# Blockchain Component

DR. RUBA AWADALLAH

# Main Component

Blockchain technology can seem complex; however, it can be simplified by examining each component individually.

At a high level, blockchain technology utilizes well-known computer science mechanisms and cryptographic primitives mixed with record keeping concepts.

**Blockchain main component:** *cryptographic hash functions, transactions, asymmetric-key cryptography, addresses, ledgers, blocks, and how blocks are chained together.*

# Main Component: 1- Hashing and Nonce :

An important component of blockchain technology is the use of cryptographic hash functions for many operations.

**to provide an additional protection when processing personal data.**

**Hash function or hashing:** A method of calculating a relatively unique output (called a hash digest) for an input of nearly any size (a file, text, image, etc.) by applying a cryptographic hash function to the input data.

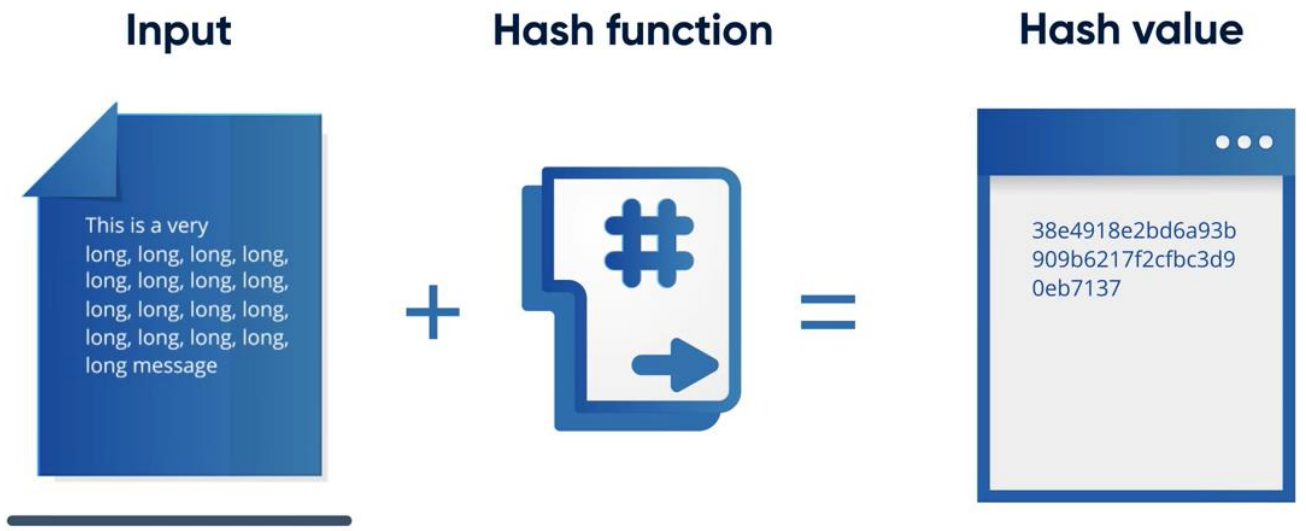
The output is called hash value or code, digest, image or hash.

Often, the term “hash” is used both in reference to the hash function as to the hash output.

The data that are to be run through the hash function are called the message or preimage.

# Hash Function Process

**Hash function is an algorithm with "one-way“ process, which converts an input data of arbitrary length to array of numbers and letters of a fixed length.**



# Example:

For example, if the hash function SHA256 is used to determine the hash value for “Hello” the output shall be the following digest:

SHA256(Hello) =
0xe633f4fc79badea1dc5db970cf397c8248bac47cc3acf9915ba60b5d76b0e88f

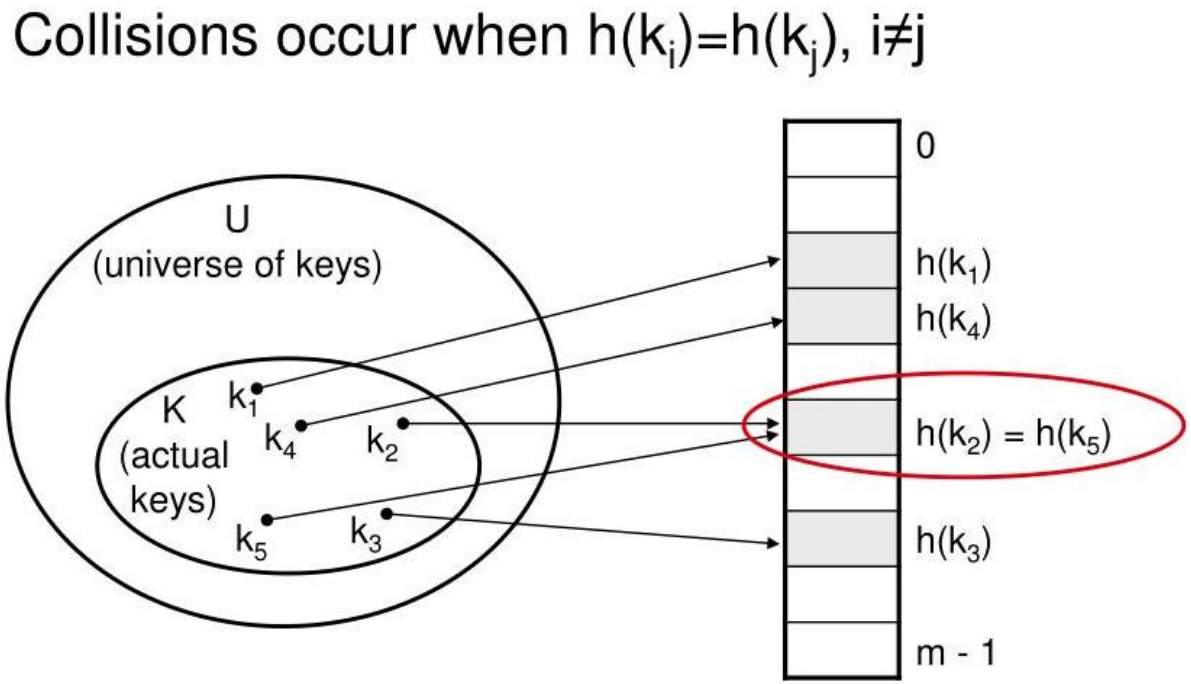
| Input | SHA-256 Digest Value |
|---|---|
| 1 | 0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b |
| 2 | 0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35 |
| hello! | 0xce06092fb948d9ffac7d1a376e404b26b7575bcc11ee05a4615fef4fec3a308b |

You can find a hash functions simulator at https://hash.online-convert.com/

# Hash Functions Properties for BC:

**Cryptographic hash functions exhibit these three properties:**

1. They are “collision-free.” This means that no two input hashes should map to the same output hash.
2. They can be hidden. It should be difficult to guess the input value for a hash function from its output.
3. They should be puzzle-friendly. It should be difficult to select an input that provides a predefined output. Thus, the input should be selected from a distribution that's as wide as possible.



# Nonce:

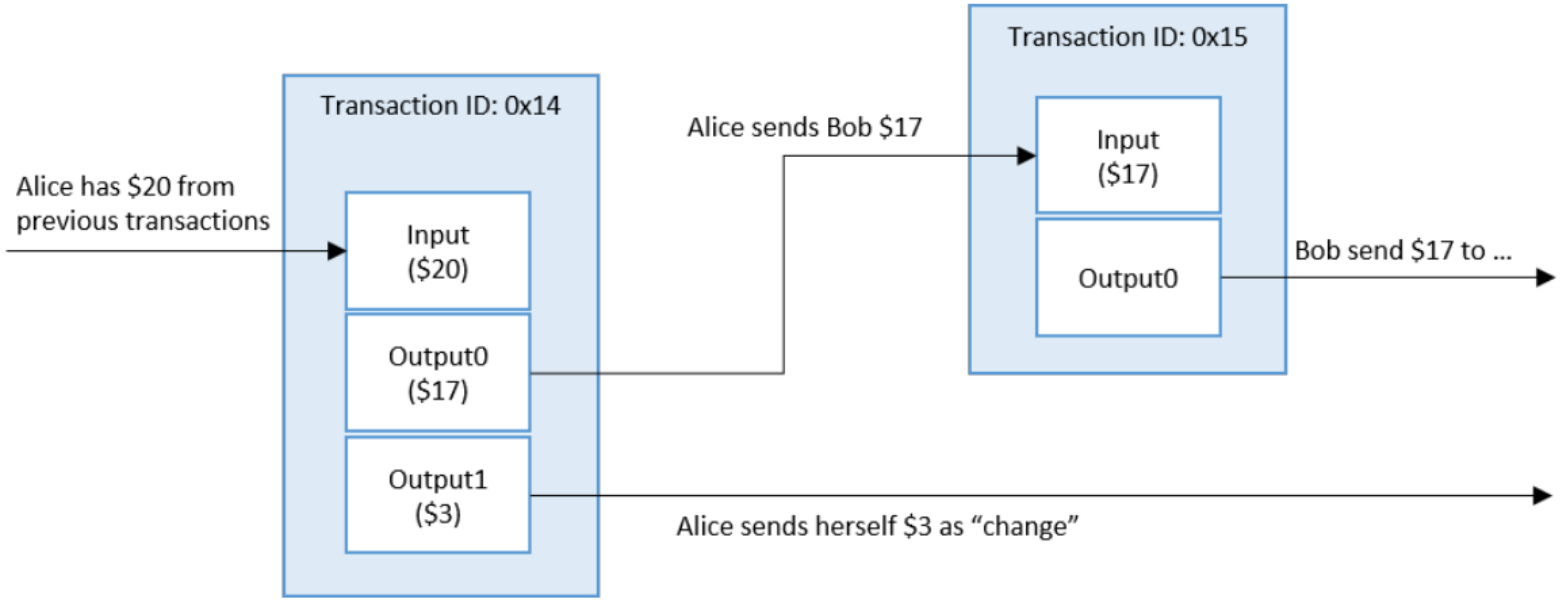
- A cryptographic nonce is an arbitrary number that is only used once.
- A cryptographic nonce can be combined with data to produce different hash digests per nonce:
  - hash (data + nonce) = digest
- Only changing the nonce value provides a mechanism for obtaining different digest values while keeping the same data. This technique is utilized in the proof of work consensus model.

# Main Component: 2- Transaction

- A transaction represents an interaction between parties.
- With cryptocurrencies, a transaction represents a transfer of the cryptocurrency between blockchain network users.
- For business-to-business scenarios, a transaction could be a way of recording activities occurring on digital or physical assets.

# Example of a Cryptocurrency Transaction :

- Each block in a blockchain can contain zero or more transactions. For some blockchain implementations, a constant supply of new blocks (even with zero transactions) is critical to maintain the security of the blockchain network.



# Transaction Data

- The transaction data can be different for every blockchain implementation.
- The mechanism for transacting is largely the same:
  - A blockchain network user sends information to the blockchain network.
  - The information sent may include:

    Sender's address (or another relevant identifier).

    Sender's public key.

    Digital signature.

    transaction inputs and transaction outputs.

# Visualization of Transaction Data Field in Detail for Bitcoin:

| Field | Data | Size | Description |
|---|---|---|---|
| Version | 01000000 | 4 bytes | Which version of transaction data structure we're using. |
| Input Count | 01 | Variable | Indicates the upcoming number of inputs. |
| Input(s) | | | |

| Field | Data | Size | Description |
|---|---|---|---|
| TXID | 796…efc | 32 bytes | Refer to an existing transaction. |
| VOUT | 01000000 | 4 bytes | Select one of its outputs. |
| ScriptSig Size | 6a | Variable | Indicates the upcoming size of the unlocking code. |
| ScriptSig | 473…825 | | A script that unlocks the input. |
| Sequence | ffffffff | 4 bytes | |

| | | | |
|---|---|---|---|
| Output Count | 01 | Variable | Indicates the upcoming number of outputs. |
| Output(s) | | | |

| Field | Data | Size | Description |
|---|---|---|---|
| Value | 4baf210000000000 | 8 bytes | The value of the output in satoshis. |
| ScriptPubKey Size | 19 | Variable | Indicates the upcoming size of the locking code. |
| ScriptPubKey | 76a9…88ac | | A script that locks the output. |

| | | | |
|---|---|---|---|
| Locktime | 00000000 | 4 bytes | Set a minimum block height or Unix time that this transaction can be included in. |

# Main Component: 3- Asymmetric Key Cryptography

**Asymmetric-key cryptography** (also referred to as public key cryptography) uses a pair of keys: a public key and a private key that are mathematically related to each other.

The public key is made public without reducing the security of the process, but the private key must remain secret if the data is to retain its cryptographic protection. Even though there is a relationship between the two keys, the private key cannot efficiently be determined based on knowledge of the public key.

One can encrypt with a private key and then decrypt with the public key. Alternately, one can encrypt with a public key and then decrypt with a private key.

# Main Component: 3- Asymmetric Key Cryptography

Asymmetric-key cryptography enables a trust relationship between users who do not know or trust one another.

Public key provides a mechanism to verify the integrity and authenticity of transactions while at the same time allowing transactions to remain public.

- The private key is used to encrypt a transaction such that anyone with the public key can decrypt it (the transactions are '**digitally signed**'). Since the public key is freely available, encrypting the transaction with the private key proves that the signer of the transaction has access to the private key.
- Alternately, one can encrypt data with a user's public key such that only users with access to the private key can decrypt it.

# Main Component: 3- Asymmetric Key Cryptography Summary

- **The use of asymmetric-key cryptography in many blockchain networks is as following:**

Private keys are used to digitally sign transactions.

Public keys are used to derive addresses.

Public keys are used to verify signatures generated with private keys.

Asymmetric-key cryptography provides the ability to verify that the user transferring value to another user is in possession of the private key capable of signing the transaction.

# Main Component: 4- Addresses and Address Derivation

**Address:** is a short, alphanumeric string of characters derived from the blockchain network user's public key using a cryptographic hash function, along with some additional data (e.g., version number, checksums).

Most blockchain implementations make use of addresses as the "to" and "from" endpoints in a transaction.

Addresses are shorter than the public keys and are not secret.

One method to generate an address is to create a public key, apply a cryptographic hash function to it, and convert the hash to text:

**public key** → **cryptographic hash function** → **address**

# Address QR Code Example:

# Main Component: 5-Ledger

- **Ledger:** is a collection of transactions.
- The distributed ledger is a public database of congregated transactions to keep track of the exchange by all participants without supreme authority.
- In contrast, the traditional ledger database is centralized ownership on behalf of a community of users. This contributed to the growing interest in distributed ownership of ledgers rather than reliability concerns related to a centralized database.

# Main Component: 6-Block

- **block** contains a block header and block data.
  - The block header contains metadata for this block.
  - The block data contains a list of validated and authentic transactions which have been submitted to the blockchain network.
- Validity and authenticity are ensured by checking that the transaction is correctly formatted and that the providers of digital assets in each transaction have each cryptographically signed the transaction.
- The other full nodes will check the validity and authenticity of all transactions in a published block and will not accept a block if it contains invalid transactions.

# Block Structure: Block header

Many blockchain implementations utilize data fields like the following:

- **Block Header**

  The block number, also known as block height in some blockchain networks.

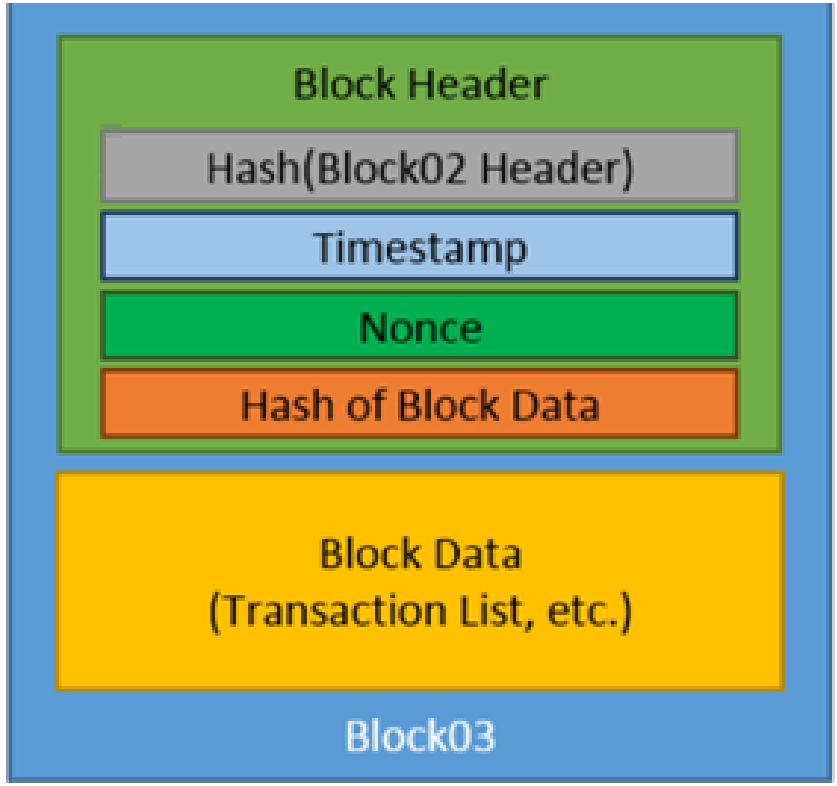
  The previous block header's hash value.

  A hash representation of the block data

  A timestamp.

  The size of the block.

  The nonce value.



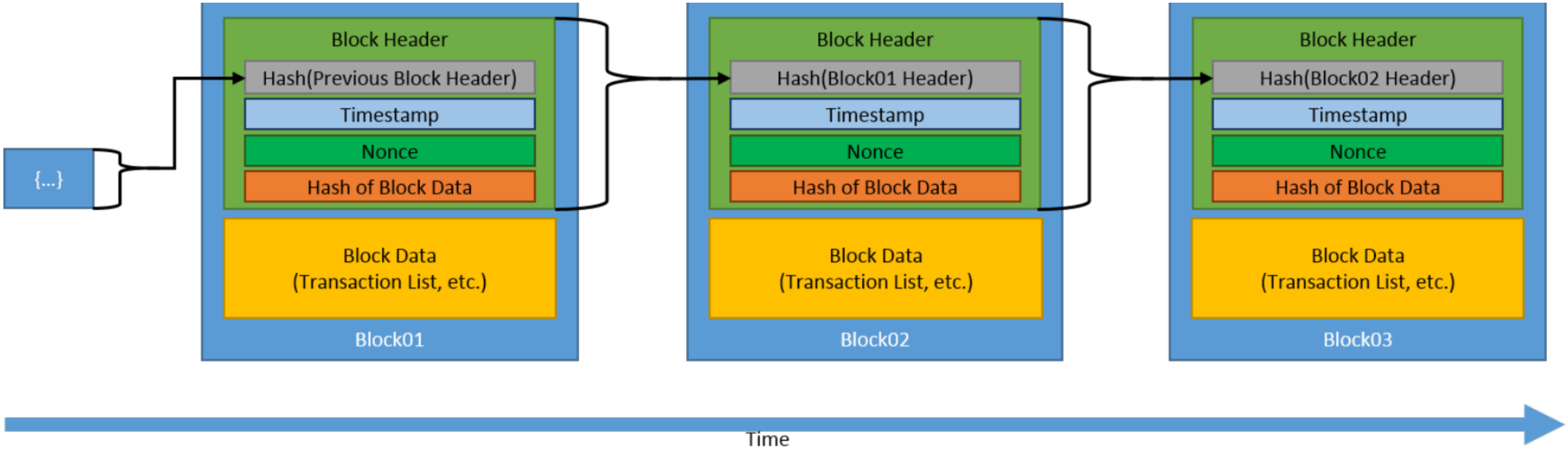
# Chaining the Blocks

Blocks are chained together through each block containing the hash digest of the previous block's header, thus forming the blockchain.

If a previously published block were changed, it would have a different hash. This in turn would cause all subsequent blocks to also have different hashes since they include the hash of the previous block.

This makes it possible to easily detect and reject altered blocks.

# Blockchain Visualisation



# Main Component: 7- Consensus

- When a user joins a blockchain network, they agree to the system's initial state.
- Every block must be added to the blockchain based on the agreed-upon consensus model.
- Block must be valid and thus can be validated independently by each blockchain network user.
- By combining the initial state and the ability to verify every block since then, users can independently agree on the current state of the blockchain.
- IF there were ever two valid chains presented to a full node, the default mechanism in most blockchain networks is that the ‘longer’ chain is viewed as the correct one and will be adopted; this is because it has had the most amount of work put into it.

# Main Component: 7- Consensus

Examples of some famous consensus models:

- Byzantine Fault Tolerance (BFT)
- Proof of Work
- Proof of Stack
- Proof of Authority
- Proof of Reputation

# Any question