इलेक्ट्रॉनिकी एवं
सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF
ELECTRONICS AND
INFORMATION TECHNOLOGY
सत्यमेव जयते

Digital India
Power To Empower

BLOCKCHAIN TECHNOLOGY

CRYPTOGRAPHIC SECURITY, HASHING AND DIGITAL SIGNATURE

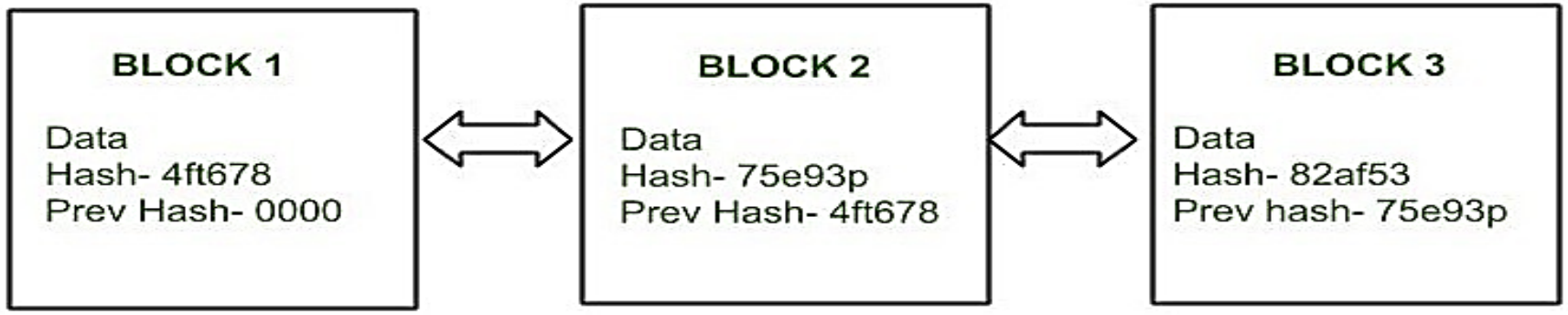
Dr. Mushtaq Ahmed
Associate Professor
Dept. of Computer Science and Engineering,
MNIT, JAIPUR

STQC
॥ गुणोत्कर्षे समृद्धि: ॥

Blockchain Technology

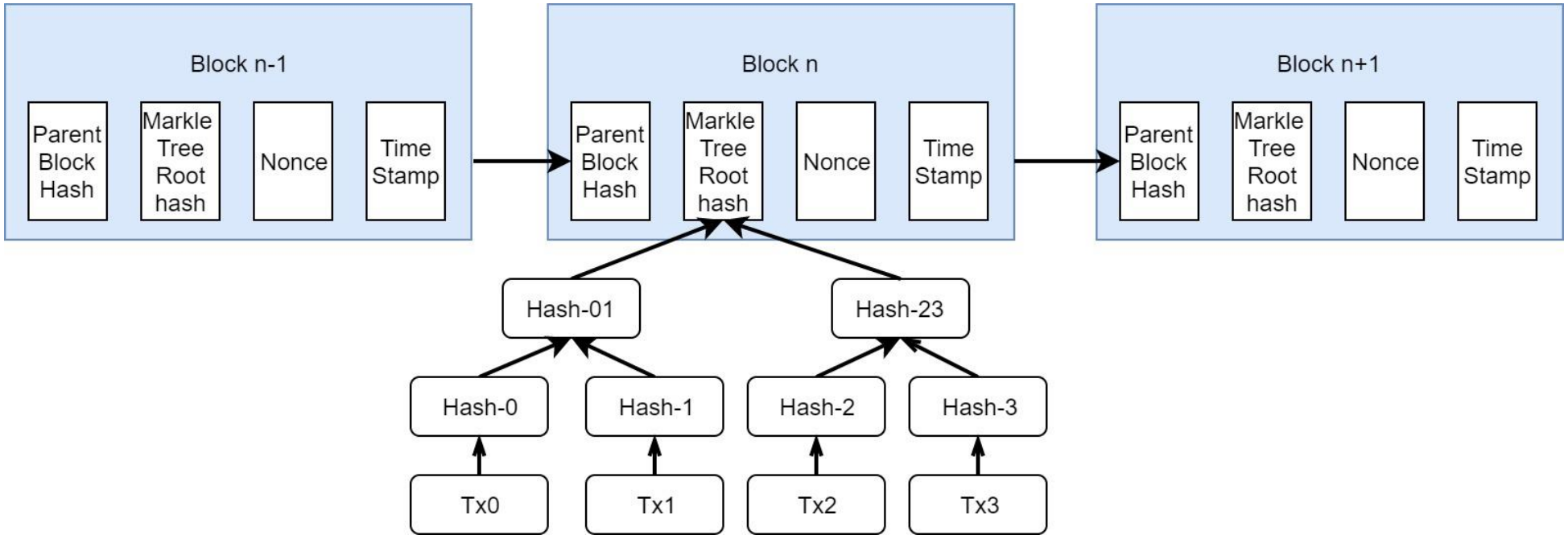
- The blockchain is a **distributed** database of records of all transactions that have been executed and shared among participating parties.
  - Each transaction is **verified** by the majority of participants of the system.
  - It contains **every single record** of each transaction.
  - **Bitcoin** is the most popular cryptocurrency and is an example of the blockchain.
  - It integrates P2P (Peer-to-Peer) protocol, digital encryption technology, consensus mechanism, smart contract and other technologies together.

- A blockchain is a chain of blocks connected to each other.



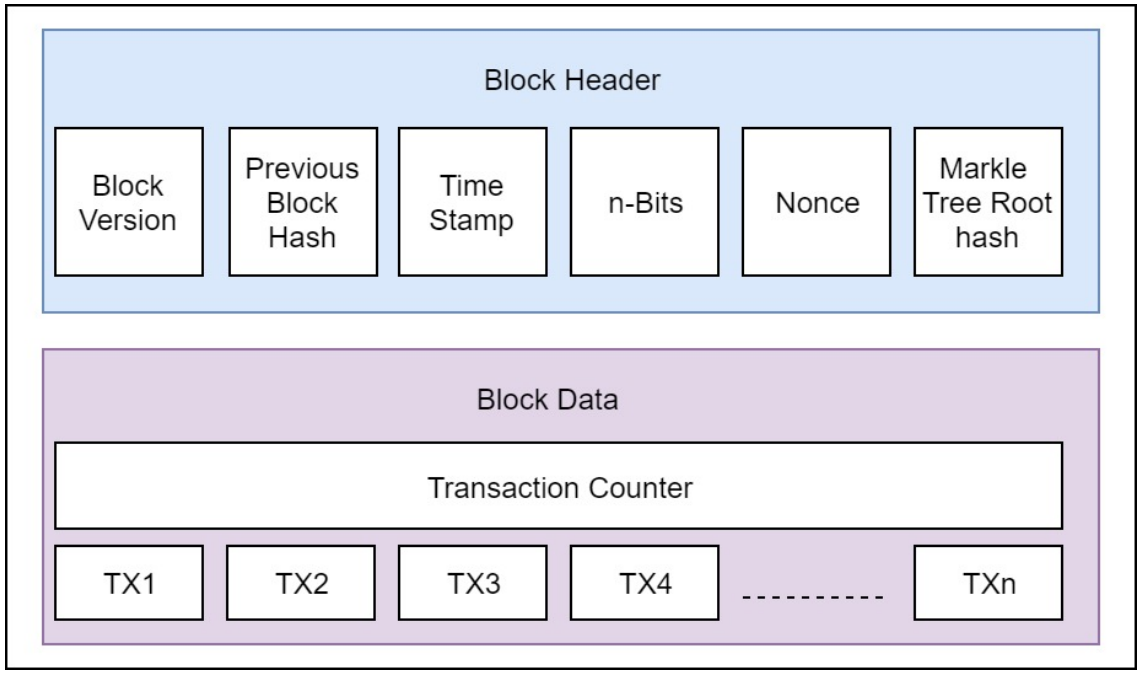
What is a block in a blockchain?

- A block in a blockchain is a digital safe to store the data and lock it forever. The data added on the block is immutable, i.e., data cannot be altered or deleted.



What is a block in a blockchain?

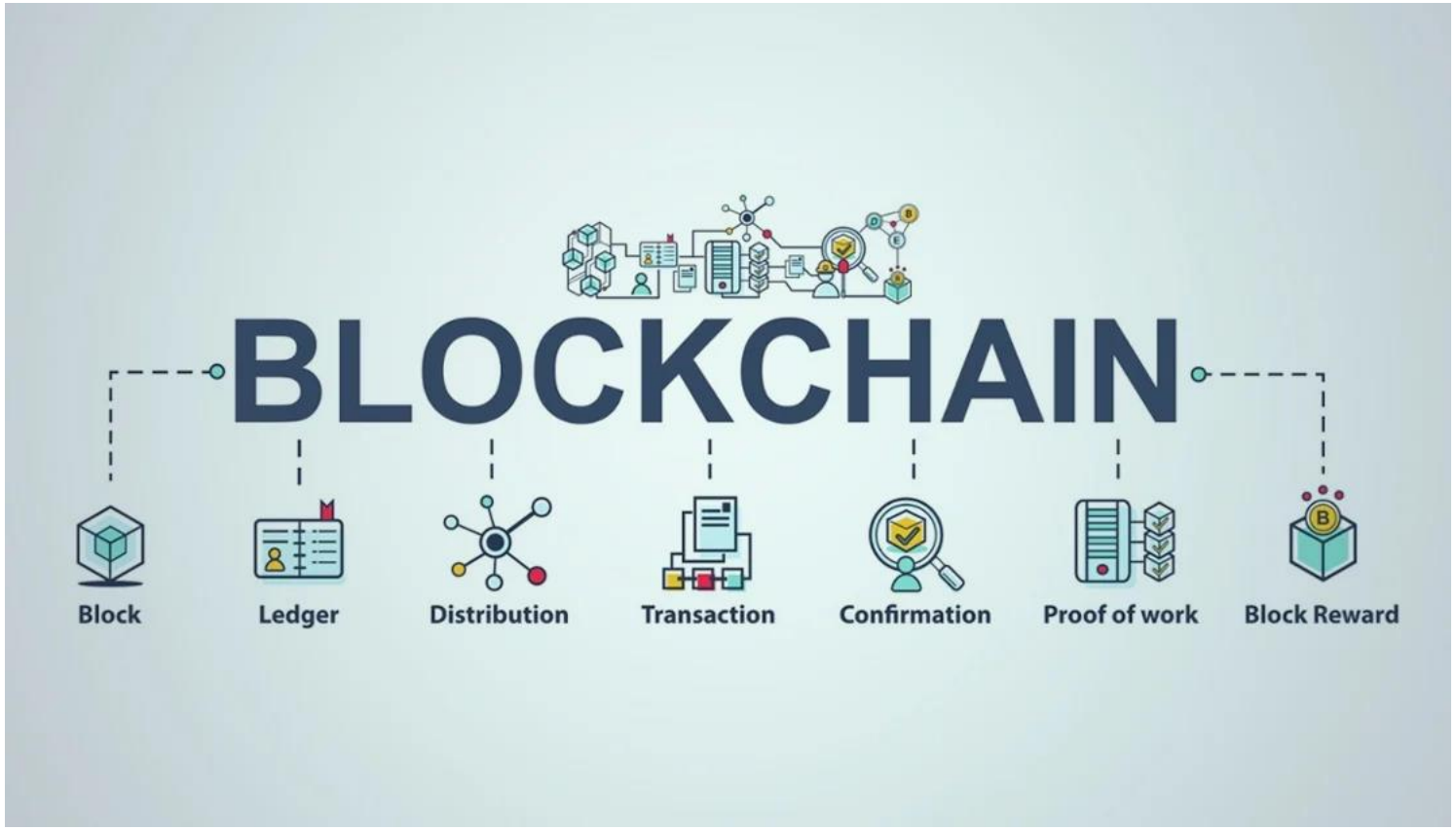
- Each block contains a **cryptographic hash** of the data of the **previous block**.
- The **nonce** is selected by the miners to solve a cryptographic puzzle for generating the next block in the chain. It is known as **proof of work**.



How does blockchain works?

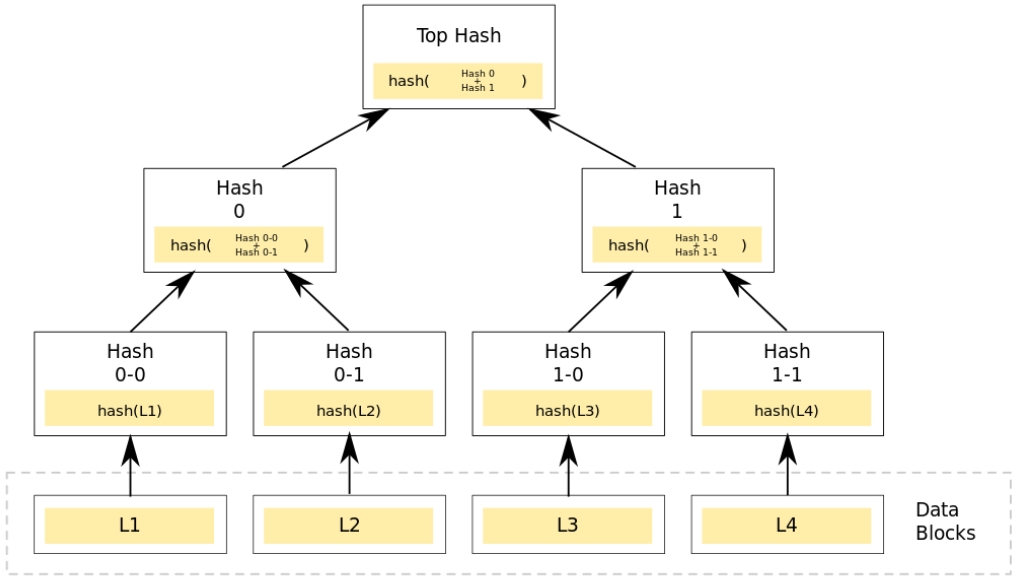
- Blocks can be recognized by their **block height(Block No.)** and block header hash. The data in the block is detected through a computerized algorithm known as a Hash function.
- It not only locks the data to be seen by the participants in the Blockchain but also makes the data immutable. Every block has its hash function.
- In Blockchain, once the data has been recorded, it will not be changed. Blockchain works like a digital notary with timestamps to avoid tampering with information

Key Components of Blockchain



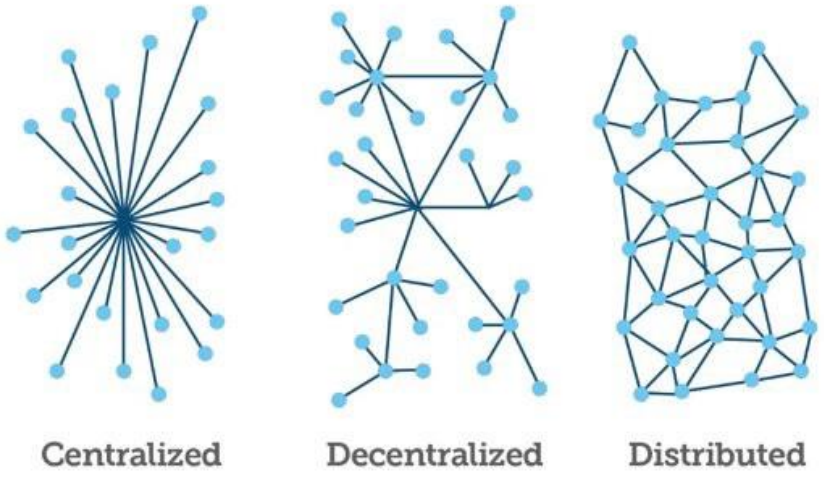
Merkle Tree

- A merkle tree is a **binary tree** with **hash pointers**.
- A Hash trees or Merkle tree is a tree in which every leaf node is labelled with the cryptographic hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.
- Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains.



Distributed Ledger Technology

- A ledger is a file which keeps the record of all the transactions taking place between the two parties on the blockchain network.
  - Keeps continuously growing
- The common types of ledgers considered by the users in the Blockchain are as follows:
  1. Centralized Ledgers
  2. Decentralized Ledgers
  3. Distributed Ledgers



Ledger Types

- **Centralized ledgers-**
  - Also known as general ledger
  - Contains all the accounts for recording transactions relating to a company's assets, liabilities, owners' equity, revenue, and expenses.
  - Every party is dependent on one central party who hold the ledger.
  - Central party has power to over the data.

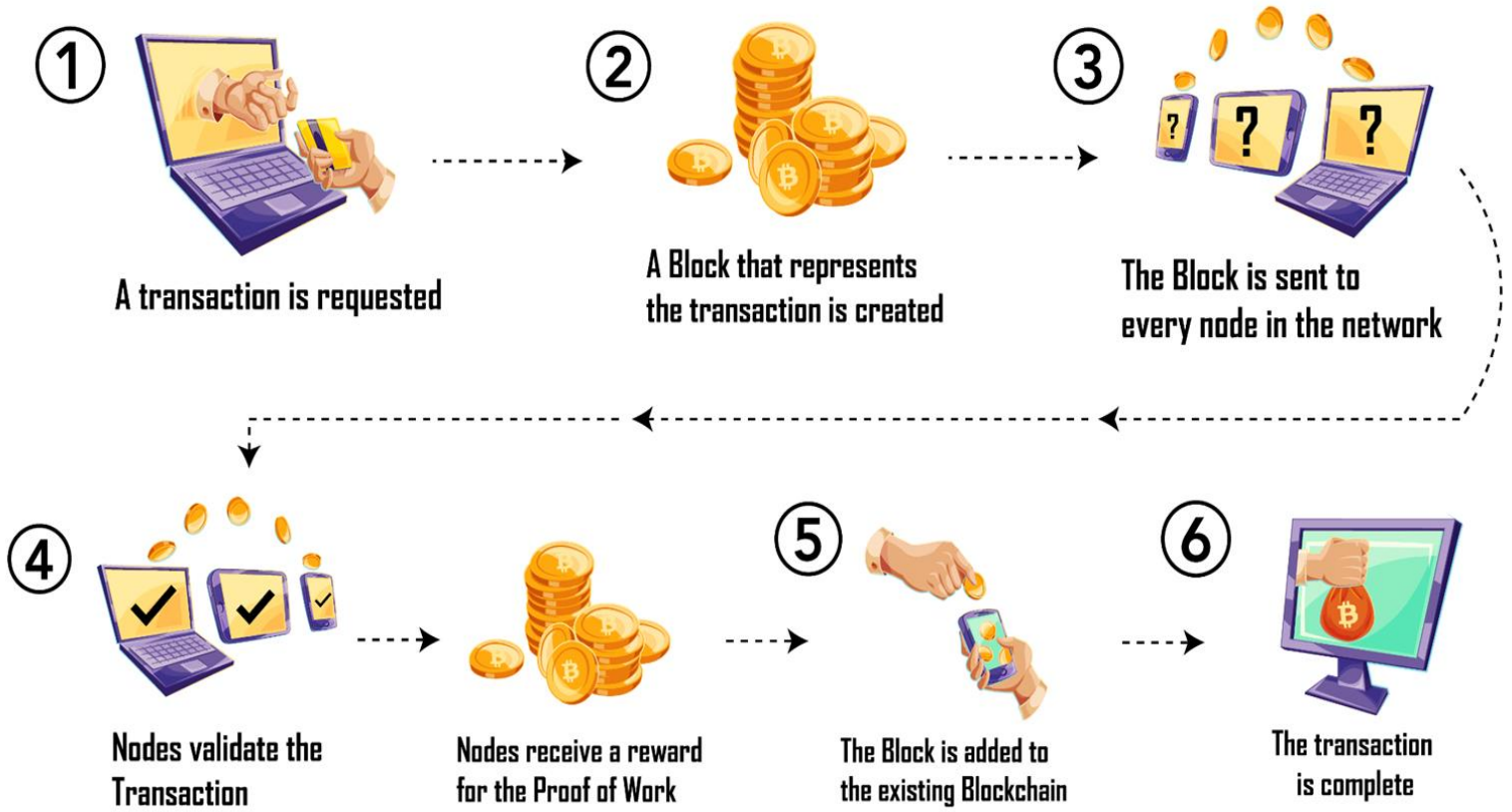
Decentralized ledgers-

- Every party holds the ledger, reducing the power of any one party over the data and providing accountability over who has altered the data and when.

Distributed Ledgers-

- Each node processes and verifies every item, thereby generating a record of each item and creating a consensus on its veracity

How transaction happens?



Blockchain platforms

Cryptography

- Cryptography is a method of developing techniques and protocols to prevent a third party from accessing and gaining knowledge of the data from the private messages during a communication process.
- There are mainly three different ways in which we can perform cryptographic algorithms, namely,
  - Symmetric-key cryptography,
  - Asymmetric-key cryptography,
  - Hash functions.

Objective of Cryptography

- **Authentication- V**erification of the identity of the sender at the receiver end.
- **Confidentiality-** Transmitted message is only received by authorize party.
- **Integrity-** Making it sure that the receiver message is same form as it was sent. Only authorized user have privileges to modify the data.

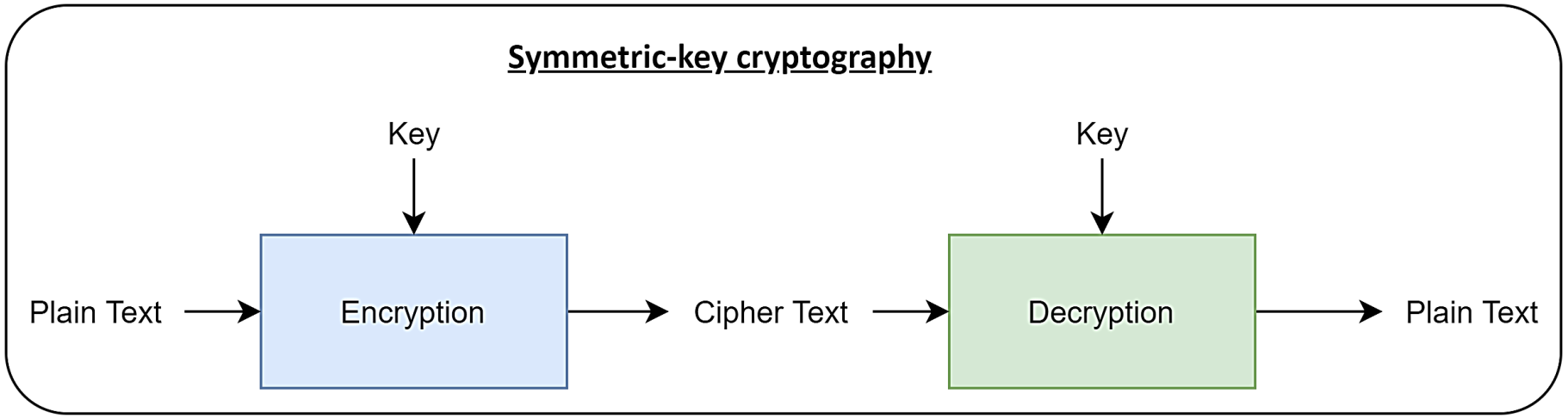
- **Access Control-** Making it sure that only authorized parties have privileges to access the given information.
- **Non Repudiation-** Method of guaranteeing message transmission between parties via digital signature or encryption. It helps to protect against the denial of authentication attempt.

Terminology used in cryptography

- **Plaintext -** Simple readable text before being encrypted into ciphertext.
- **Ciphertext**- Transformation of original message into non-readable message before transmission.
- **Encryption-** It is a process of plaintext (normal text) to a ciphertext (random sequence of bits).
- **Decryption-** The inverse process of encryption, conversion of ciphertext to plaintext.
- **Key-** Key is numeric or alphanumeric text used for the encryption of plaintext and decryption of the ciphertext.
- **Key-pair**
  - **Private key** The private key is one which is accessible only to the signer. It is used to generate the digital signature which is then attached to the message.
  - **Public key** The public key is made available to all those who receive the signed messages from the sender. It is used for verification of the received message.

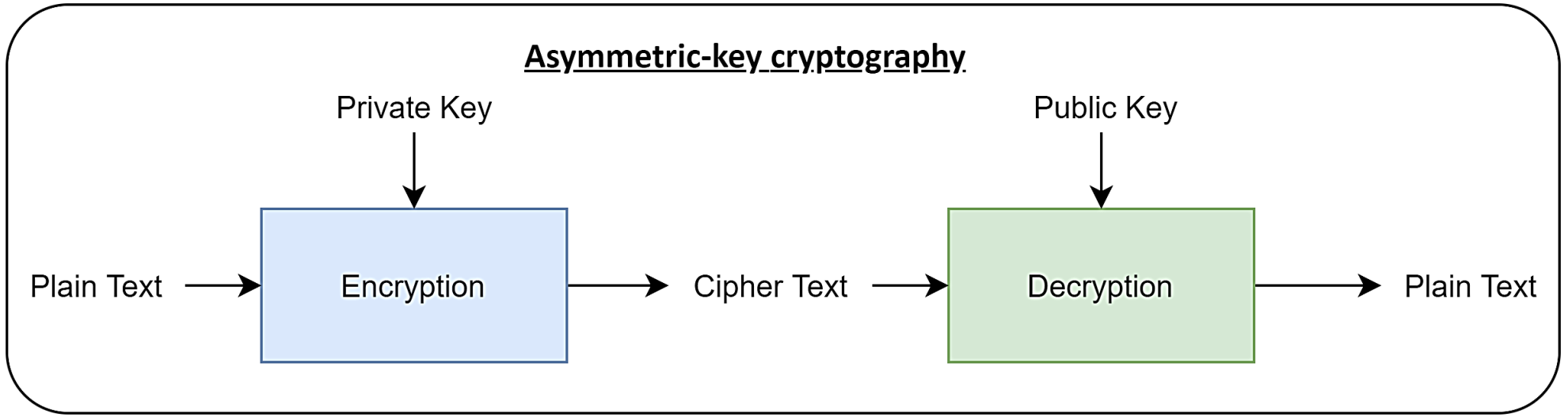
Symmetric-Key Cryptography

- In this encryption method, we take a single key into application.
- This common key is used for both the encryption as well as the decryption process.
- Using a common single key creates a problem of securely transferring the key between the sender and the receiver.
- It is also called Secret-Key Cryptography.



Asymmetric-Key Cryptography

- This encryption method uses a pair of keys, an encryption key, and a decryption key, named **public key** and **private key** respectively.
- The **key pair** generated by this algorithm consists of a private key and a unique public key that is generated using the same algorithm.
- It is also called **Public-Key** Cryptography.



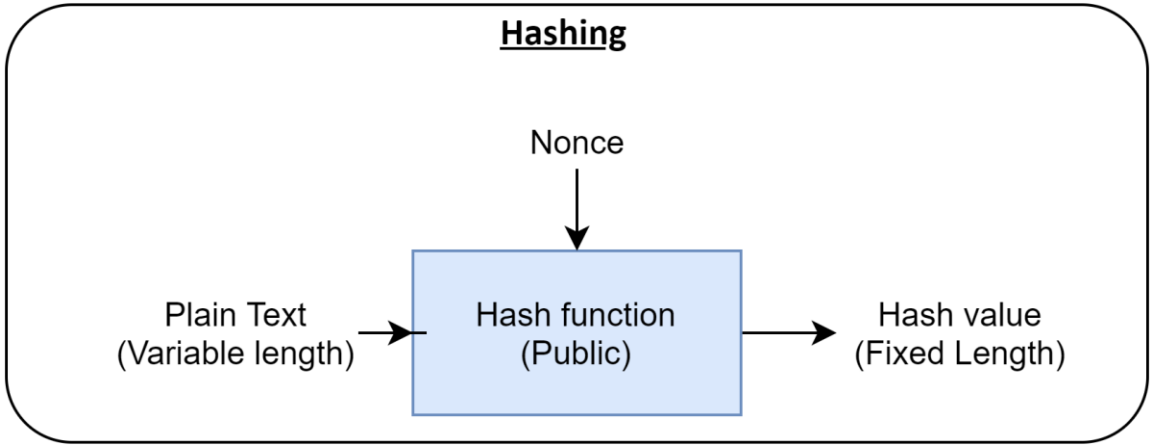
Hashing

A **hash function** takes some input data and calculates a **hash value**, (can be anything). The process of doing this is called **hashing**.

The hash value is always the same size, no matter what the input looks like.

Hashing is usually a one-way calculation.

- Can't normally get the original data back from the hash value, because there are many more possible input data combinations than there are possible hash values. Computationally infeasible to find data mapping to specific hash (one-way property) and Computationally infeasible to find two data to same hash (collision-free property).



Hashing

The main advantages are:

- The hash value is always the same size
- The same input will always generate the same output.
- If it's a good hash function, different inputs will *usually* generate different outputs, but it's still possible that two different inputs generate the same output (this is called a **hash collision**).

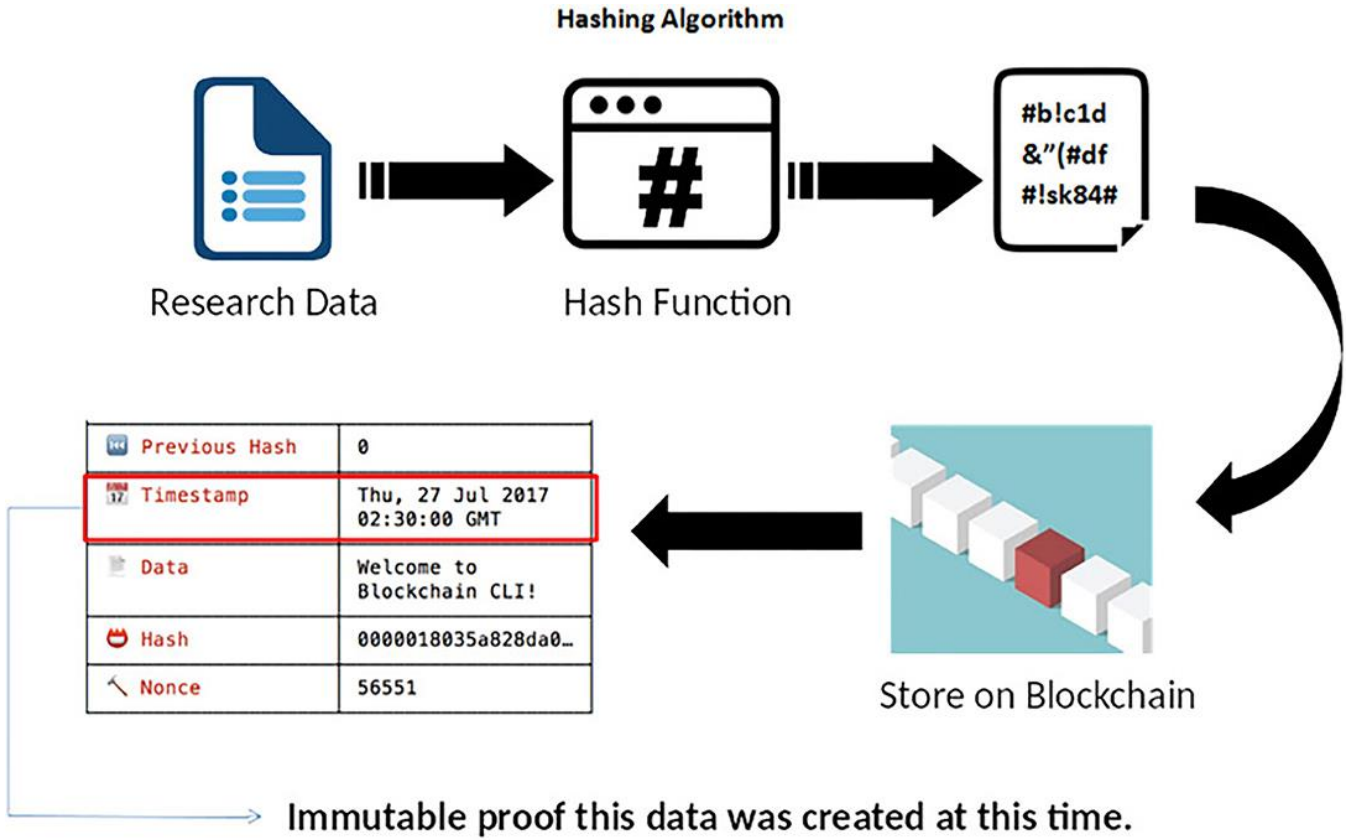
If you have a **cryptographical hash function** you also get one more advantage:

- From having only the hash value, it's impossible (unfeasible) to come up with input data that would hash to this value. Never mind that it's not the original input data, *any* kind of input data that would hash to the given output value is impossible to find in a useful timeframe.

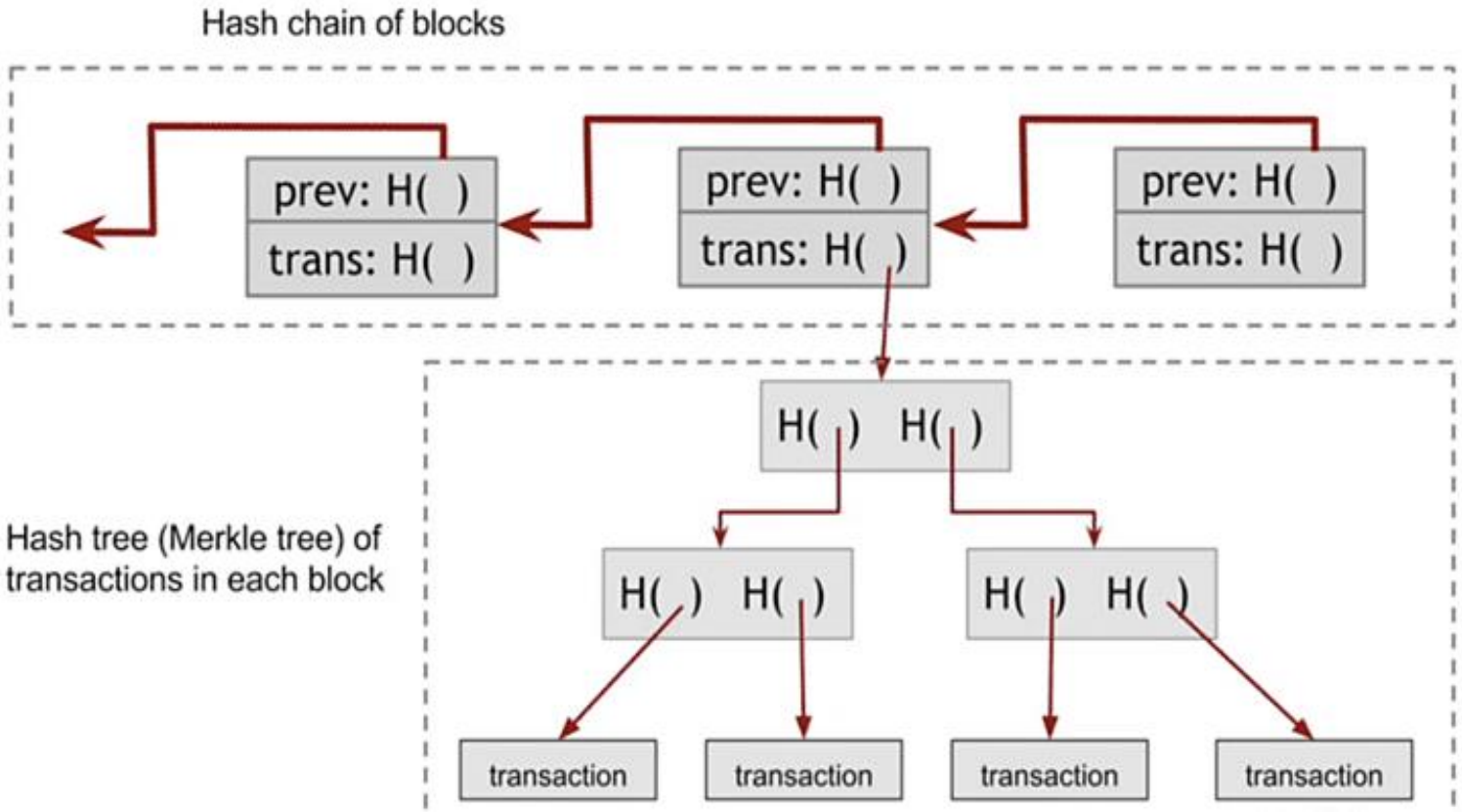
Hash Function Uses

- Message Integrity Check (MIC)
  - send hash of message (digest)
  - MIC always encrypted, message optionally
- Message Authentication Code (MAC)
  - send keyed hash of message
  - MAC, message optionally encrypted
- Digital Signature (non-repudiation)
  - Encrypt hash with private (signing) key
  - Verify with public (verification) key

Hashing in Blockchain



Use of Hashing in Blockchain



Digital Signature

- A digital signature is a mathematical scheme that is used to authenticate the sender of an electronic document.
- A digital signature is nothing but an attachment to any piece of electronic information, which represents the content of the document and the identity of the owner of that document uniquely.

MOTIVATION

- To provide Authenticity, Integrity and Non-repudiation to electronic documents.
- To use the internet as the safe and secure medium for e-Commerce and e-Governance.

Cryptography in Blockchain

- Public-key encryption serves as the basis for blockchain wallets and transactions. Each transaction is signed with a private key and then can be further verified with a public key.
- Cryptographic hash functions provide the immutability and Merkle trees organize transactions while enabling blockchains to be more efficient.
- Digital signatures are primarily used to verify the authenticity of transactions.

- Hashing, Public-Private key pair, and the Digital Signature together constitute the foundation for the blockchain.
- These cryptographic features make possible the reliability and immutability of the data stored on the blockchain.

Variants of Hashing

- MD4 and MD5 by Ron Rivest (1990,1994)
- SHA-0, SHA-1 by NSA (1993, 1995)
- RIPEMD-160 (1996)
- SHA-2 (2002 – 224, 256, 385, 512)
- Whirlpool
- Tiger
- GOST-3411
- SHA-3

Secure Hash Algorithm

SHA originally designed by NIST & NSA in 1993

a revised version was issued as FIPS 180-1 in 1995 and referred as SHA-1

- The algorithm is SHA, the standard is SHS (Secure Hash Standard)

SHA is based on design of MD4 with key differences.

SHA-1 produces 160-bit hash values.

In 2005, some security concern are raised on security of SHA-1.

Revised Secure Hash Algorithm

In 2002, NIST issued a revised SHA standard, as FIPS 180-2,

- That defined three new versions of SHA, known as SHA-256, SHA-384, and SHA-512. Collectively,
- These hash algorithms are known as SHA-2.

In 2008, Again a revised document was issued as FIP PUB 180-3

- Added a 224-bit version.

In 2005, NIST announced the intention to phase out approval of SHA-1 and move to a reliance on the other SHA versions by 2010.

SHA Versions comparison

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

SHA in Blockchain

A Bitcoin's blockchain uses **SHA-256**.

Part of the SHA 2 family.

Goal is to compute a unique hash value for any input data or message

No matter the size of the input, the output is the fixed size message digest

Since then, SHA-256 has been adopted by a number of different blockchain projects, including several coins created from forks of the original Bitcoin source code.

SHA-3

SHA-1 not yet "broken"(not less than brute-force time)

- but similar to broken MD5 & SHA-0
- so considered as less secure

SHA-2 (esp. SHA-512) seems secure

- shares same structure and mathematical operations as predecessors so have concern

In 2012, NIST released Keccak also known as SHA-3.

- It is the latest generation secure hashing algorithm released by NIST.
- Keccak is a family of cryptographic sponge functions and is designed as an alternative to SHA-256
- An algorithm used by Bitcoin and various other crypto currencies.
- Compared to SHA-256, Keccak (SHA-3) is much faster and is more secure.

SHA-256, is **a hashing algorithm used to convert text of any length into a fixed-size string of 256 bits (32 bytes)**.

Orange

4f 72 61 6e 67 65
12-bit Hex Input

SHA-256

78e7771b8b46e11ddb34ba48887e1330525215f96d94778980d1186e6f09f6

Fixed 256-bit Value Digest (Output Hash Value)

The quick brown fox jumps over the lazy dog

88-bit Hex Input Message

SHA-256

c03905fcdab297513a620ec81ed46ca44ddb62d41cbbd83eb4a5a3592be26a

Fixed 256-bit Value Digest (Output Hash Value)

Image File

SHA-256

017fa150d2b079a9fee41169cb09142f2b91badbbbf311797e90982b8d

Fixed 256-bit Value Digest (Output Hash Value)

SHA256 Properties

Cryptographic hashing function needs to have certain properties in order to be completely secured. These are :

- Compression
- Avalanche Effect
- Determinism
- Pre-Image Resistant (One Way Function)
- Collision Resistance
- Efficient (Quick Computation)

Compression

- **Output hash should be a fixed number of characters, regardless of the size of the input message !**

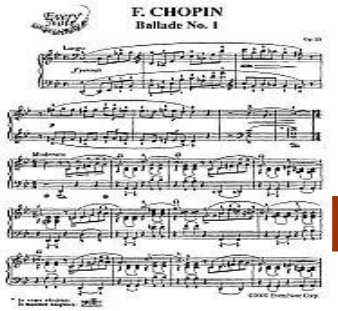
a

2-bit Hexadecimal Value

SHA-256

a7da489976d0047490617adb4f7a1f27f7af8b52a5176fd002ffe471863520ab

Fixed 256-bit Value Digest (Output Hash Value)



5 MB MP3 File

SHA-256

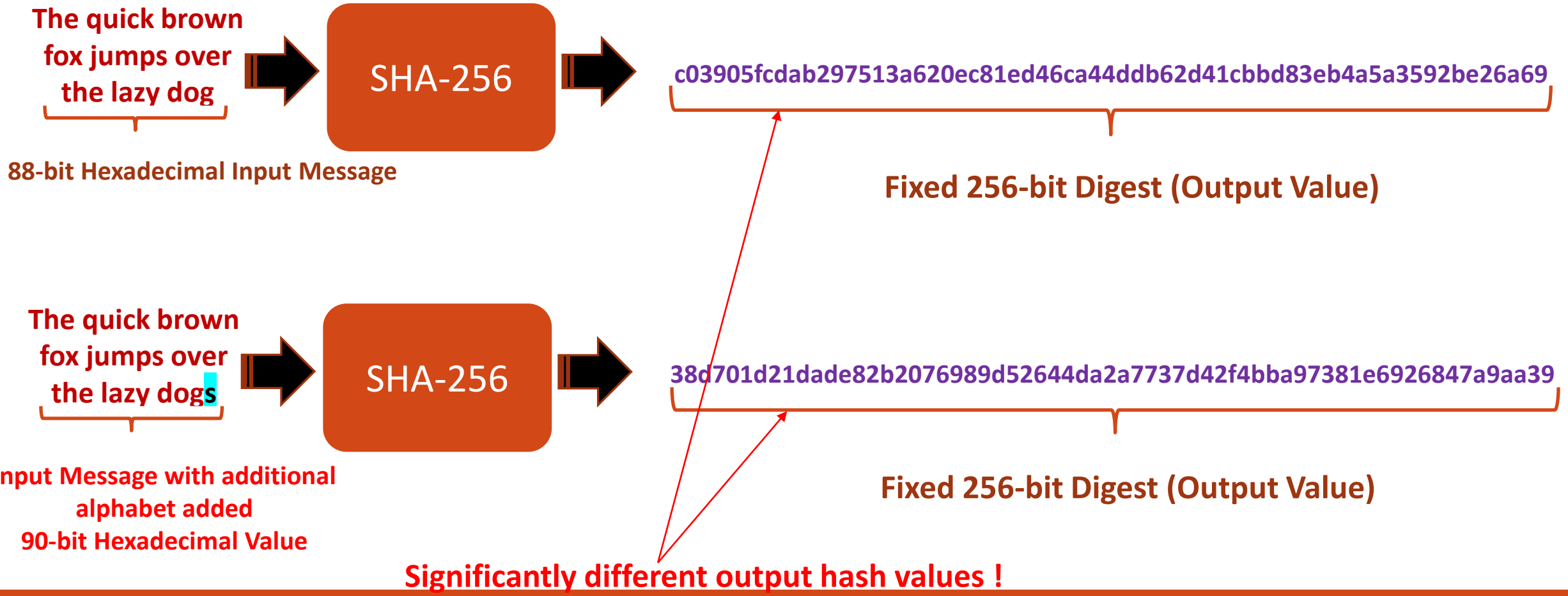
d9904266f642d4659769bba49eb8552e2d53ef09b1faabd8f72154530ba16fc8

Fixed 256-bit Value Digest (Output Hash Value)

For input message of 2-bit value or 5 MB MP3 File, output hash value is fixed size of 256-bit value

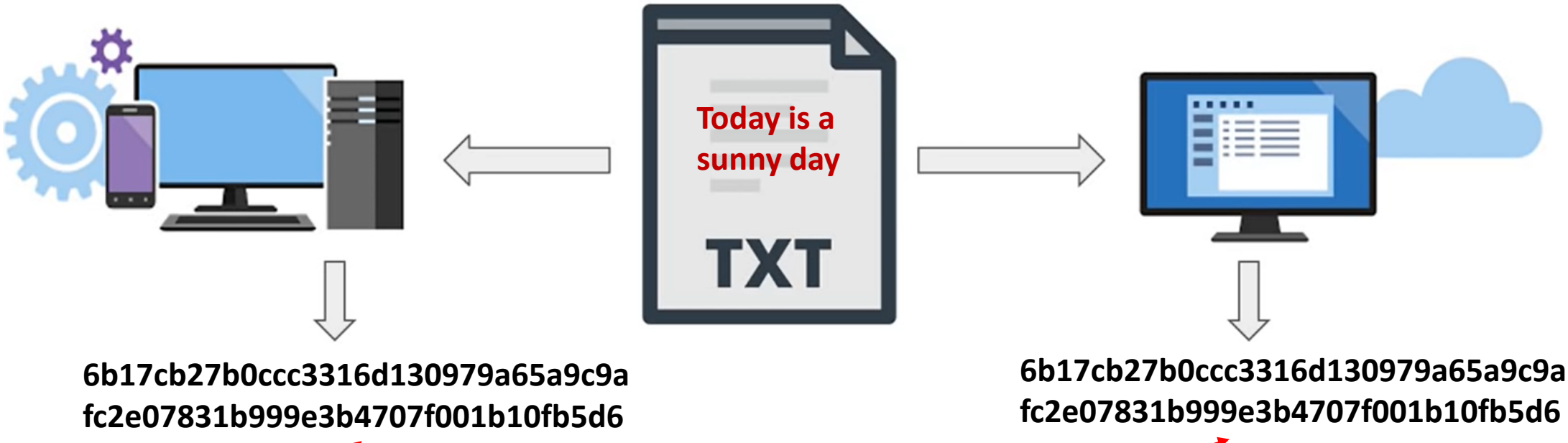
Avalanche Effect

- **A minimal change in the input change the output hash value dramatically !**
  - This is helpful to prevent hacker to predict output hash value by trial and error method



Determinism

- **Same input must always generate the same output by different systems**
  - Any machine in the world which understands hashing algorithm should able generate same output hash value for a same input message



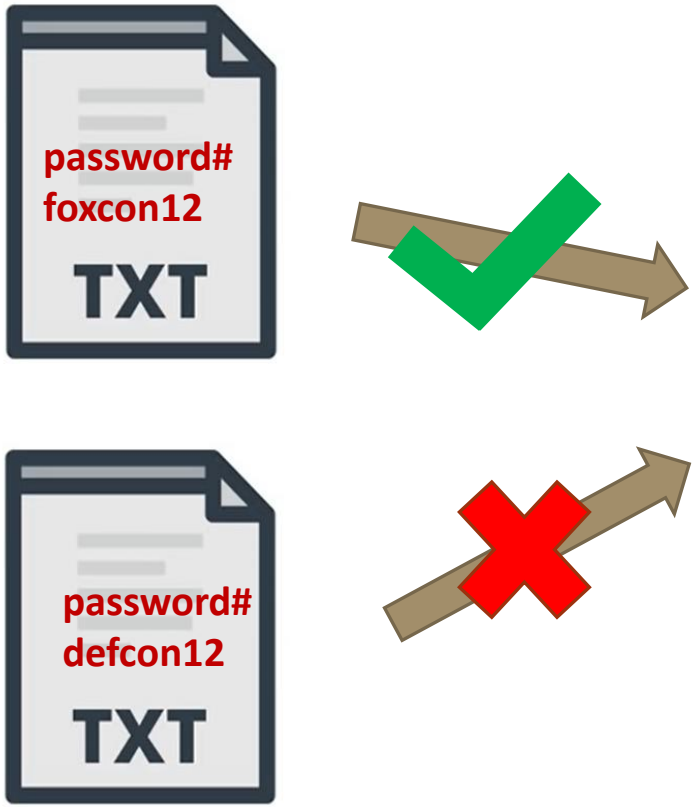
Pre-Image Resistant (One-Way) And Efficient

- **Secure hashing algorithm should be a One-Way function**
  - There should be no way to reverse the hashing process to retrieve the original input message !
  - If input message can be retrieved from output hash value then the whole concept will fail !

- **Efficient : Creating the output hash should be a fast process that doesn't make heavy use of computing power**
  - Should not need supercomputers or high end machines to generate hash !
  - More feasible for usage !

Collision Resistance

- **Practically impossible to find two different inputs that produce the same output**
  - Since input can be large combination values and output is smaller fixed value, it is mathematically possible to find two input messages having same output hash value
  - It must withstand collision !



db2c26da2750dea1add7d7677c22d6dc
b6dc4e2674357c82c39bb96d563f0578

Two different input blocks with same output hash value should be practically impossible even though mathematically possible !

Applications of SHA256 : Verifying File Integrity

Software manufacturer wants to ensure that the executable file is received by users without modification

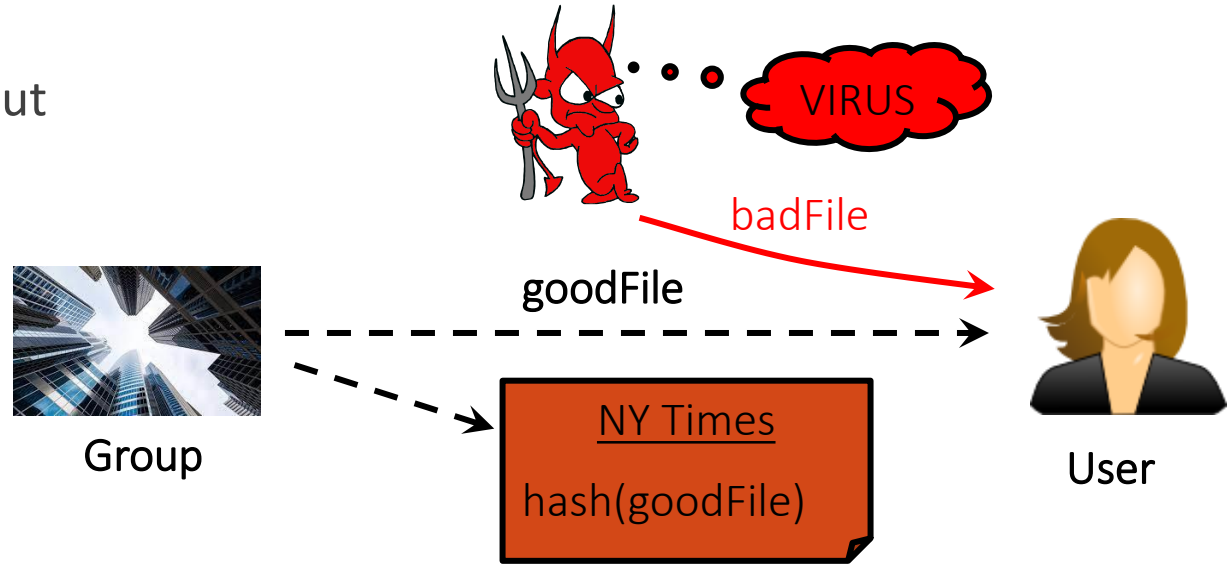
Sends out the file to users and publishes its hash in NY Times

The goal is integrity, not secrecy

Idea: given goodFile and hash(goodFile),

very hard to find badFile

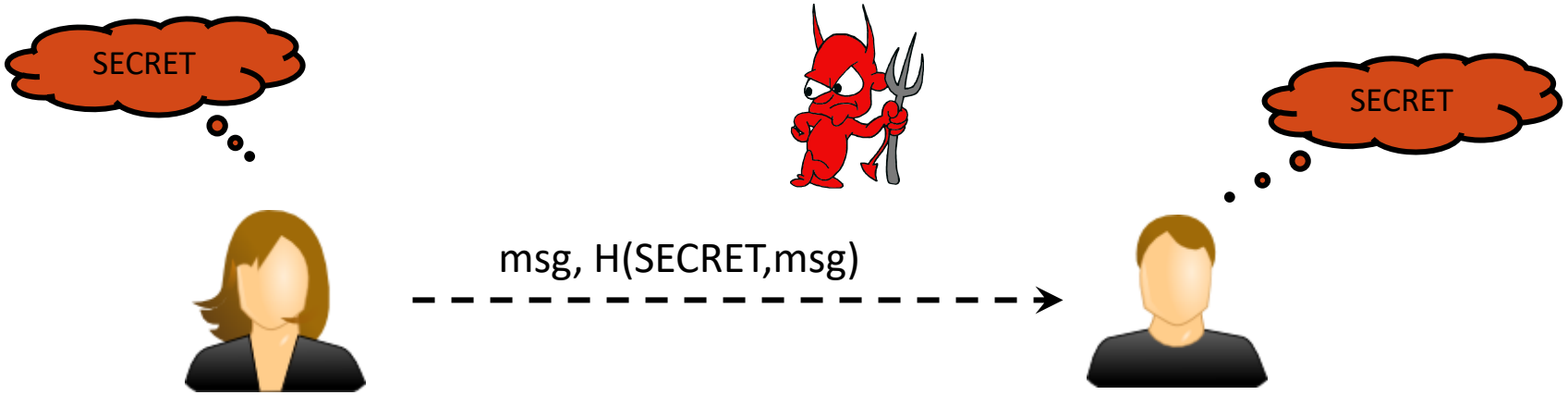
such that hash(goodFile)=hash(badFile)



Applications of SHA256 : Authentication

- **Alice wants to ensure that nobody modifies message in transit(both integrity and authentication)**

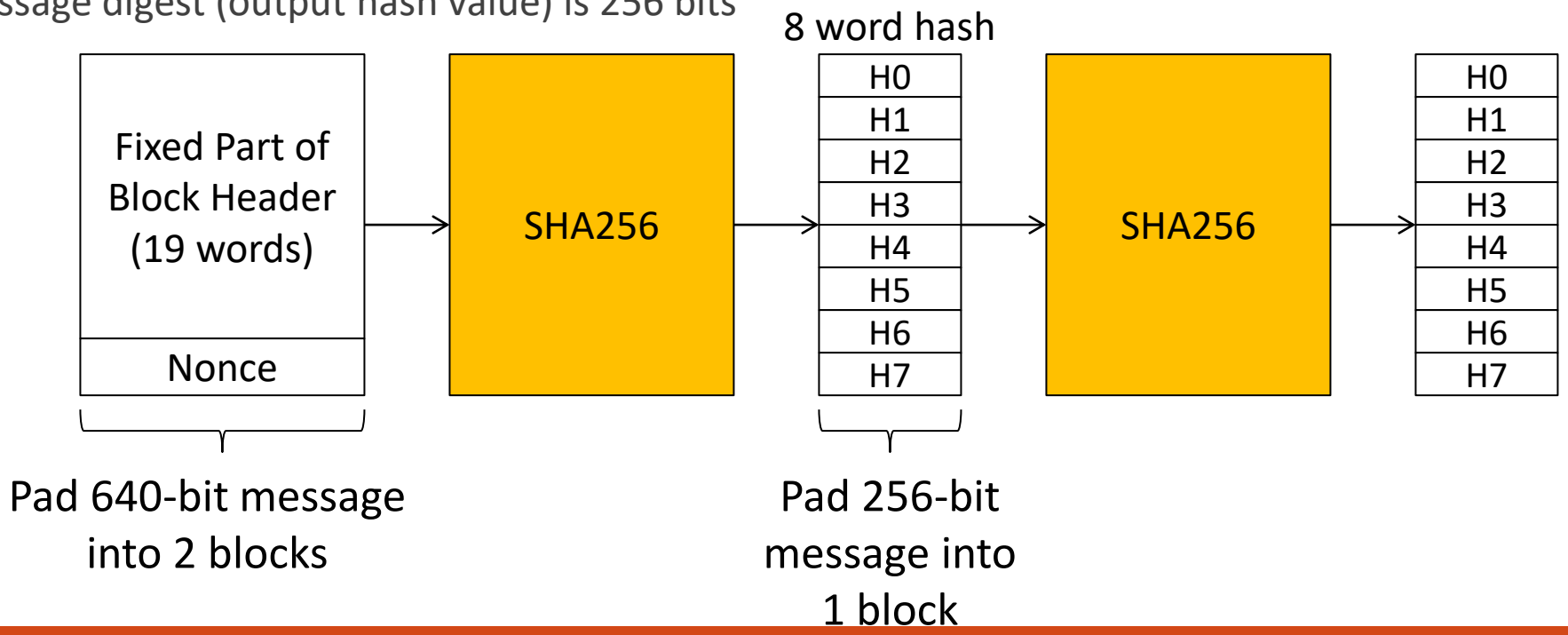
- **Idea: given msg,**
  very hard to compute H(SECRET, msg) without SECRET;
  easy with SECRET



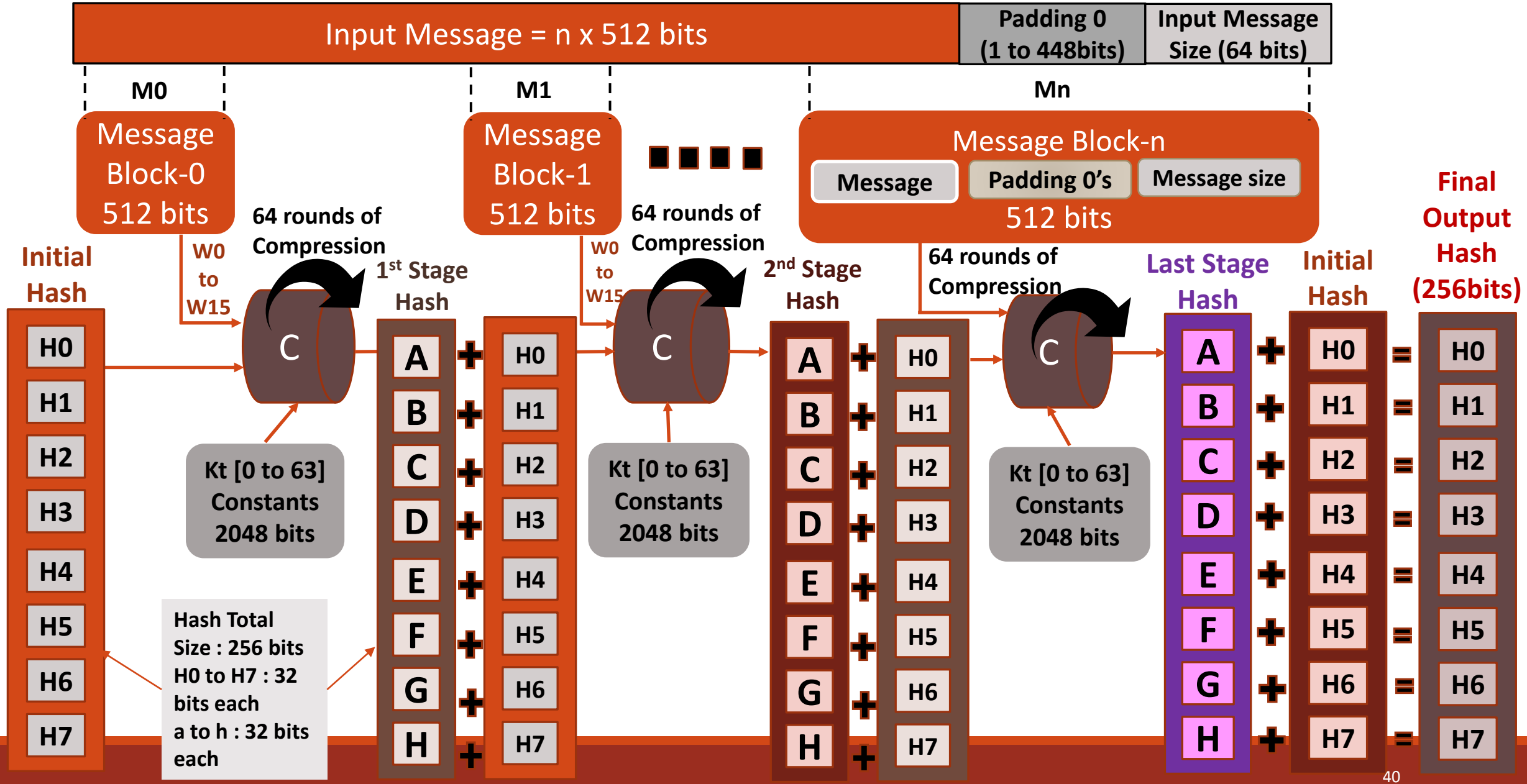
SHA256 Algorithm

General Assumptions

- Input message must be <=2^{64} bits
- Message is processed in 512-bit blocks sequentially
- Message digest (output hash value) is 256 bits



SHA256 Algorithm



SHA256 Algorithm – Compression Function

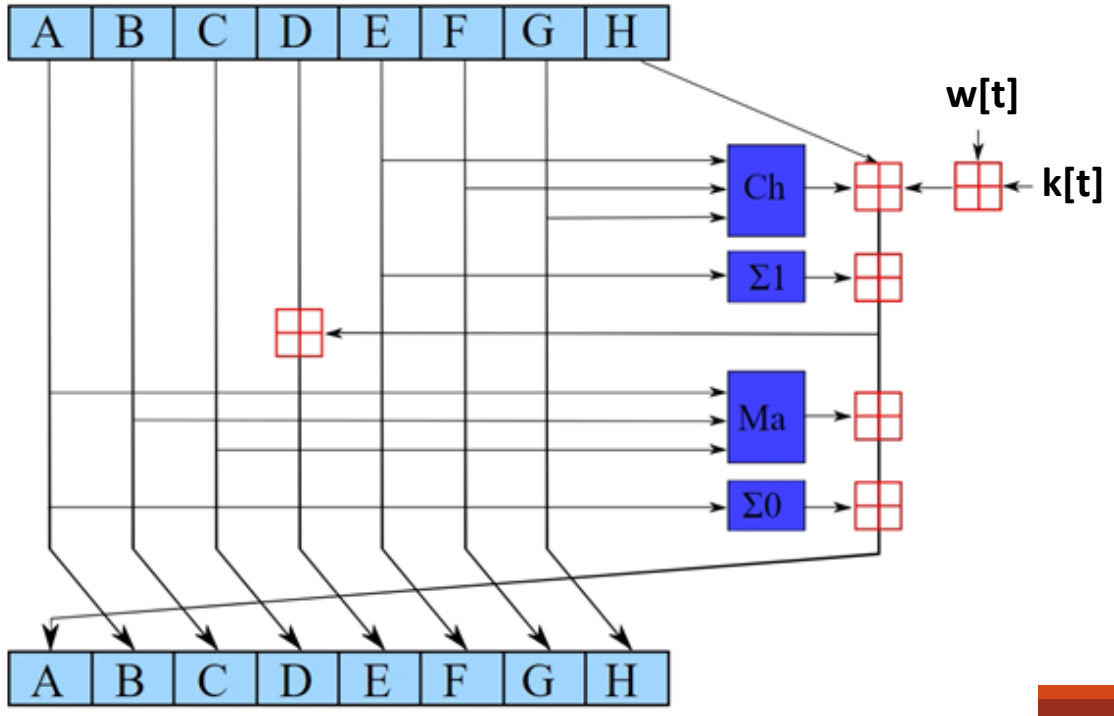
C

Step 1: Word Expansion

```
for (t = 0; t < 64; t++) begin
    if (t < 16) begin
        w[t] = dpsram_tb[t]; // Get Input Message 512-bit block and store in Wt array
    end else begin
        s0 = rightrotate(w[t-15], 7) ^ rightrotate(w[t-15], 18) ^ (w[t-15] >> 3);
        s1 = rightrotate(w[t-2], 17) ^ rightrotate(w[t-2], 19) ^ (w[t-2] >> 10);
        w[t] = w[t-16] + s0 + w[t-7] + s1;
    end
end
```

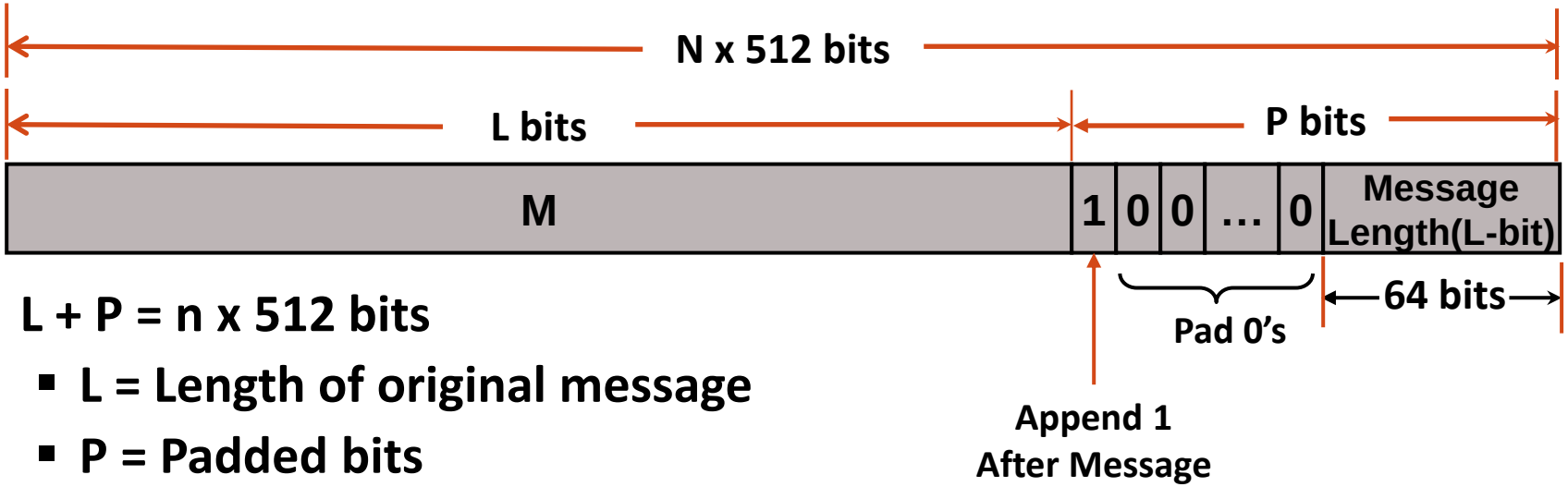
Step 2: SHA256 Operation

Performed 64 times t = 0 to 63



SHA256 Algorithm

- **Step 1:** Append padding bits (1 and 0's)
  - A **L**-bit message **M** is padded in the following manner:
    - Add a single "1" to the end of **M**
    - Then pad message with "0's" until the length of message is congruent to 448, modulo 512 (which means pad with 0's until message is 64-bits less than some multiple of 512).
- **Step 2** : Append message length bits in 0 to 63 bit position
  - Since SHA256 supports until 2^64 input message size, 64 bits are required to append message length

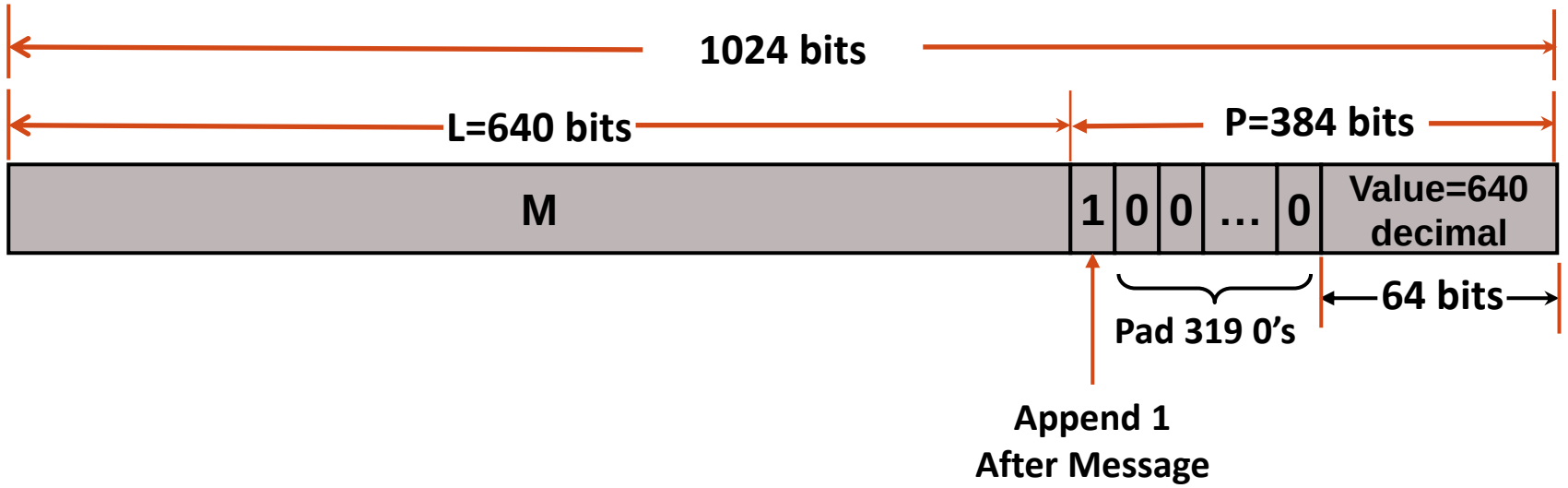


L + P = n x 512 bits

- **L = Length of original message**
- **P = Padded bits**

SHA256 Algorithm

- **Example :** Lets say, original Message is **L** = 640 bits
  - Since message blocks have minimum 512 chunks, to fit original message of 640 bits in 512 bits chunks, it would require 2 message blocks **(n = 2)**
    - **M0** (first block) Size = 512 bits (no padding required)
    - **M1** (second block) Size = 512 bits after padding
      - **512 bits** = 128 bits of original message + 1 bit for appending '1' + 319 bits of 0's + 64 bit message length
    - **Message length**=decimal value 640 stored in 0 to 63 bits



SHA256 Algorithm

- **Step 3 :** Buffer Initialization
  - Initialize message digest (MD) buffers / output hash to these 8 32-bit words

H0 = 6a09e667
H1 = bb67ae85
H2 = 3c6ef372
H3 = a54ff53a
H4 = 510e527f
H5 = 9b05688c
H6 = 1f83d9ab
H7 = 5be0cd19

SHA256 Algorithm

- **Step 4 :** Processing of the message (algorithm)
  - Divide message M into 512-bit blocks, M_0, M_1, … M_j, …
  - Process each M_j sequentially, one after the other
  - Input:
    - W_t: a 32-bit word from the message
    - K_t : a constant array
    - H0, H1, H2, H3, H4, H5, H6, H7 : current MD (Message Digest)
  - Output:
    - H0, H1, H2, H3, H4, H5, H6, H7 : new MD (Message Digest)

SHA256 Algorithm

- **Step 4 :** Cont'd
  - At the beginning of processing each M_j, initialize (A, B, C, D, E, F, G, H) = (H0, H1, H2, H3, H4, H5, H6, H7)
  - Then 64 processing rounds of 512-bit blocks
  - Each step t ($0 \le t \le 63$): Word expansion for W_t
    - If $t < 16$
      - W_t = t^{th} 32-bit word of block M_j
    - If $16 \le t \le 63$
      - $s_0 = (W_{t-15}$ **rightrotate** 7) **xor** (W_{t-15} **rightrotate** 18) **xor** (W_{t-15} **rightshift** 3)
      - $s_1 = (W_{t-2}$ **rightrotate** 17) **xor** (W_{t-2} **rightrotate** 19) **xor** (W_{t-2} **rightshift** 10)
      - $W_t = W_{t-16} + s_0 + W_{t-7} + s_1$

SHA256 Algorithm

- **Step 4:** Cont'd
  - K_t constants

    K [0..63] = 0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,
    0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5, 0xd807aa98,
    0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe,
    0x9bdc06a7, 0xc19bf174, 0xe49b69c1, 0xefbe4786, 0x0fc19dc6,
    0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
    0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3,
    0xd5a79147, 0x06ca6351, 0x14292967, 0x27b70a85, 0x2e1b2138,
    0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e,
    0x92722c85, 0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
    0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070, 0x19a4c116,
    0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a,
    0x5b9cca4f, 0x682e6ff3, 0x748f82ee, 0x78a5636f, 0x84c87814,
    0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2

SHA256 Algorithm

- **Step 4 :** Cont'd
  - Each step t ($0 \le t \le 63$):

Σ0 S_0 = (A **rightrotate** 2) **xor** (A **rightrotate** 13) **xor** (A **rightrotate** 22)

Ma maj = (A **and** B) **xor** (A **and** C) **xor** (B **and** C)

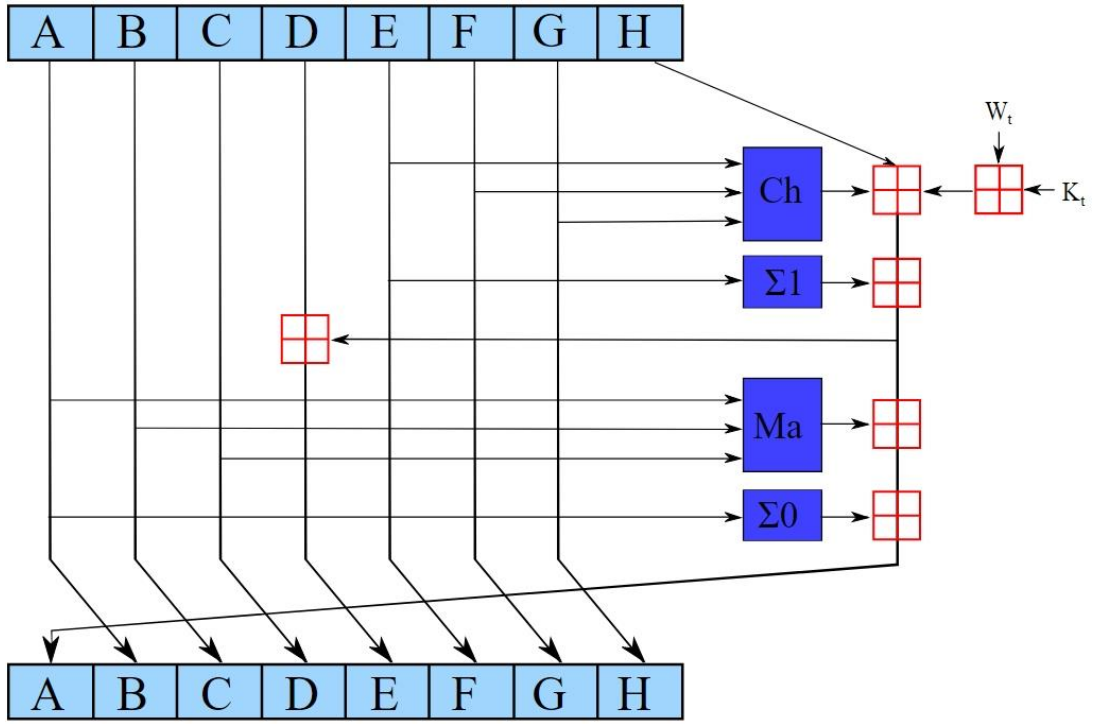
$t_2 = S_0 + maj$

Σ1 S_1 = (E **rightrotate** 6) **xor** (E **rightrotate** 11) **xor** (E **rightrotate** 25)

Ch ch = (E **and** F) **xor** ((**not** E) **and** G)

$t_1 = H + S_1 + ch + K[t] + W[t]$

$(A, B, C, D, E, F, G, H) = (t_1 + t_2, A, B, C, D + t_1, E, F, G)$



SHA256 Algorithm

- **Step 4 :** Cont'd
  - Finally, when all 64 steps have been processed, set

    H0 = H0 + a H1 = H1 + b

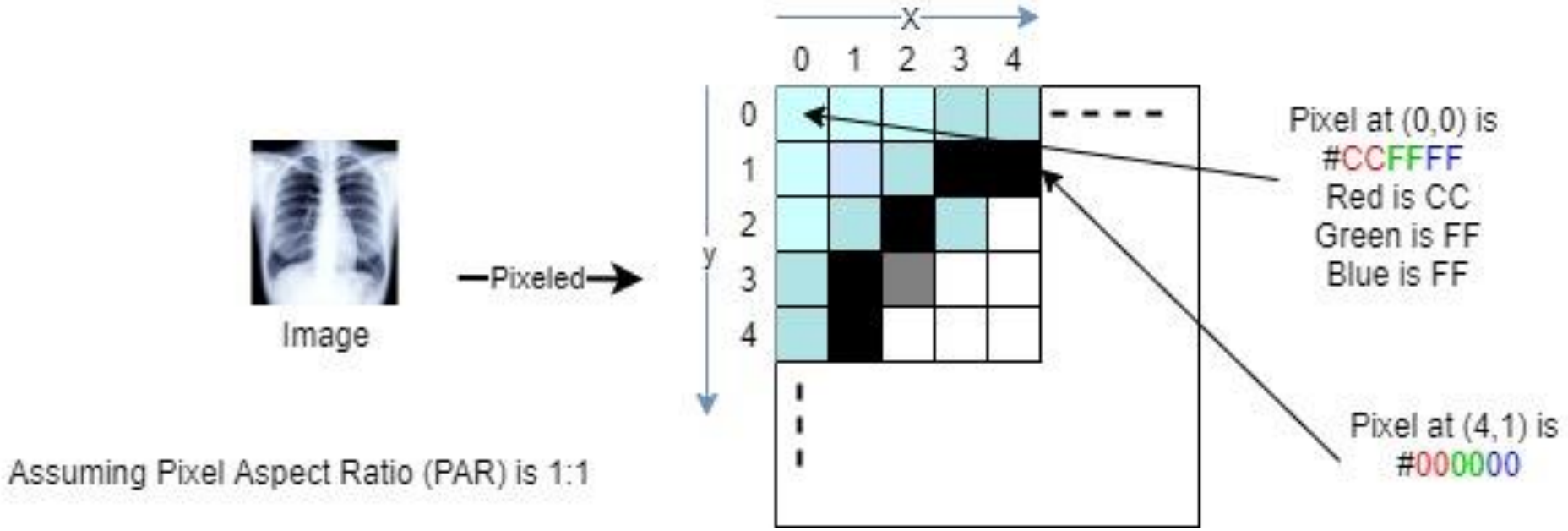
    H2 = H2 + c H3 = H3 + d

    H4 = H4 + e H5 = H5 + f

    H6 = H6 + g H7 = H7 + h
- **Step 5 :** Output
  - When all M_j have been processed, the 256-bit hash of M is available in H0, H1, H2, H3, H4, H5, H6, H7

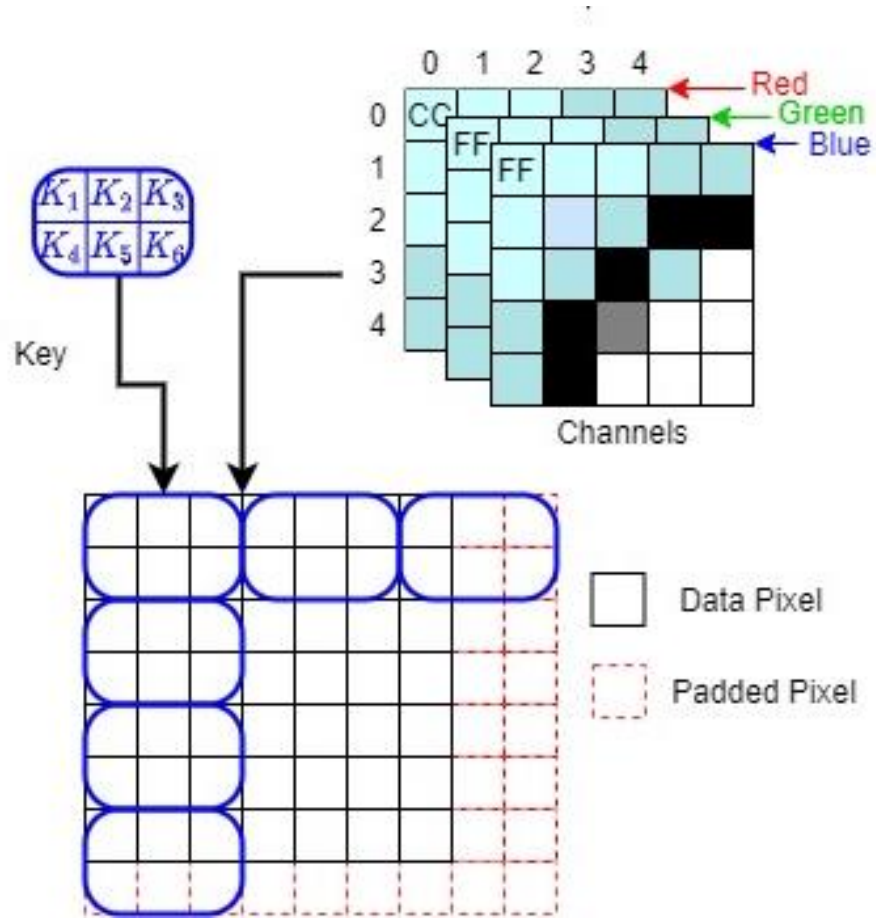
Image Encryption for Blockchain

Image is divided into Pixels



Methodology for Image Encryption

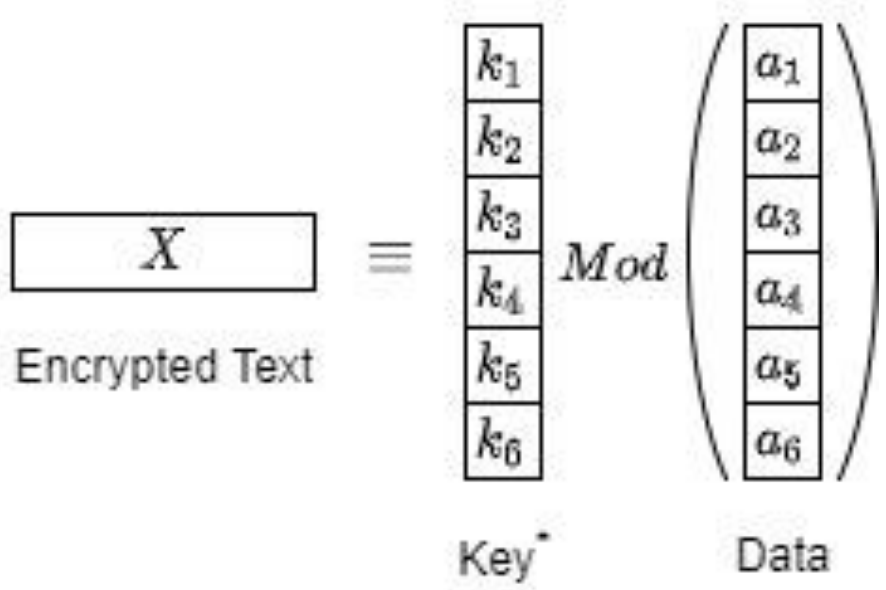
Image is split into channels (colour-wise)



Methodology for Image Encryption

Chinese Remainder Theorem based method is used

to encrypt the pixel values and stored into blockchain



इलेक्ट्रॉनिकी एवं
सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF
ELECTRONICS AND
INFORMATION TECHNOLOGY
सत्यमेव जयते

Digital India
Power To Empower

Thank You