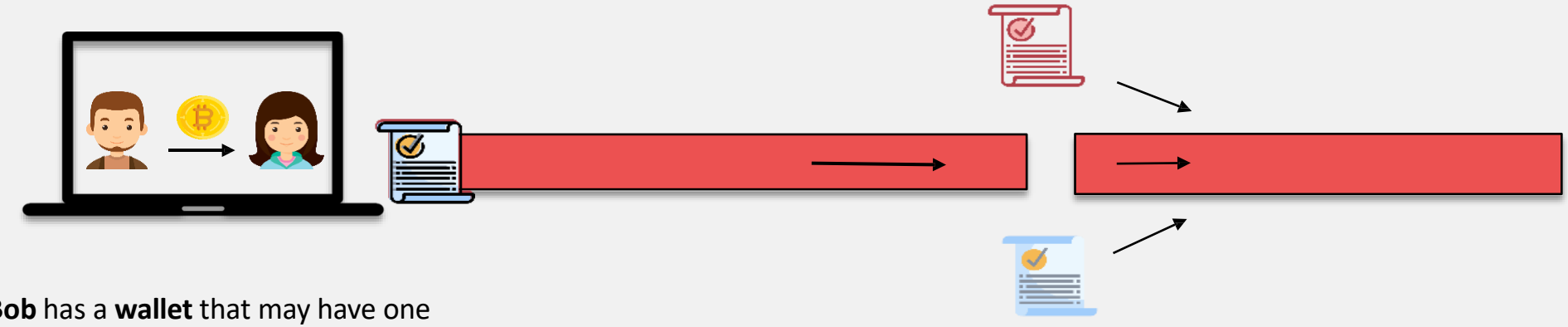
Source: https://medium.com/coinmonks/merkle-trees-concepts-and-use-cases-5da873702318 by Teemu Kanstren

Bitcoin Blockchain Merkle Tree



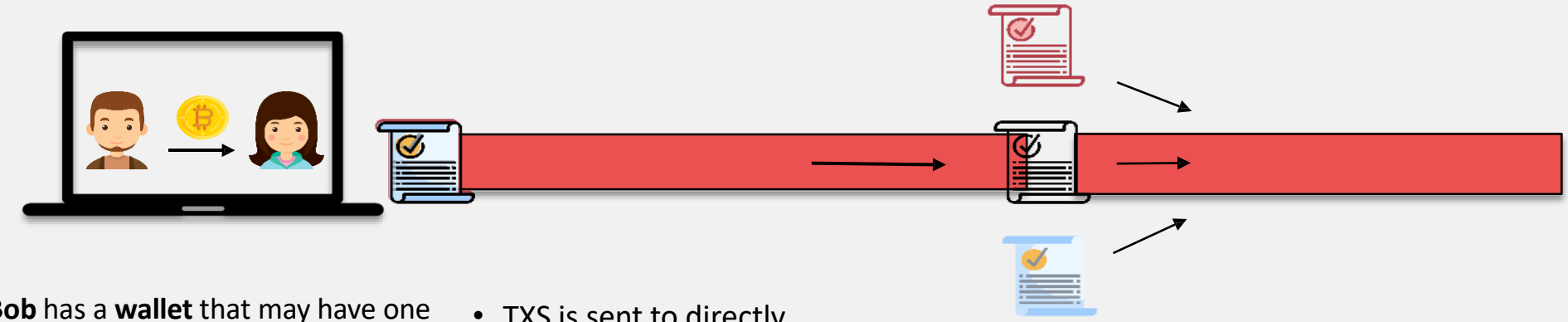
Source: https://medium.com/coinmonks/merkle-trees-concepts-and-use-cases-5da873702318 by Teemu Kanstren

Blockchain Process Flow



- **Bob** has a **wallet** that may have one or more addresses
- Using its wallet bob initiates a transfer of X Bitcoin to **Alice**
- A transaction (TXS) is created and is **digitally signed** by private key of Bob

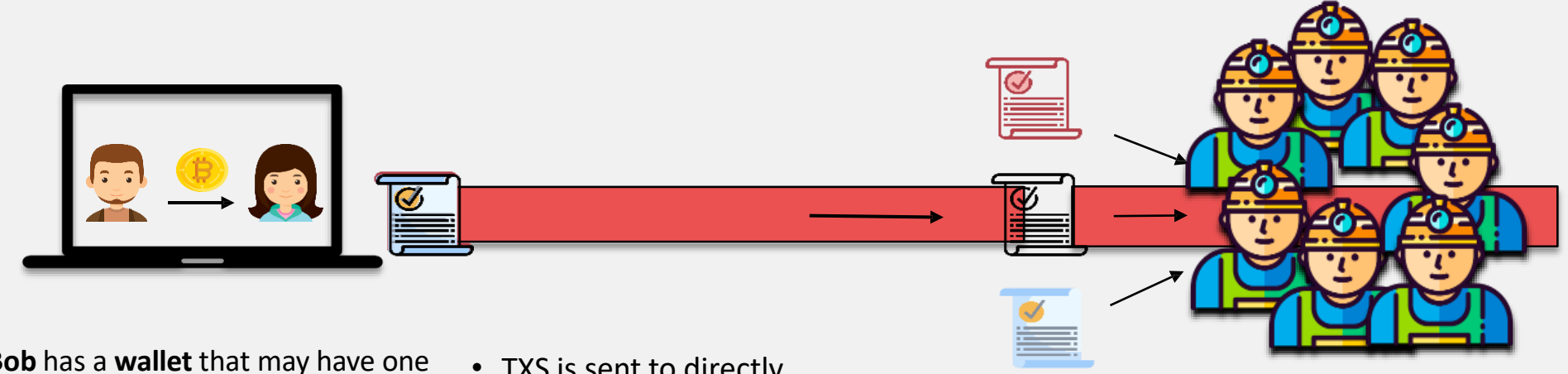
Blockchain Process Flow



- **Bob** has a **wallet** that may have one or more addresses
- Using its wallet bob initiates a transfer of X Bitcoin to **Alice**
- A transaction (TXS) is created and is **digitally signed** by private key of Bob

- TXS is sent to directly connected node(s) which forward further
- Thus, TXS gets broadcasted to the P2P decentralized blockchain network

Blockchain Process Flow



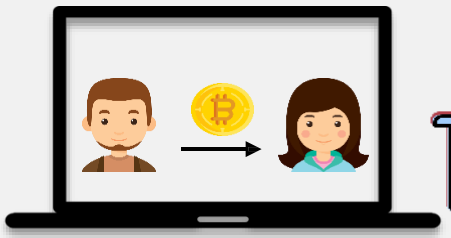
- **Bob** has a **wallet** that may have one or more addresses
- Using its wallet bob initiates a transfer of X Bitcoin to **Alice**
- A transaction (TXS) is created and is **digitally signed** by private key of Bob

- TXS is sent to directly connected node(s) which forward further
- Thus, TXS gets broadcasted to the P2P decentralized blockchain network

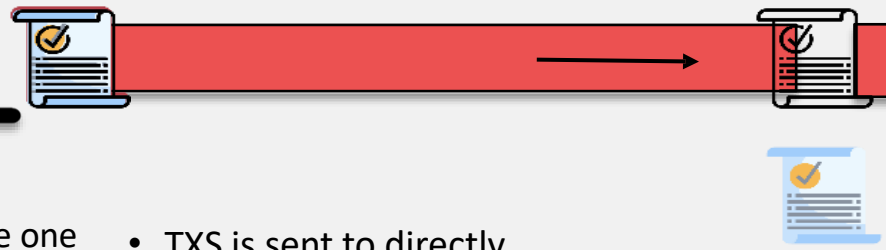
- Miners collect the TXS & store in mem-pool of unconfirmed TXSs for a time window
- Unconfirmed TXS picked up by Miners and are verified and validated
- Creation of **Merkle Tree** (TXS as leaf nodes) and computation of **Merkle root hash**

Blockchain Process Flow

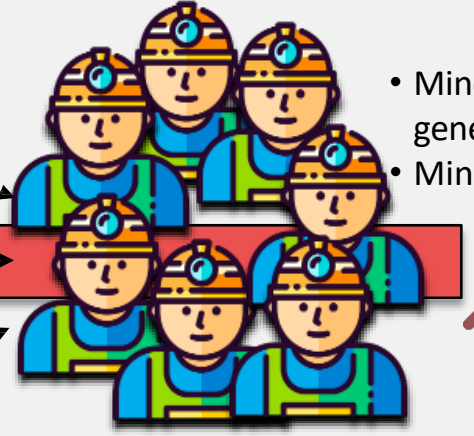
Blockchain Process Flow



- **Bob** has a **wallet** that may have one or more addresses
- Using its wallet bob initiates a transfer of X Bitcoin to **Alice**
- A transaction (TXS) is created and is **digitally signed** by private key of Bob



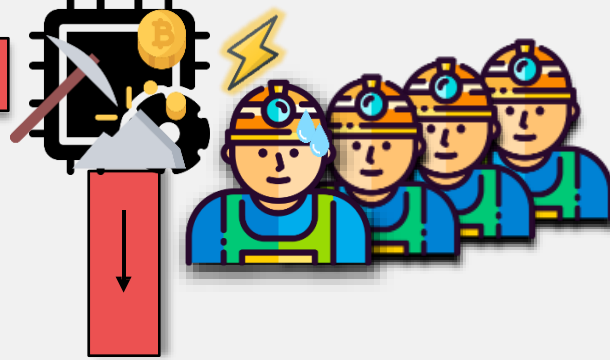
- TXS is sent to directly connected node(s) which forward further
- Thus, TXS gets broadcasted to the P2P decentralized blockchain network



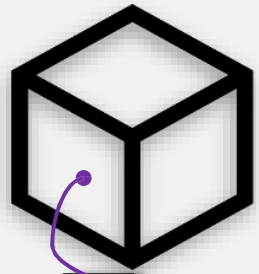
- Miners collect the TXS & store in mem-pool of unconfirmed TXSs for a time window
- Unconfirmed TXS picked up by Miners and are verified and validated
- Creation of **Merkle Tree** (TXS as leaf nodes) and computation of **Merkle root hash**

Block Generation

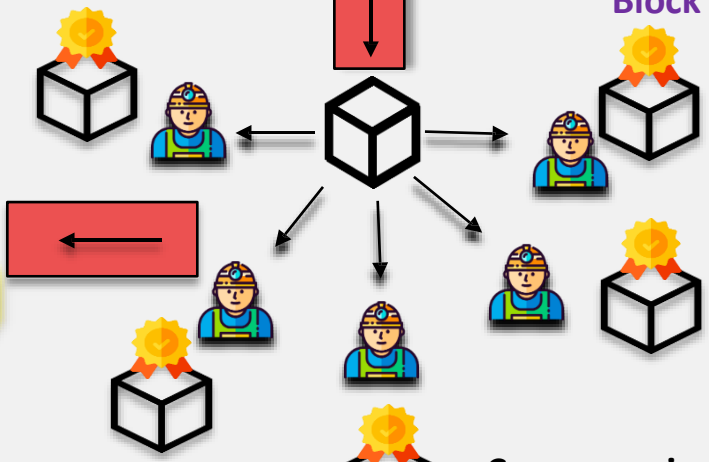
- Miners participate in the process of generating new block
- Miner who finds valid block broadcasts it



Block Formation

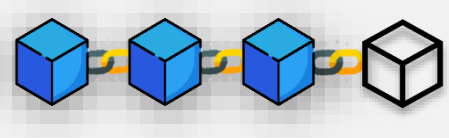


Previous Block Hash

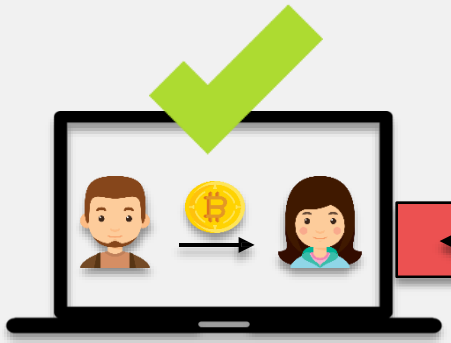


- **Consensus is established**

Bitcoin reward to winning Miner



New block appended to the Blockchain



- **Transaction is confirmed**
- **Alice gets Bitcoin credited**

The Coinbase Transaction

- All transactions that take place on the cryptocurrency network are not the result of payment between two people.
- Some transactions are a little bit different.
- The first transaction that took place was in Bitcoin.
- It was a special transaction that formatted reward transactions for miners inside the Genesis block (the very first block of a blockchain).
- Such reward transactions are specially allocated to the miner for their work. This type of transaction is known as a **Coinbase transaction**.

The Coinbase Transaction

- The coinbase transaction is the **first transaction** of each new block mined in the blockchain
- This is the way the Bitcoin (or similar) blockchain **mints new bitcoins** and gives them as **reward** to the miner who successfully mined that new block
- Thus, the overall supply of the bitcoins increases
- A coinbase transaction has two parts:
  - **Block Rewards**: It is of new bitcoins (BTCs) which are generated by the system and given to the wining miner
  - **Transaction Fees**: This value is some of all the transaction fees that comes from the transactions which are confirmed by the newly mined block

Bitcoin Transaction Security

- In general, in financial world there are two type of transactions – **Push** and **Pull**
- In **push** transaction,
  - The **sender** (who owns an account) **initiates** the **transaction** to transfer some fund to a given address (recipient)
  - Sender **need <u>not</u>** to share any information related to his/her account with the recipient
- In pull transaction,
  - The **recipient** who is suppose to receive the funds **initiates** the **transaction** (example when using credit card, the merchant initiates a transaction for you)
  - Sender **<u>need</u>** to share their account details with the receiver
  - To overcome this issue, **pull** payment networks (like **Visa**) provide **chargebacks**, or the ability to **dispute** a transaction and ask for a **refund**
- Thus, **pull** transaction are **less secure** than **push** transaction
- Thus, **Bitcoin** follows **push** transaction and thus **more** secure

When an fraud transaction can happen in Blockchain?

- When initiating a transaction, a sender never has to reveal any of their account information.
- The only way a fraudulent transaction can take place is if an unauthorized person gets a copy of someone's private key.
- Only if an **attacker** gets **access** to the **private key** of the sender

- With the state-of-the-art technology, it is considered to be impossible to guess or reverse engineer someone's private key

- Brute force is the only way to guess a private key

When an fraud transaction can happen in Blockchain?

- Brute force is the only way to guess a private key

- Since the length of the private key is 256 bits, there will be 2^{256} combinations to test

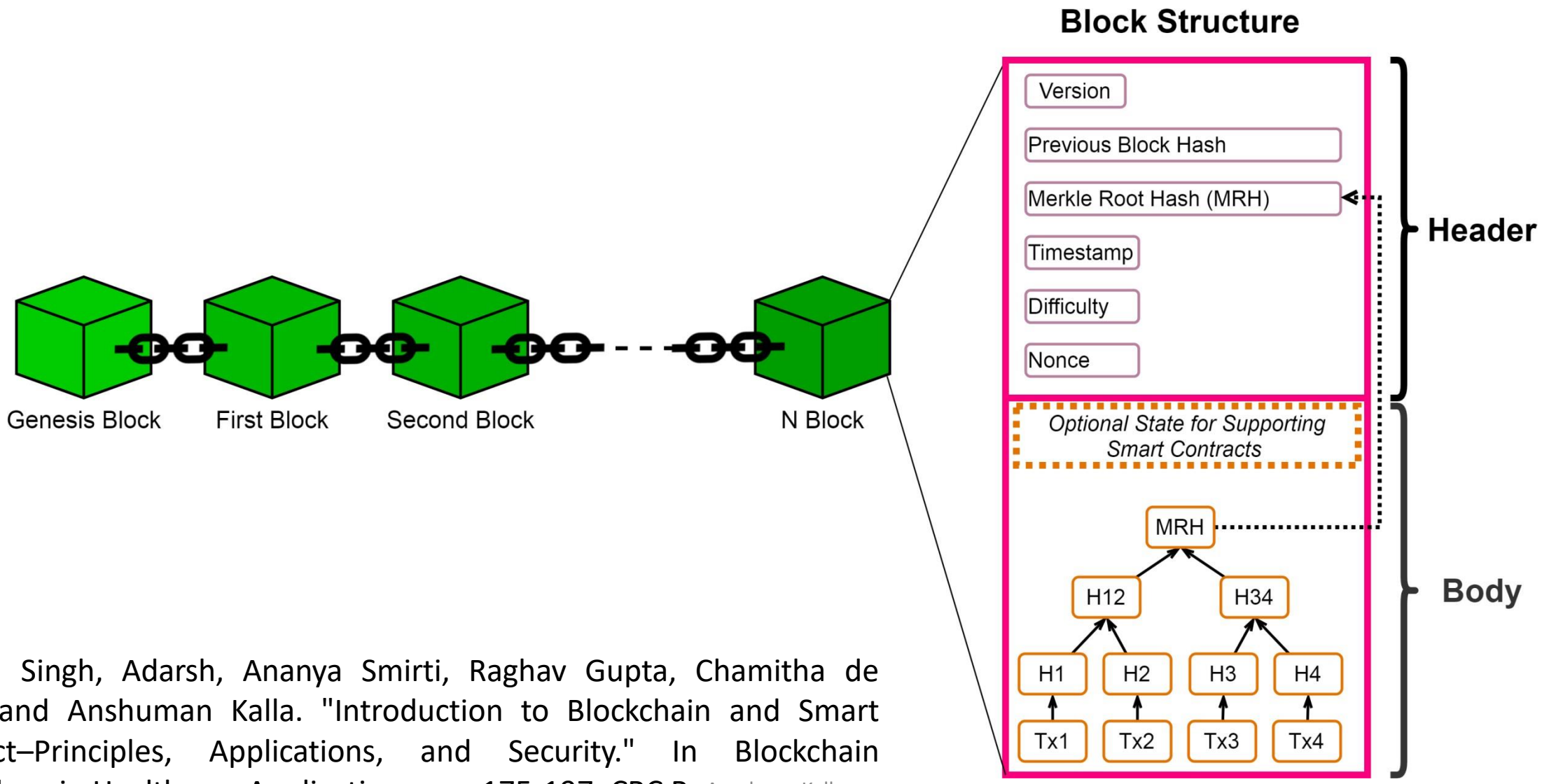
$$2^{256} = 1.15^{77} = 4 \text{ billion}^{8}$$

- To try this many combination it would take following number of years even with the current processing capacity

$$4{,}589{,}678{,}828{,}851^{37} \text{ years}$$

The most common way for an unauthorized person to get hold of a private key is by breaking into an unsecured server or database.

Blockchain (Chain of Blocks) and Block Structure



Source: Singh, Adarsh, Ananya Smirti, Raghav Gupta, Chamitha de Alwis, and Anshuman Kalla. "Introduction to Blockchain and Smart Contract–Principles, Applications, and Security." In Blockchain Technology in Healthcare Applications, pp. 175-197. CRC Press.

Block Hash

- A block hash is computed by hashing the block's **header** part
- Thus, it is a **unique identifier** of each block
- Every block stores the block hash value of **previous block**
- Thus, in Bitcoin block does not contain its own block hash, but it does contain the hash of the previous block it is building on
- Hence the name blockchain → blocks are chained with a hash-based cryptographic chain

Importance of Block Hash

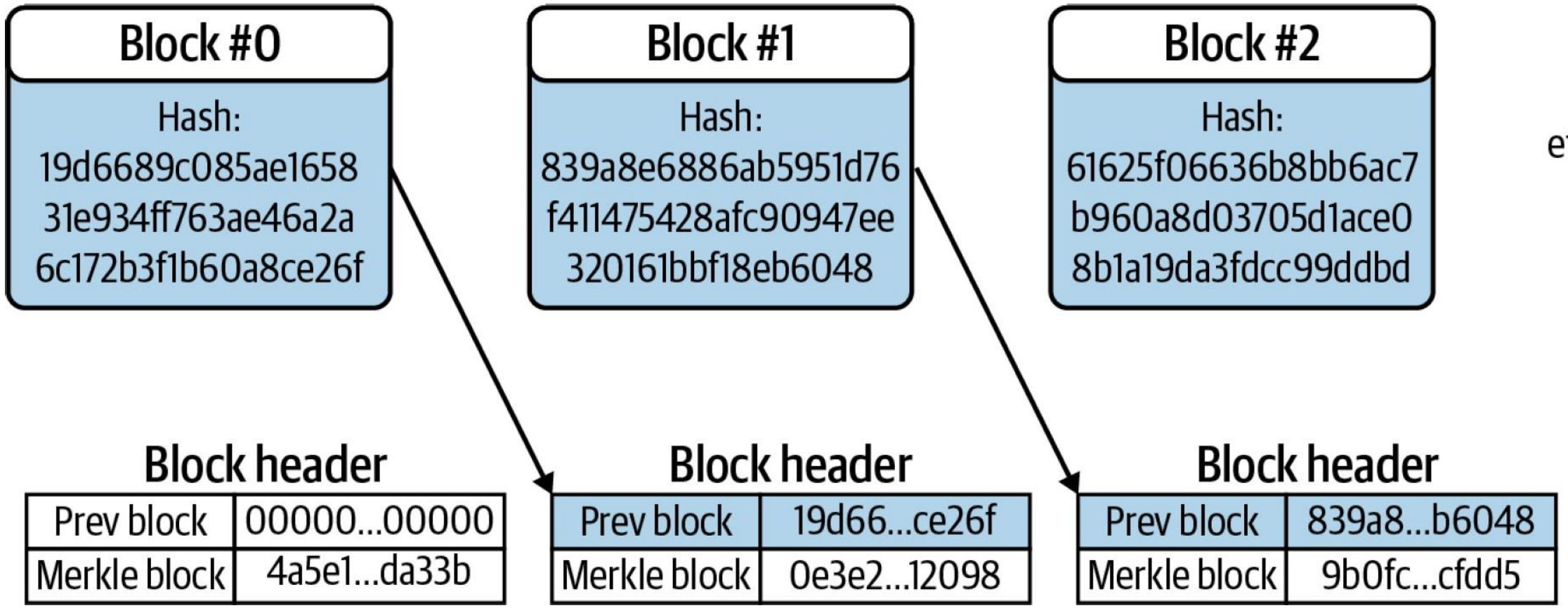
- Block hash of the most recent block is actually a snapshot of the entire blockchain

- In accounting terms, it’s like a balance sheet for the entire network. Every node in the network refers to the block hash to verify that its view of the network is the exact same as everyone else’s. If there’s even one minor difference in a node’s ledger, its hash will look significantly different. This is what makes blockchain tamper-evident; if the content experiences tampering or corruption, the resulting hash will no longer be the same.

Importance of Block Hash

- Any small change (even a single bit flip) in any transaction within any previous blocks (say **x** block)
  - will change the Merkle root hash of **x** block
  - which will change the block hash of that **x** block
  - which will change the previous_block_hash field of the next block **x+1**
  - which will change the block hash of the **x+1** block
  - which will change the previous_block_hash filed of the **x+2** block
  - so on….
- Every node keeps track of the **block hash of the current block** to ensure its view of the current state of the blockchain is same that of others nodes
- This way all the nodes are in **sync** and blockchain becomes **tamper resistant**
- Bitcoin blockchain uses **double SHA-256** hash function on block's header
  → SHA256(SHA256(Block Header))

Importance of Block Hash



Importance of Block Hash

- The block hash at Anonymous #4 is different as compared to the block hash value at all the other nodes.
- This implies that node #4 is either not working fine or it might be compromised node (i.e., under attack)
- Since the block hash of the most current node is the digital finger print of the entire blockchain in 256 bits, it is **very easy for nodes to compare their state of the blockchain**
- Thus, no need to compare the entire blockchain transaction-by-transaction (or bit-by-bit).