

Cryptographic Primitives

Use of Cryptography in Blockchain

- Blockchain make use of two types of cryptographic algorithms,
 - Asymmetric key algorithm
 - Hash functions.
- Hash function are used to provide the functionality of single view of blockchain to every participant. Blockchain generally use the SHA-256 hashing algorithm as their hash function.
- Hash functions have a major role in linking the blocks to one another Any alteration the block data can lead to inconsistency and break the blockchain, making it invalid. This is achieved by the property of the hash function, called the '**avalanche effect**'.

Benefits of Cryptography in Blockchain

- **Avalanche effect** – A slight change in the data can result in a significantly different output.
- **Uniqueness** – Every input has a unique output.
- **Deterministic** – Any input will always have the same output if passed through the hash function.
- **Quickness** – The output can be generated in a very small amount of time.
- Reverse engineering is not possible, i.e. we cannot generate the input by having the output and the hash function.

Avalanche Effect

Example

According to this, if we make even a slight change in the input to the hash function, we will end up getting a totally unrelated output as compared to the original output. Let us take an example of an SHA-256 hash function, and compare their outputs,

Input: Blockchain at EC-Council

Output: 04f0ecc95159533982d7571eada5f8d76592b6e97ead964467c603d31b9e7a9c

Input with a slight difference: Blockchain at ECCouncil

Output: 80b069904b6a8db46ed94e7091ff4e5fc72fae5422d46cc57d8f66db7abf4781

OVERVIEW

- Terminologies
- Symmetric key algorithms
 - Vernam cipher
 - A5/1
 - DES
 - AES
- Asymmetric key algorithms
 - RSA
 - Diffie Hellman
- Some cryptographic hashes
- Tools for cryptanalysis

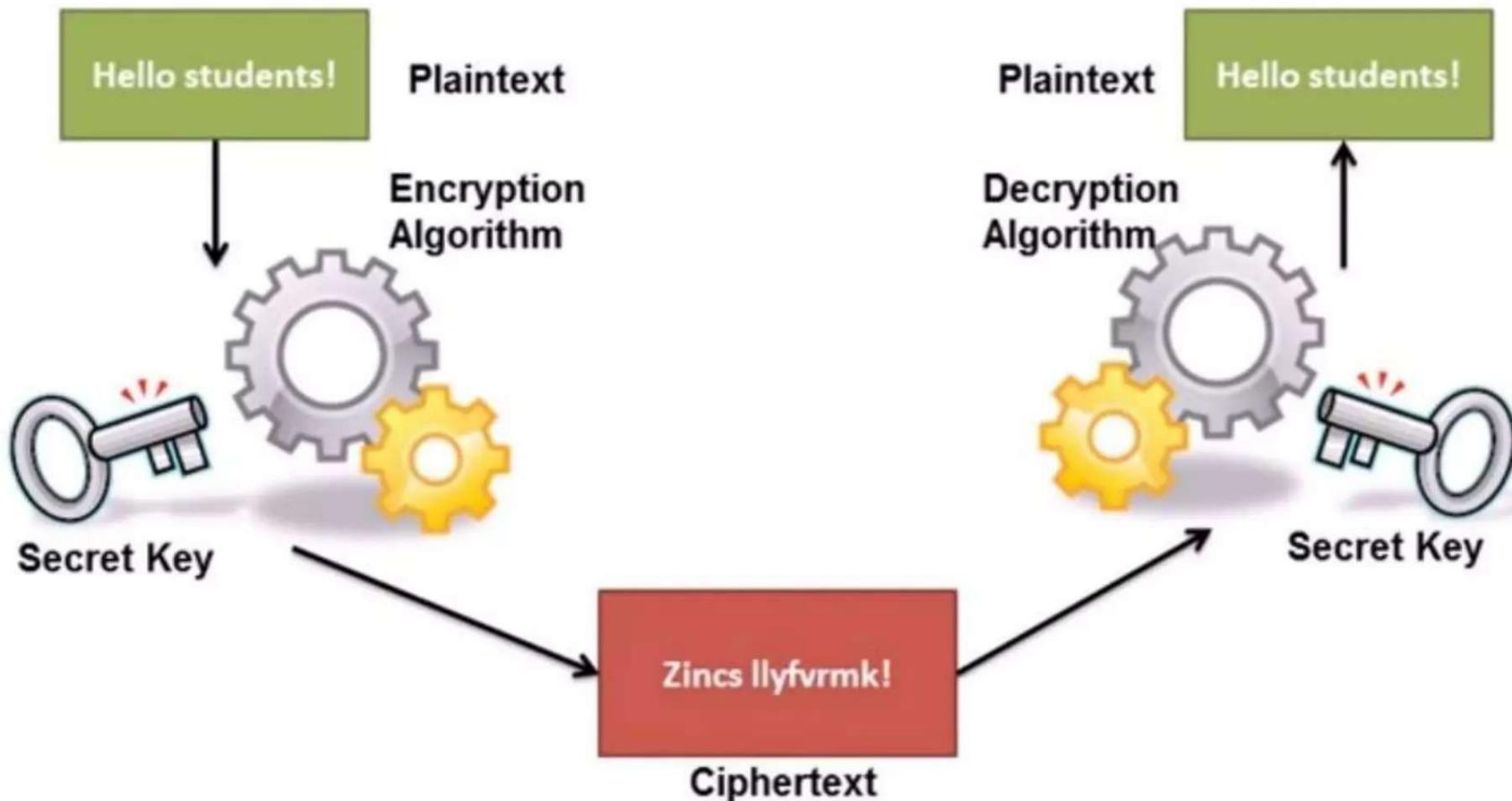
TERMINOLOGY

- **Cryptology**-Art and science of making “secret codes”.
- **Cryptography**- The practice and study of hiding information.
- **Cryptanalysis**-Art of finding some weakness and insecurity in a cryptographic scheme.

CRYPTOGRAPHIC TERMINOLOGY

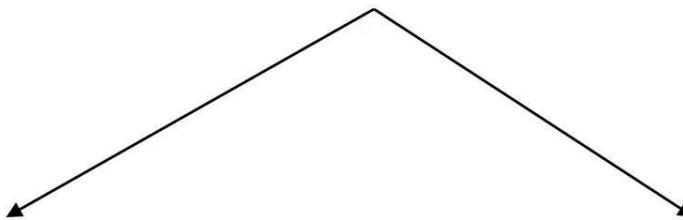
- **Plain text**-The format of the data before being encrypted.
- **Cipher Text**-The “scrambled” format of data after being encrypted.
- **Key**-A secret value used during the encryption and decryption process
- **Encryption**-Method of transforming plain text into an unreadable format
- **Decryption**-Method of obtaining the encrypted message back to its original form.

ENCRYPTION AND DECRYPTION



TYPES OF ALGORITHMS

Cryptographic algorithms



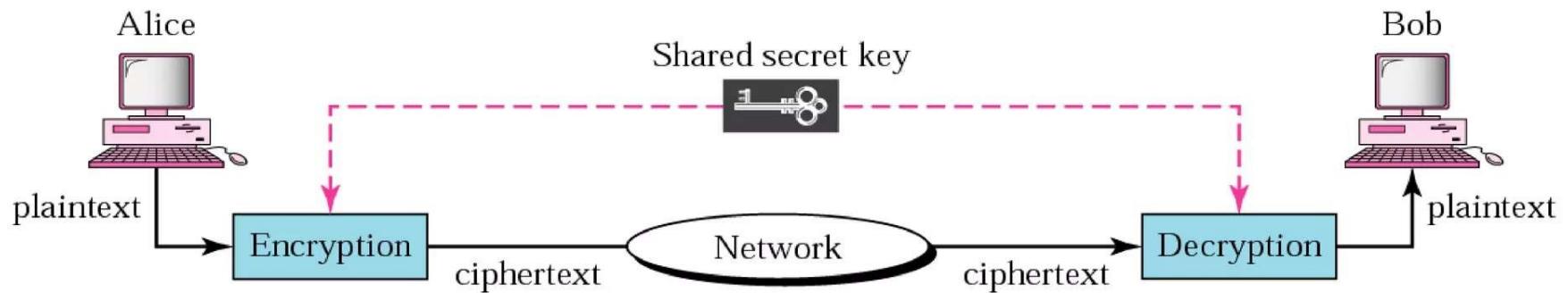
Symmetric key

(Shared secret key)

Asymmetric key

(Public key)

SYMMETRIC KEY ENCRYPTION



TYPES OF SYMMETRIC CIPHERS

- **Stream ciphers**
 - Encrypts one bit/character at a time
- **Block ciphers**
 - Break plaintext message in equal-size blocks
 - Encrypts each block as a unit

SUBSTITUTION CIPHER

- Substituting by a character “key” places ahead of the current character
 - a) **Monoalphabetic cipher** (Stream cipher)
 - Eg. PlainText : THIS IS AN EASY TASK
 - Key : 3
 - Encryption : WKLV LV DQ HDVB WDVN
 - b) **Polyalphabetic cipher** (Block cipher)
 - Eg : THIS IS AN EASY TASK.
 - Make group of 3 characters and a set of keys used could be 135.
THI SIS ANE...
Encryption : UKN TLX...

TRANSPOSITION CIPHER

- Transposition ciphers use the letters of the plaintext message, but they permute the order of the letters.

Encrypt : hello my dear friend

Key: 2143

- Remove spaces
- Divide the text into blocks of 4 characters.
- Add bogus character(s) at the end(if required).

Ciphertext:

After decryption :

hello myde arfr iend
ehol ymed rarf eidn
hello myde arfr iend

VERNAM CIPHER

- Each character from the plaintext is encrypted by a modular addition which a number from the **secret random key pad** which is of the same length as the plain text.

Step 1: Convert the letters to their numeric equivalents

V	E	R	N	A	M	C	I	P	H	E	R
21	4	17	13	0	12	2	8	15	7	4	17

Assume the random 2 digit no. series (key)

76 48 16 82 44 03 58 11 60 05 10 88

Step 2: Add the numeric equivalent and the corresponding random no.

Random no + numeric equivalent =sum

Sum 97 52 33 95 44 15 60 19 75 12 14 105

VERNAM CIPHER

Step 3 : Perform sum mod 26

19 0 7 17 18 15 8 19 23 12 14 1

Ciphertext ----→

t a h r s p i t x m o b

Decryption

Step 1

$a = (\text{numeric equivalent of ciphertext} - \text{key})$

Step 2

$a \bmod 26$

(if a negative then keep adding 26 till you get a positive no.)

Step 3

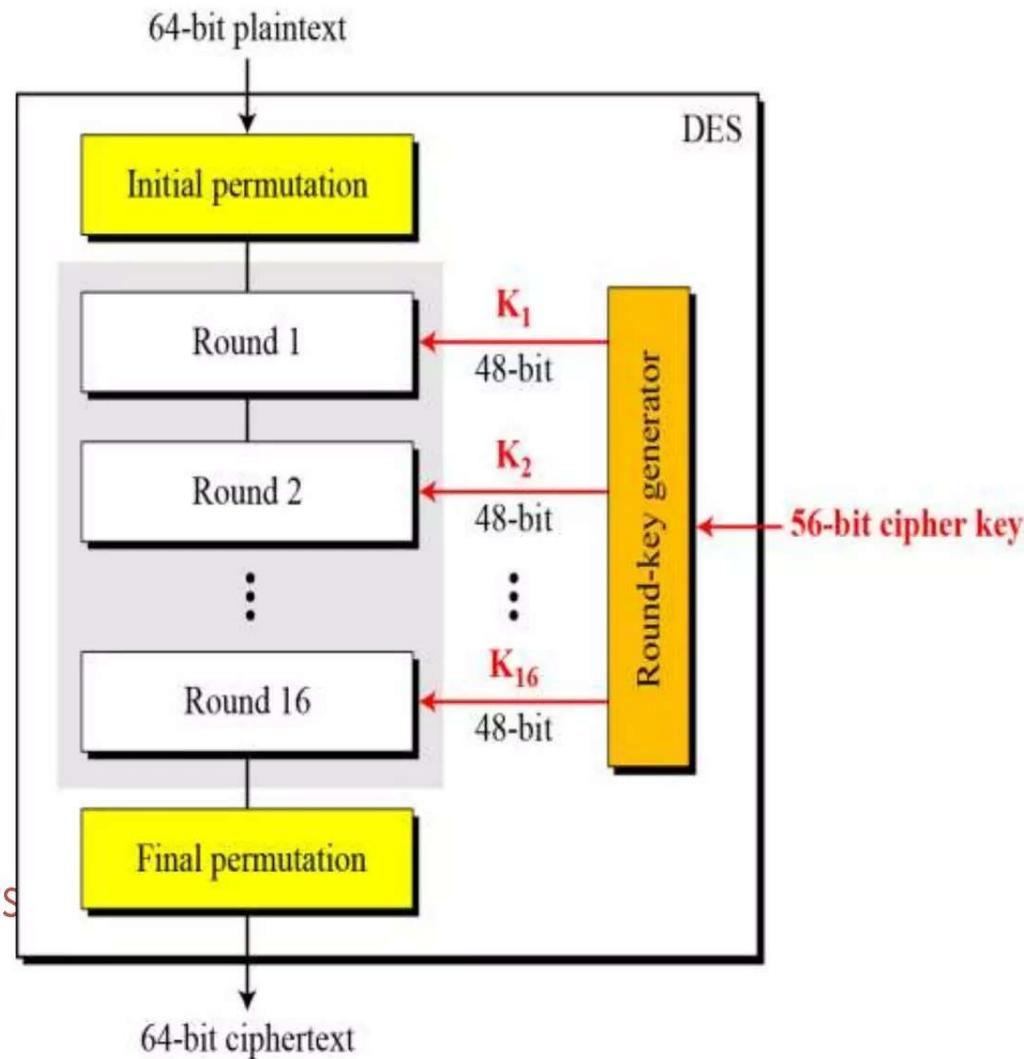
Convert numeric equivalent back to alphabet

DATA ENCRYPTION STANDARD (DES)

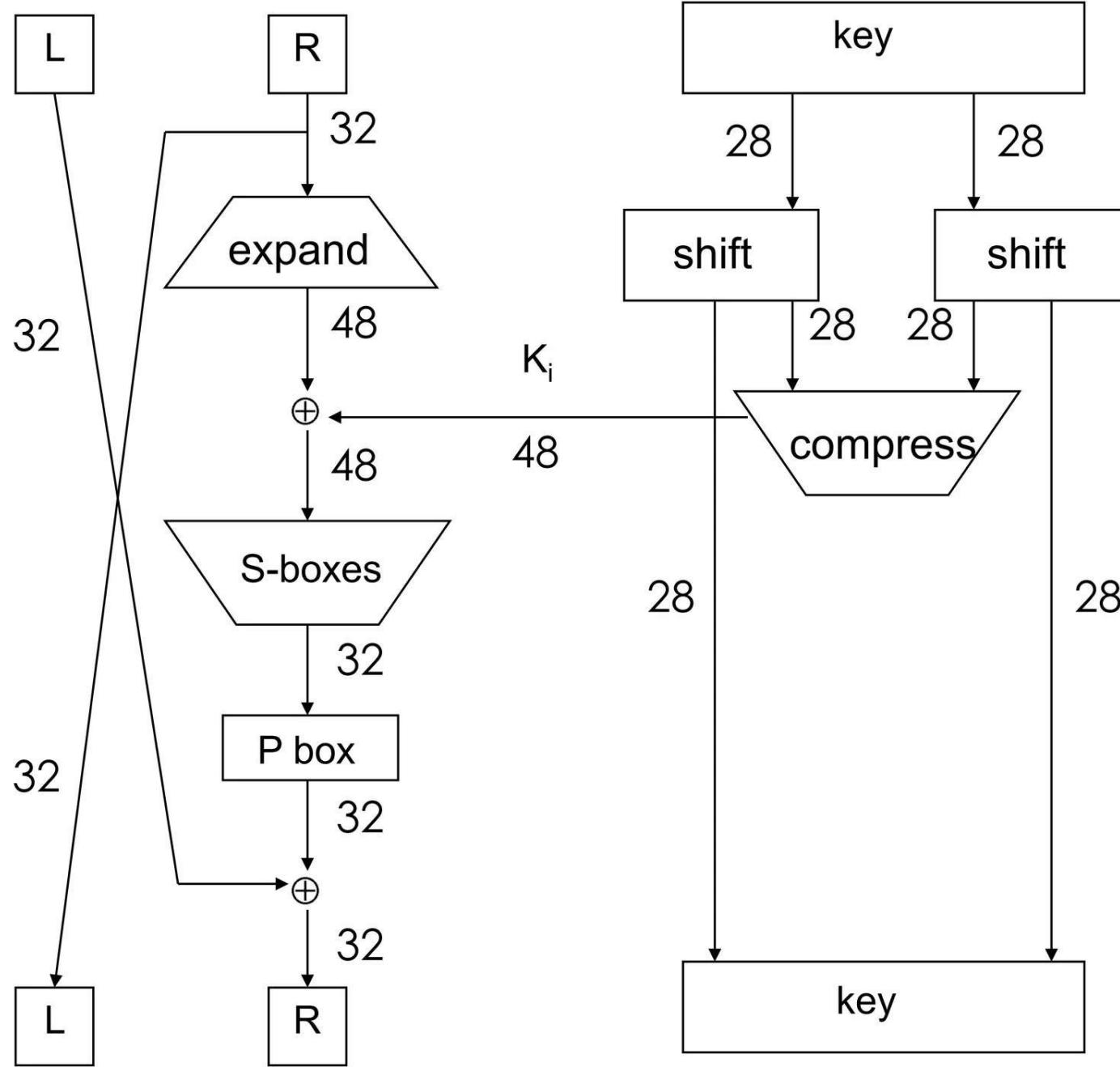
- Modern symmetric key block cipher.
- Developed by IBM and then published by National Institute of standards and technology(NIST).
- Vulnerable only because of its small key length.
- Often used in VPN servers.

DES ALGORITHM

- DES is a Feistel cipher
 - 64 bit block length
 - 56 bit key length
 - 16 rounds
 - 48 bits of key used each round (subkey)
- Each round is simple (for a block cipher)
- Security depends primarily on “S-boxes”
 - Each S-boxes maps 6 bits to 4 bits



One Round of DES



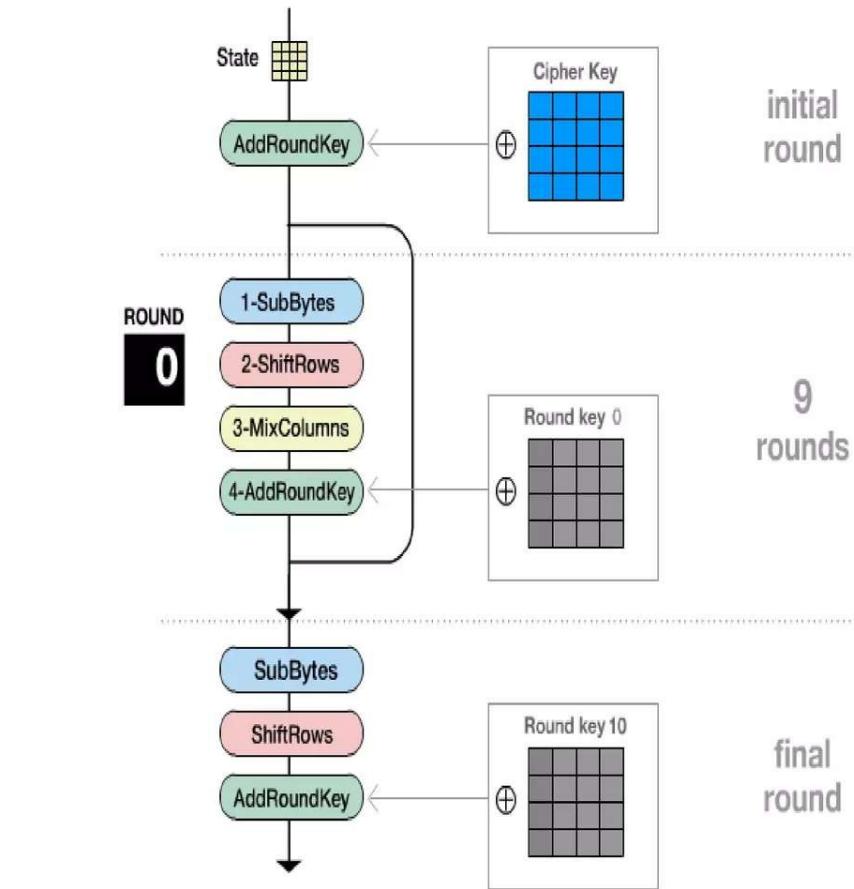
ADVANCED ENCRYPTION STANDARD (AES)

- Replacement for DES
- AES competition (late 90's)
 - NSA openly involved
 - Transparent process
 - Many strong algorithms proposed
 - Rijndael Algorithm ultimately selected
- Iterated block cipher (like DES)
- Not a Feistel cipher (unlike DES)
- 3 versions are : AES - 128
 - AES - 192
 - AES – 256
- Used in Open SSL and WPA2

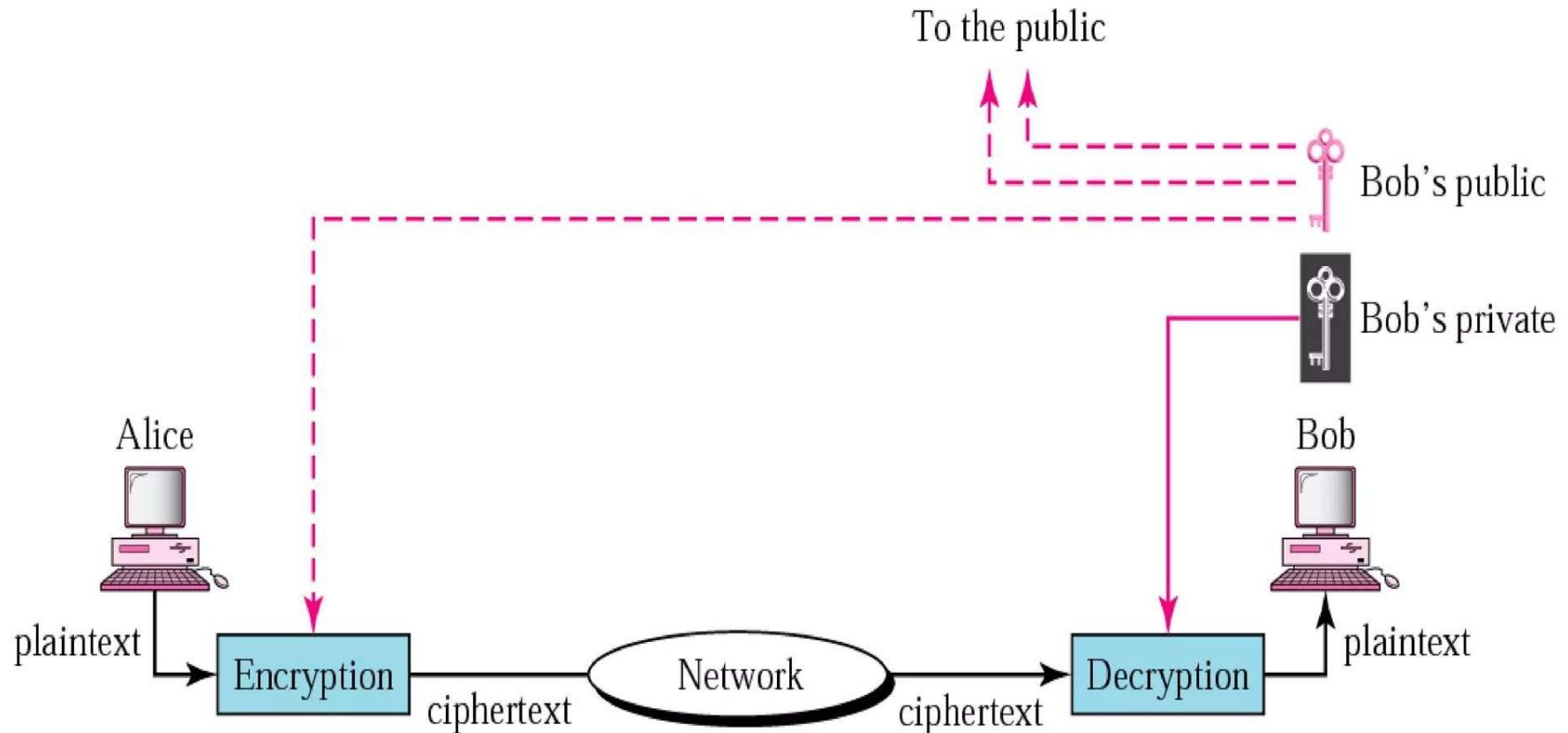
AES OVERVIEW

- **Block size:** 128, 192 or 256 bits
- **Key length:** 128, 192 or 256 bits (independent of block size)
- **10 to 14 rounds** (depends on key length)
- Each round uses **4 functions** (in 3 “layers”)
 - **ByteSub** (nonlinear layer)
 - **ShiftRow** (linear mixing layer)
 - **MixColumn** (nonlinear layer)
 - **AddRoundKey** (key addition layer)

Encryption process



ASYMMETRIC KEY ENCRYPTION



RSA

- The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).
- Applications
 1. To protect web traffic, in the SSL protocol (**Security Socket Layer**),
 2. To guarantee email privacy and authenticity in **PGP** (**Pretty Good Privacy**)
 3. To guarantee remote connection in **SSH** (**Secure Shell**)
 4. Furthermore it plays an important role in the modern payment systems through **SET** protocol (**Secure Electronic Transaction**).

ALGORITHM

- Let p and q be two large prime numbers
- Let $N = pq$ be the modulus
- Find $\phi(n) = (p-1).(q-1)$
- Choose e such that it is relatively prime to $\phi(n)$.
- Choose d such that : $e \times d \text{ mod } \phi(n) = 1$
- Public key is (N, e)
- Private key is d
- To encrypt message M compute
 - $C = M^e \text{ mod } N$
- To decrypt C compute
 - $M = C^d \text{ mod } N$

RSA -Example

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select: $e=7$
5. Determine $d= 23$
6. Publish public key $KU=\{7\}$
7. Keep secret private key $KR=\{23\}$

RSA

- Encryption: The key is used to convert a plain-text to a cypher-text; $M' = E(M, k)$
- $M' = (M^e) \text{ mod } n$
- Decryption: The key is used to convert the cypher-text to the original plain text;

$$M = D(M', k) = M = (M'^d) \text{ mod } n$$

Cryptographic primitives - blockchain

- Basic cryptographic primitives behind blockchain technology
 - Cryptographically Secure Hash Functions
 - Digital Signature
- **Hash Function:** Used to connect the “**blocks**” in a “**chain**” in a **tamper-proof way**
- **Digital Signature:** Digitally sign the data so that no one can “**deny**” about their own activities. Also, others can check whether it is authentic.

Cryptographic Hash Functions

- Takes any arbitrarily sized string as input

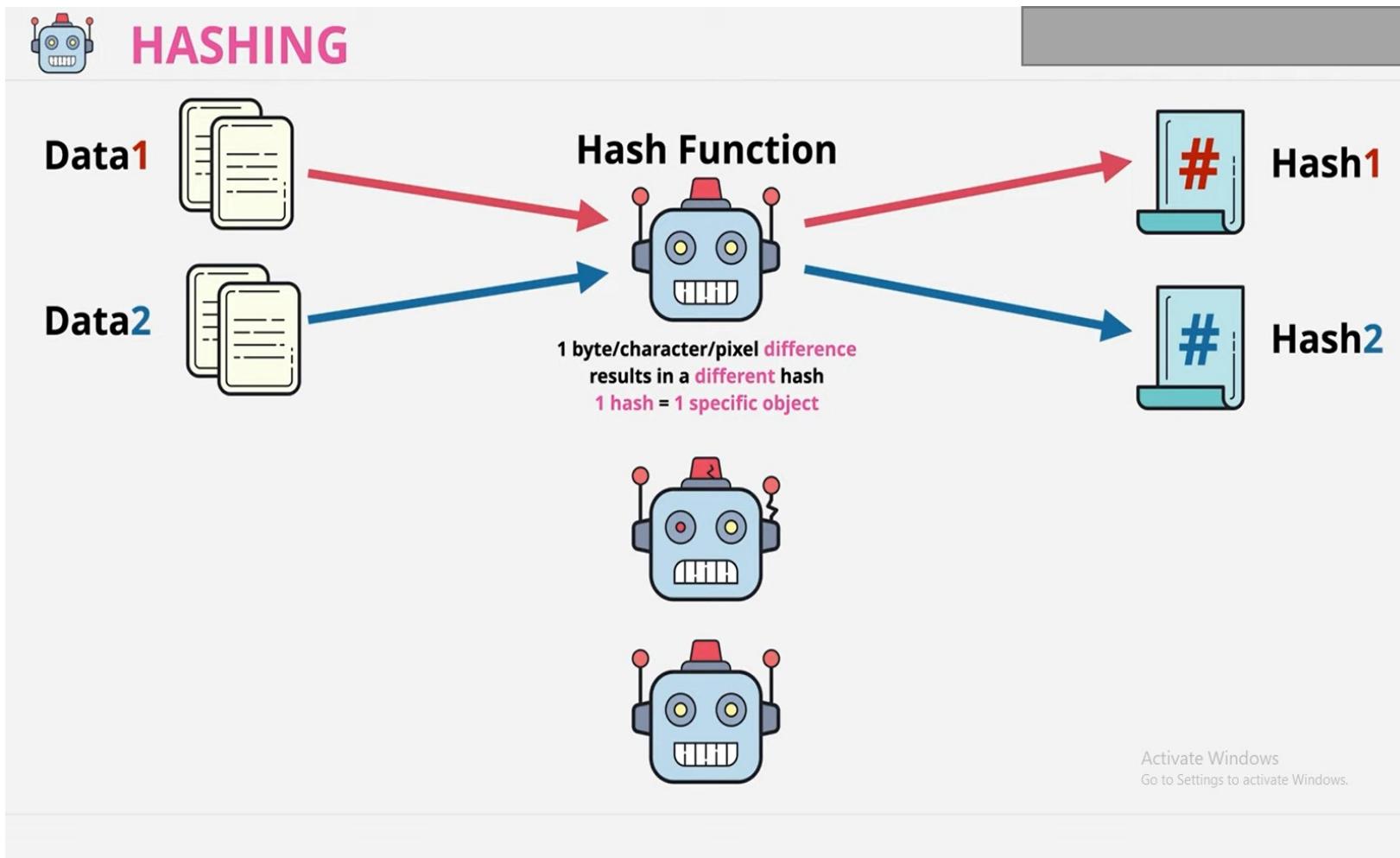
Input M: The message

- Fixed size output (We typically use 256 bits in Blockchain)
- Output $H(M)$: We call this as the message digest.
- Efficiently computable

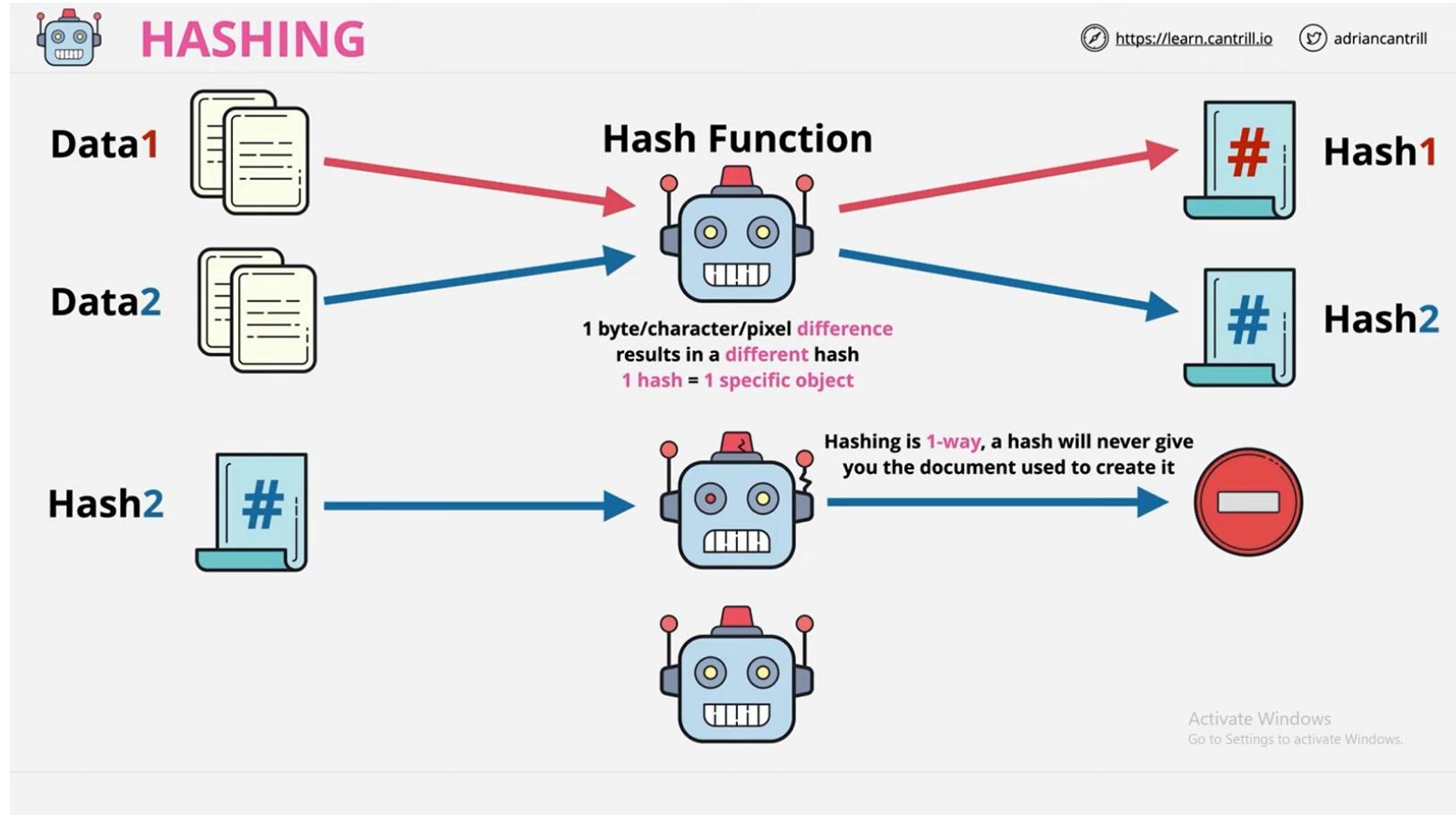
Cryptographic Hash Functions: Properties

- **Deterministic**
Always yields identical hash value for identical input data
- **Collision-Free**
If two messages are different, then their digests also differ
- **Hiding** : Hide the original message; remember about the avalanche effect
- **Puzzle-friendly** : Given X and Y, find out k such that $Y = H(X \parallel k)$ - used to solve the mining puzzle in Bitcoin Proof of Work
- <http://www.blockchain-basics.com/HashFunctions.html>

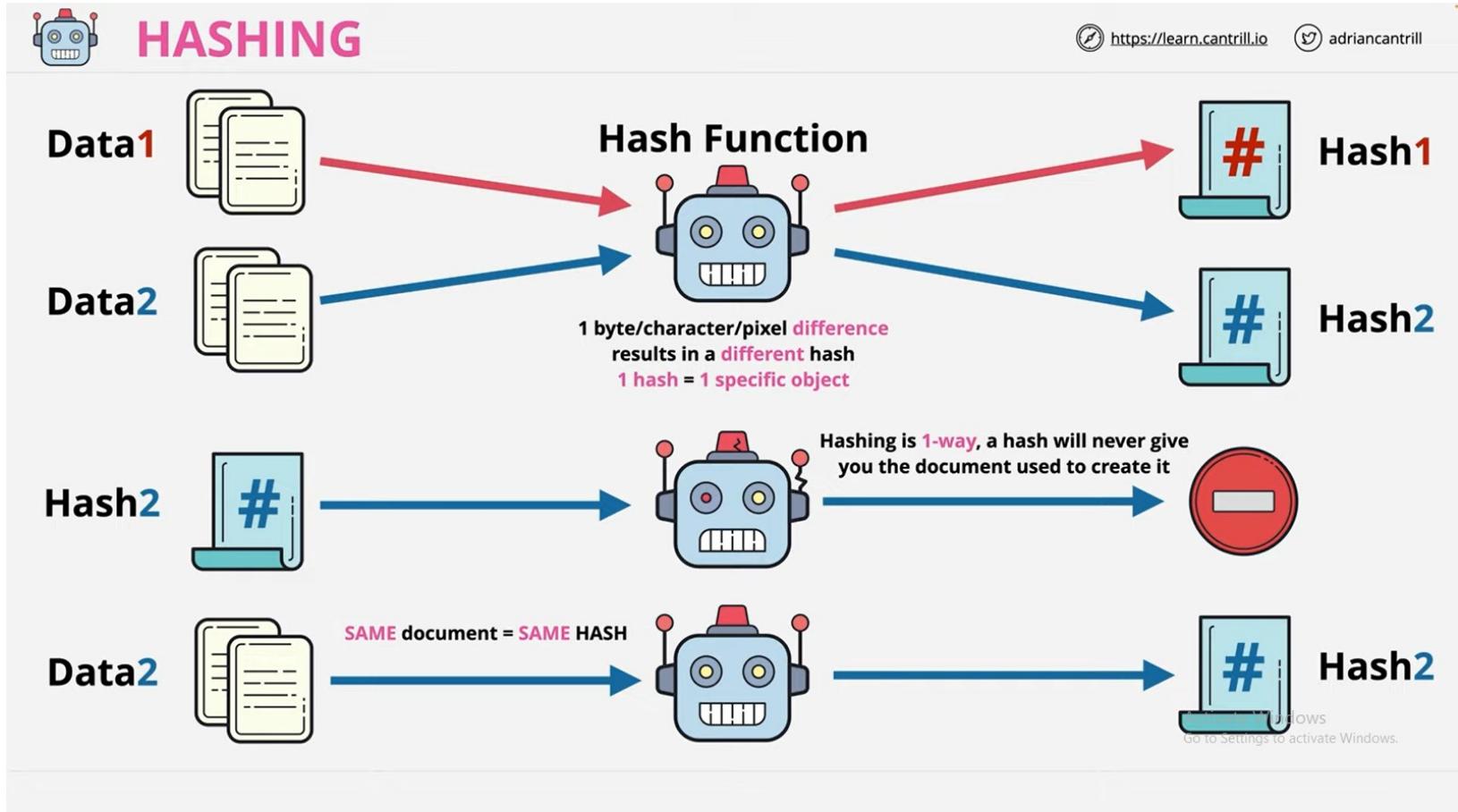
No two data will have a same hash value.



By sending Hash as input, we cannot get the original data.



Same data, we will get the same Hash.



Hash functions

- Hash functions are used to create **fixed-length** digests of **arbitrarily-long input** strings.
- Hash functions **are keyless**, and they provide the data **integrity service**.
- They are usually built using iterated and dedicated hash function construction techniques. Various families of hash functions are available, such as **MD**, **SHA-1**, **SHA-2**, **SHA-3**, **RIPEMD**, and **Whirlpool**.
- Hash functions are commonly used for **digital signatures and Message Authentication Codes (MACs)**, such as **HMACs**.
- They have 3 security properties
 - **preimage resistance**
 - **second preimage resistance and**
 - **collision resistance**.

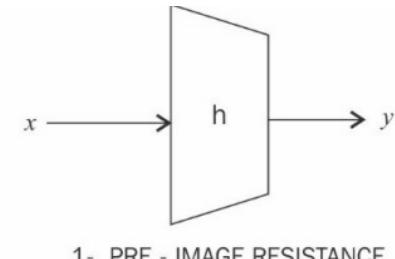
Hash functions

- It is Easy to compute
- **Preimage resistance:** $h(x) = y$; h is the hash function, x is the input, and y is the hash.
- The first security property requires that y cannot be reverse-computed to x . x is **considered a preimage of y** , This is also called a **one-way** property.
- **Second preimage resistance:** given x and $h(x)$, it is almost impossible to find any other message m , where $m \neq x$ and $h(m) = h(x)$.
- This property is also known as **weak collision resistance**.

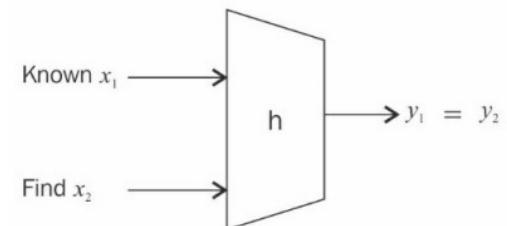
Hash functions

- **Collision resistance:** requires that two different input messages should not hash to the same output. In other words, $h(x) \neq h(z)$.

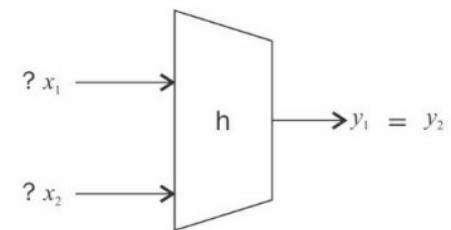
This property is also known as strong collision resistance.



1- PRE - IMAGE RESISTANCE



2- SECOND PRE IMAGE RESISTANCE



3- STRONG COLLISION RESISTANCE

Three security properties of hash functions

Hash functions

- **SHA-0:** This is a 160-bit function introduced by NIST in 1993.
- **SHA-1:** was introduced in 1995 by NIST as a replacement for SHA-0. This is also a 160-bit hash function. SHA-1 is used commonly in SSL and TLS implementations.
- **SHA-2:** This category includes four functions defined by the number of bits of the hash: SHA-224, SHA256, SHA-384, and SHA-512.
- **SHA-3:** This is the latest family of SHA functions. SHA-3-224, SHA-3-256, SHA-3-384, and SHA-3- 512 are members of this family.

Hash functions

- **RIPEMD:** is the acronym for RACE Integrity Primitives Evaluation Message Digest. It is based on the design ideas used to build MD4.
- There are multiple versions of RIPEMD, including 128-bit, 160-bit, 256-bit, and 320-bit.
- **Whirlpool:** This is based on a modified version of the Rijndael cipher known as W. It uses the MiyaguchiPreneel compression function, which is a type of one-way function used for the **compression of two fixed length inputs into a single fixed-length output**.

Hash functions

- Hash functions have many **practical applications** ranging from simple file integrity checks and password storage to use in cryptographic protocols and algorithms.
- They are used in hash tables, distributed hash tables, bloom filters, virus fingerprinting, peer-to-peer file sharing, and many other applications.
- Hash functions play a vital role in blockchain. The **PoW function** in particular **uses SHA-256 twice** in order to verify the computational effort spent by miners.
- **RIPEMD 160** is used to produce **Bitcoin addresses**.

MD5

- MD5 is quite fast and produces 128 bit message digest
- After some initial processing the input text is processed in 512 bit of blocks.
- This 512 bit blocks are divided into 16, 32 bit sub blocks ($16 \times 32 = 512$)
- The output is set of 4×32 bit blocks = 128 bits message digest

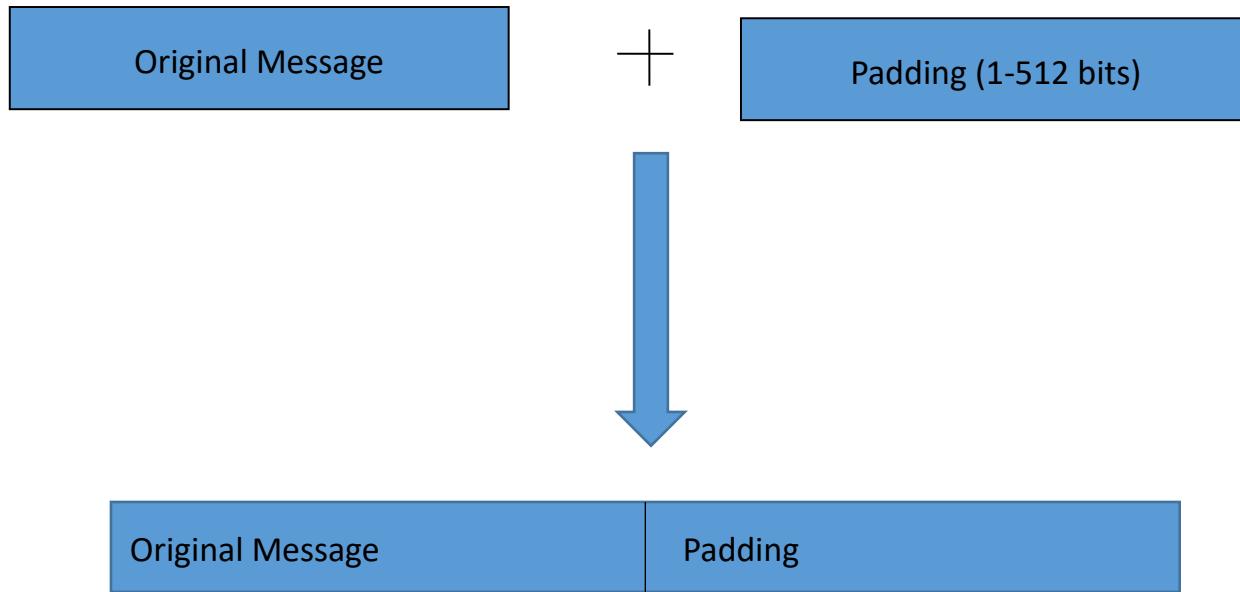
Working of MD5

1. Padding
2. Append length
3. Divide the input into 512 bit blocks
4. Initialize chaining variables
5. Process blocks

Step 1: Padding

- The aim of this step is to make the length of the original message equal to a value which is **64 bit less than the exact multiple of 512**

- 512 512-64 448
- 512 x 2 1024-64 960
- 512 x 3 1536-64 1472



Total length should be 64 bit less than the exact multiple of 512

If 448 bits ($448 = 512 - 64$)

If 960 bits ($960 = 2 \times 512 - 64$)

If 1472 bits ($1472 = 3 \times 512 - 64$)

Note: Padding is always added even if it is in terms of 512

Step 1 cont'd

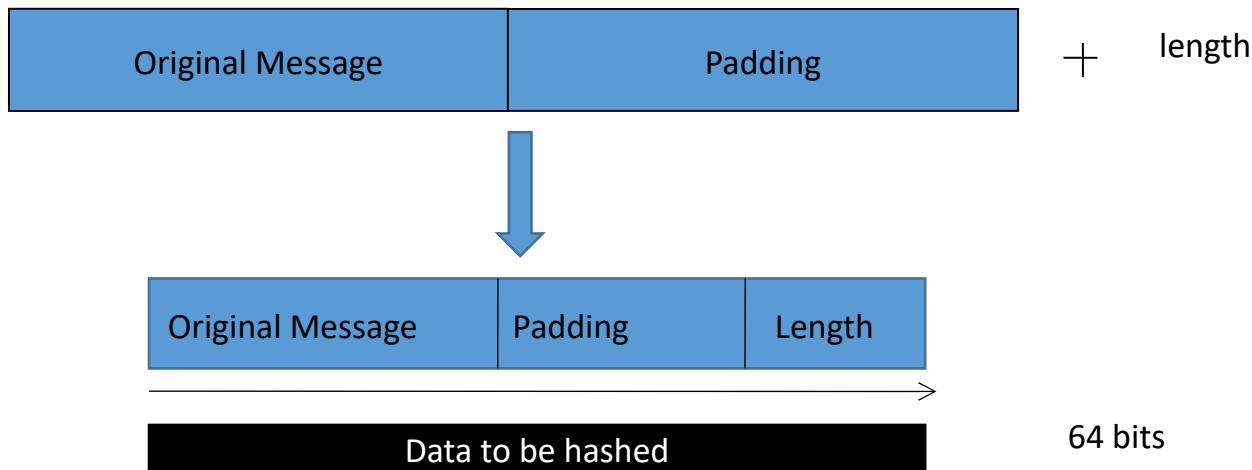
- If message length is 1000 add 472 = 1472
- If message length is 448 add 512 = 960 , even though 448 is 64 bits less than multiple of 512

Reason

512	512-64	448
512 x 2	1024-64	960
512 x 3	1536-64	1472

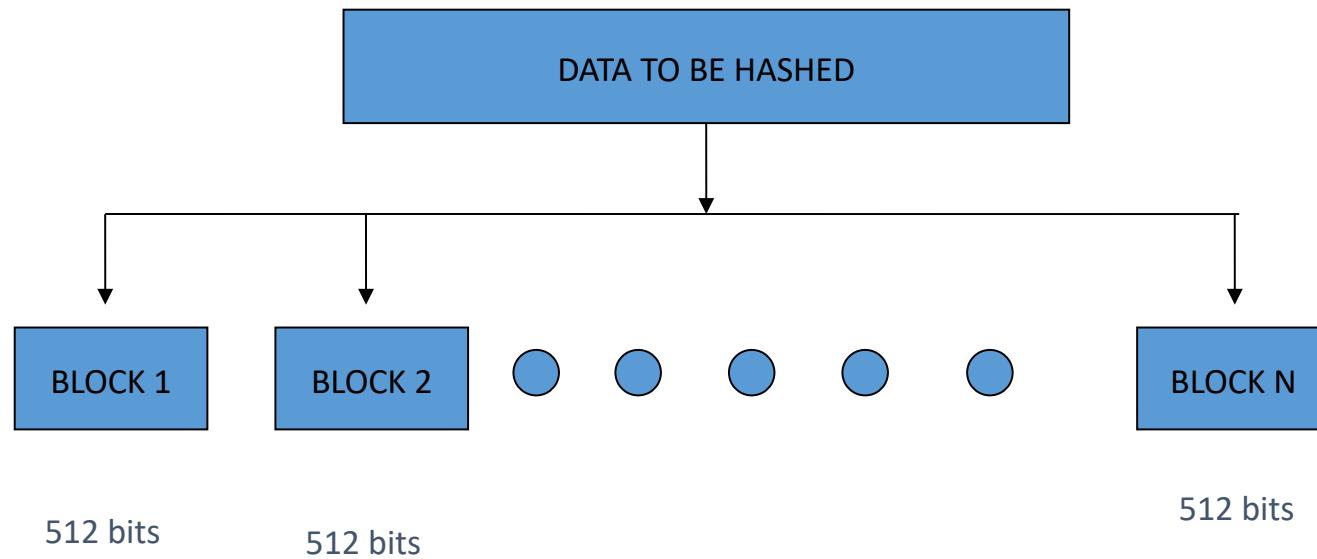
Step 2: Append length

- The length of the original message is Calculated and is appended to the end of the message block + padding



- The length of the original message (excluding the padding) is calculated
- It is appended to the end of the message block + Padding
- The length is expressed in 64 bits
- If the length of the message exceeds 2^{64} bits of length then only last 64 bits are used. (by taking mod 64)
- After 64 bit of length is appended this becomes the final message
- This final message is to be hashed
- The length of the message is in terms of 512 bits

Step 3: Divide the i/p into 512 bits blocks



Step 4 – Initialize chaining variables

- Four variables called as chaining variables are initialized
- They are called as A B C and D
- Each of these is a 32 bit number
- The initial hex values are

A Hex 01 23 45 67

B Hex 89 AB CD EF

C Hex FE DC BA 98

D Hex 76 54 32 10

Step 5

- After all the initializations the real algorithm begins
- Its quiet complicated , we will discuss it step by step
- There is a loop that runs for as many 512 bit blocks as are in the message

Step 5.1 : Process blocks

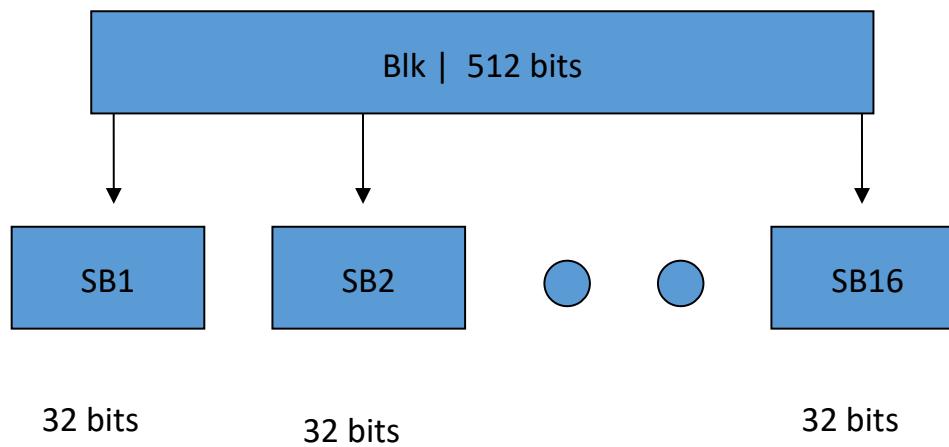
- Copy the 4 chaining variables into 4 corresponding variables a ,b ,c, d
- A -> a
- B -> b
- C -> c
- D -> d



128 bit single register

Step 5.2 Divide

- Divide the current 512 bit block into 16 sub blocks M [i]



Step 5.3

- There are 4 rounds.
- In each round process all the 16 sub blocks
- Inputs to each round are
 - All the 16 sub blocks
 - a, b, c, d
 - Some constants (t)

t

- t is an array of constants
- It contains 64 elements with each element consisting of 32 bits
- t[1] , t[2]t[64]

What is done in these four rounds?

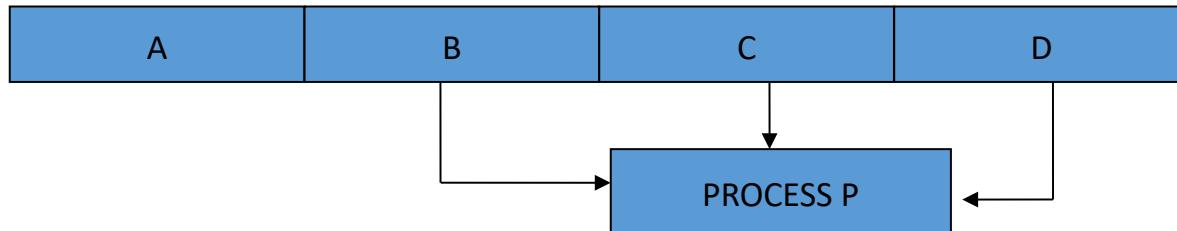
- All the four rounds vary in one major way
- STEP 1 of the rounds has different processing
- The other steps in all the four rounds are the same
 - In each round we have 16 input sub blocks names $M[0]$, $M[1]$, $M[15]$ each of 32 bits
 - We have $t[k]$, k varies form I to 64,
 - 16 out of 64 values of t is used in each round

Iterations on all 4 rounds

After copying abcd we have 16 such iterations

1. Process P is performed on b,c,d . This process P is different in all the four rounds

Process P

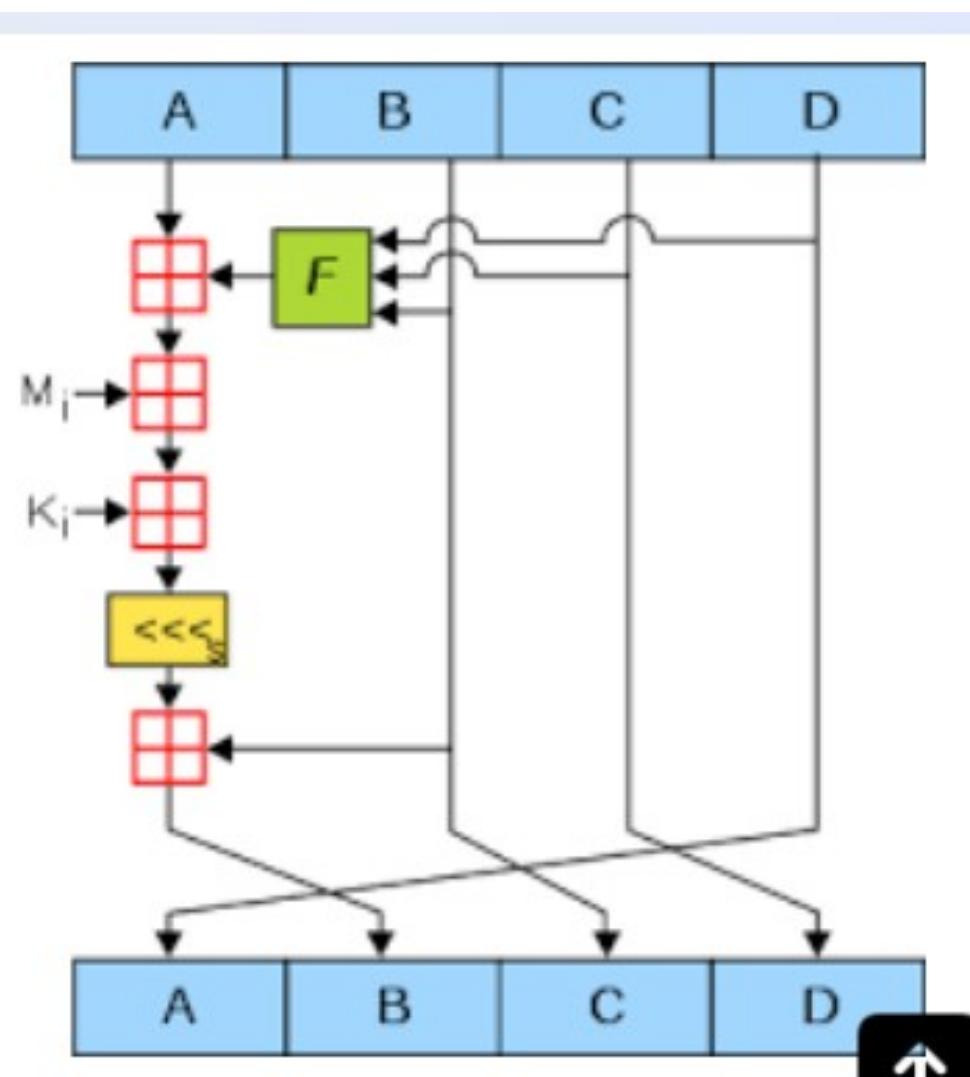


Process P: some basic Boolean operations on b,c,d

Round	Process P
1	$(b \text{ AND } c) \text{ OR } ((\text{NOT } b) \text{ AND } (d))$
2	$(b \text{ AND } d) \text{ OR } (c \text{ AND } (\text{NOT } d))$
3	$(b \text{ XOR } c \text{ XOR } d)$
4	$c \text{ XOR } (b \text{ OR } (\text{NOT } d))$

2. The variable a is added to the output of process P
3. The message sub block $M[i]$ is added to the output of step 2 i.e. to the register abcd
4. The constant $t[k]$ is added to the output of step 3 i.e. to the register abcd
5. The output of step 4 is circular left shifted by s bits
6. The variable b is added to the output of step 5
7. The output of step 6 now becomes abcd for the next step

MD5



Mathematically expressing MD5

- $b = b + ((a + \text{process } P(b,c,d) + M[i] + T[k] \ll s))$
- abcd – chaining variables
- Process P – A non linear operation
- $M[i]$, $T[i]$ – Sub blocks, constants
- $\ll s$ – Circular left shift by s bits

Strength of MD5

- The possibility that 2 messages producing the same message digest using MD5 is in the order of 2^{64}

Collision Problem in MD5

- Hash functions

- Take an input of any size
- Create a fixed size string
- Message digest, checksum

- The hash should be unique

- Different inputs should never create the same hash
- If they do, it's a collision

- MD5 has a collision problem

- Found in 1996
- Don't use MD5 for anything important

```
d131dd02c5e6eec4693d9a0698aff95c 2fcab58712467eab4004583eb8fb7f89  
55ad340609f4b30283e488832571415a 085125e8f7cdc99fd91dbdf280373c5b  
d8823e3156348f5bae6dacd436c919c6 dd53e2b487da03fd02396306d248cda0  
e99f33420f577ee8ce54b67080a80d1e c69821bcb6a8839396f9652b6ff72a70
```

```
d131dd02c5e6eec4693d9a0698aff95c 2fcab50712467eab4004583eb8fb7f89  
55ad340609f4b30283e4888325f1415a 085125e8f7cdc99fd91dbd7280373c5b  
d8823e3156348f5bae6dacd436c919c6 dd53e23487da03fd02396306d248cda0  
e99f33420f577ee8ce54b67080280d1e c69821bcb6a8839396f965ab6ff72a70
```

MD5 hash: 79054025255fb1a26e4bc422aef54eb4

Activate Windows
Go to Settings to activate Windows.

Design of SHA-256

- SHA-256 has the input message size < 2⁶⁴ -bits. Block size is 512-bits, and it has a word size of 32-bits.
- The output is a **256-bit digest**. The compression function processes a 512-bit message block and a 256-bit intermediate hash value.
- There are **two main components** of this function: the **compression function** and a **message schedule**.

Design of SHA-256

1. Preprocessing

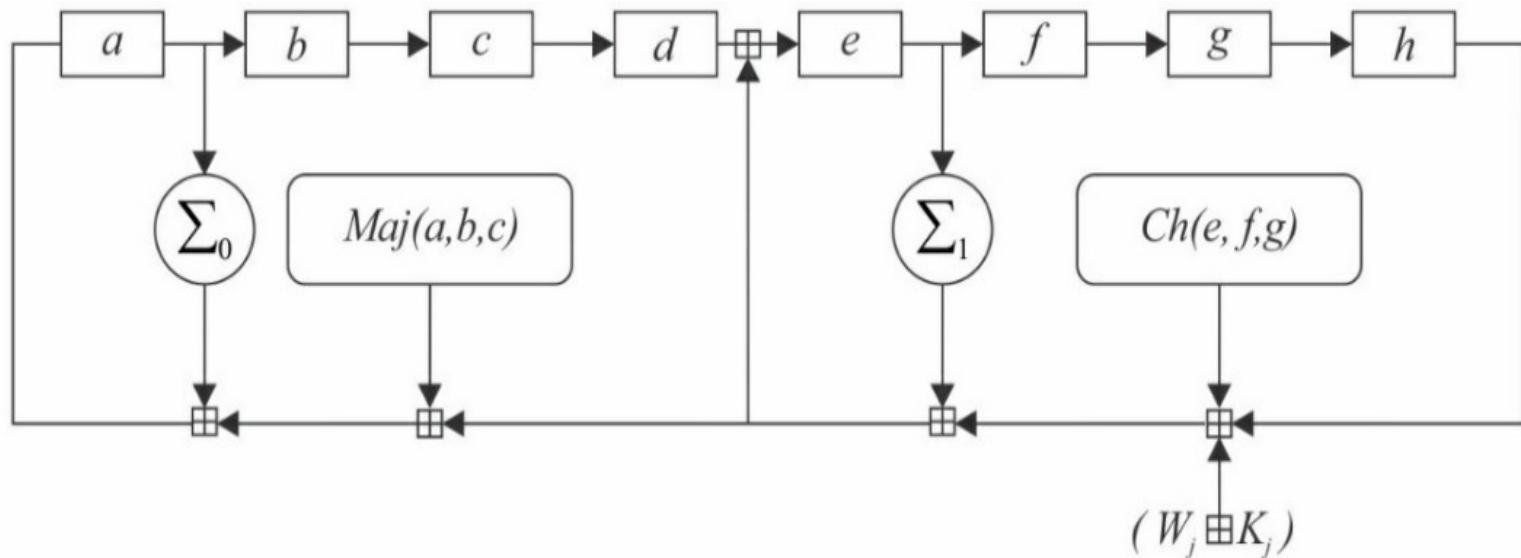
- Padding of the message is used to adjust the length of a block to 512-bits if it is smaller than the required block size of 512-bits.
- Parsing the message into message blocks, which ensures that the message and its padding is divided into equal blocks of 512-bits.
- Setting up the initial hash value, which consists of the eight 32-bit words obtained by taking the first 32-bits of the fractional parts of the square roots of the first eight prime numbers.
- These initial values are randomly chosen to initialize the process, and they provide a level of confidence that no backdoor exists in the algorithm.

Design of SHA-256

2. Hash computation:

- Each message block is then processed in a sequence, and it requires 64 rounds to compute the full hash output.
- Each round uses slightly different constants to ensure that no two rounds are the same.
- The message schedule is prepared.
- Eight working variables are initialized.
- The intermediate hash value is calculated.
- Finally, the message is processed, and the output hash is produced

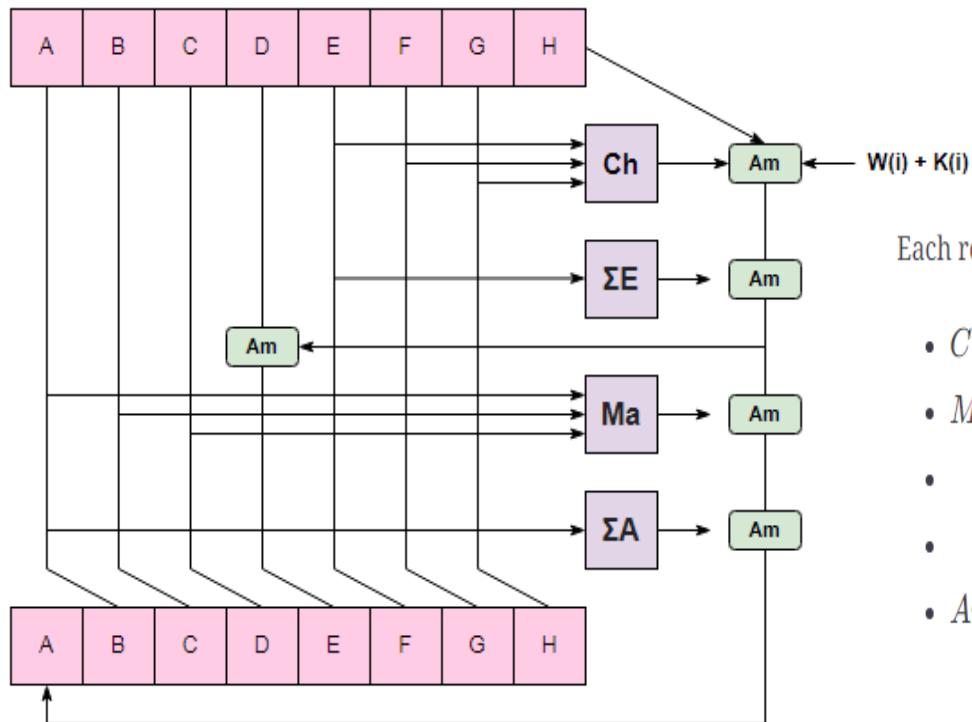
Design of SHA-256



One round of a SHA-256 compression function

In the preceding diagram, **a**, **b**, **c**, **d**, **e**, **f**, **g**, and **h** are the registers. *Maj* and *Ch* are applied bitwise. Σ_0 and Σ_1 performs bitwise rotation. Round constants are W_j and K_j , which are added, *mod* 2^{32} .

SHA-256



Each round carries out the following set of computations:

- $Ch = (E \text{ AND } F) \text{XOR } ((\text{NOT } E) \text{ AND } G)$
- $Ma = (A \text{ AND } B) \text{XOR } (A \text{ AND } C) \text{XOR } (B \text{ AND } C)$
- $A = (A >>> 2) \text{XOR } (A >>> 13) \text{XOR } (A >>> 22)$
- $E = (E >>> 6) \text{XOR } (E >>> 11) \text{XOR } (E >>> 25)$
- $Am = (\text{addition}) \bmod (2^{32})$

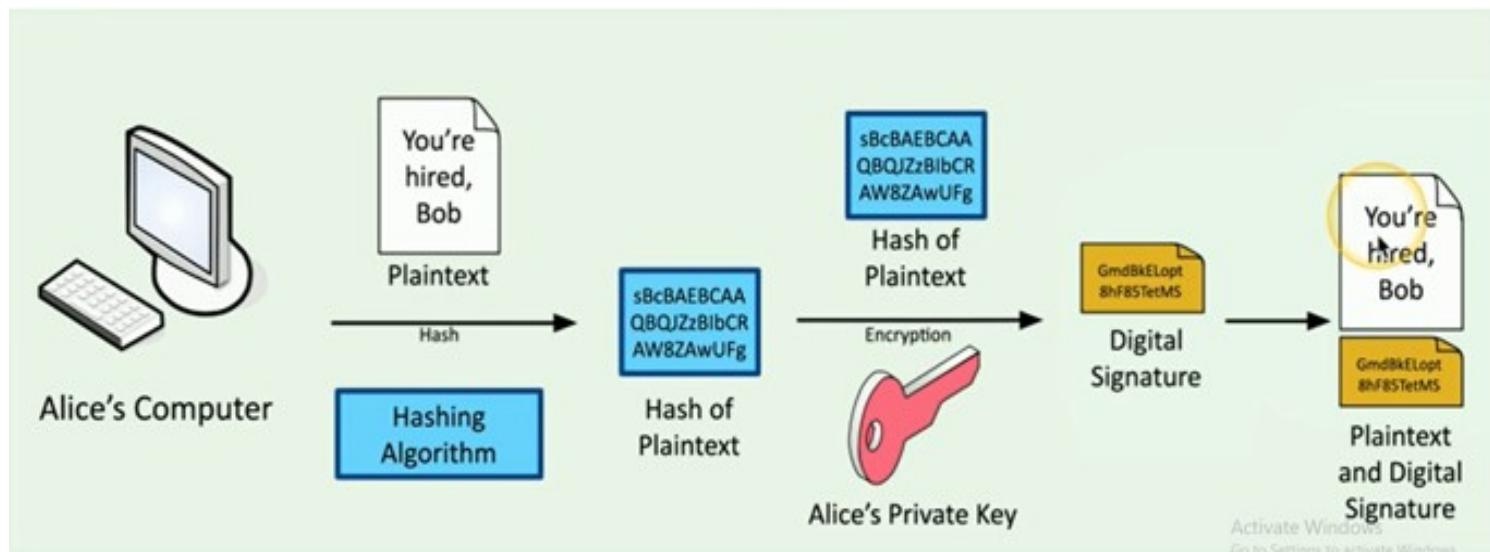
SHA Versions

	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

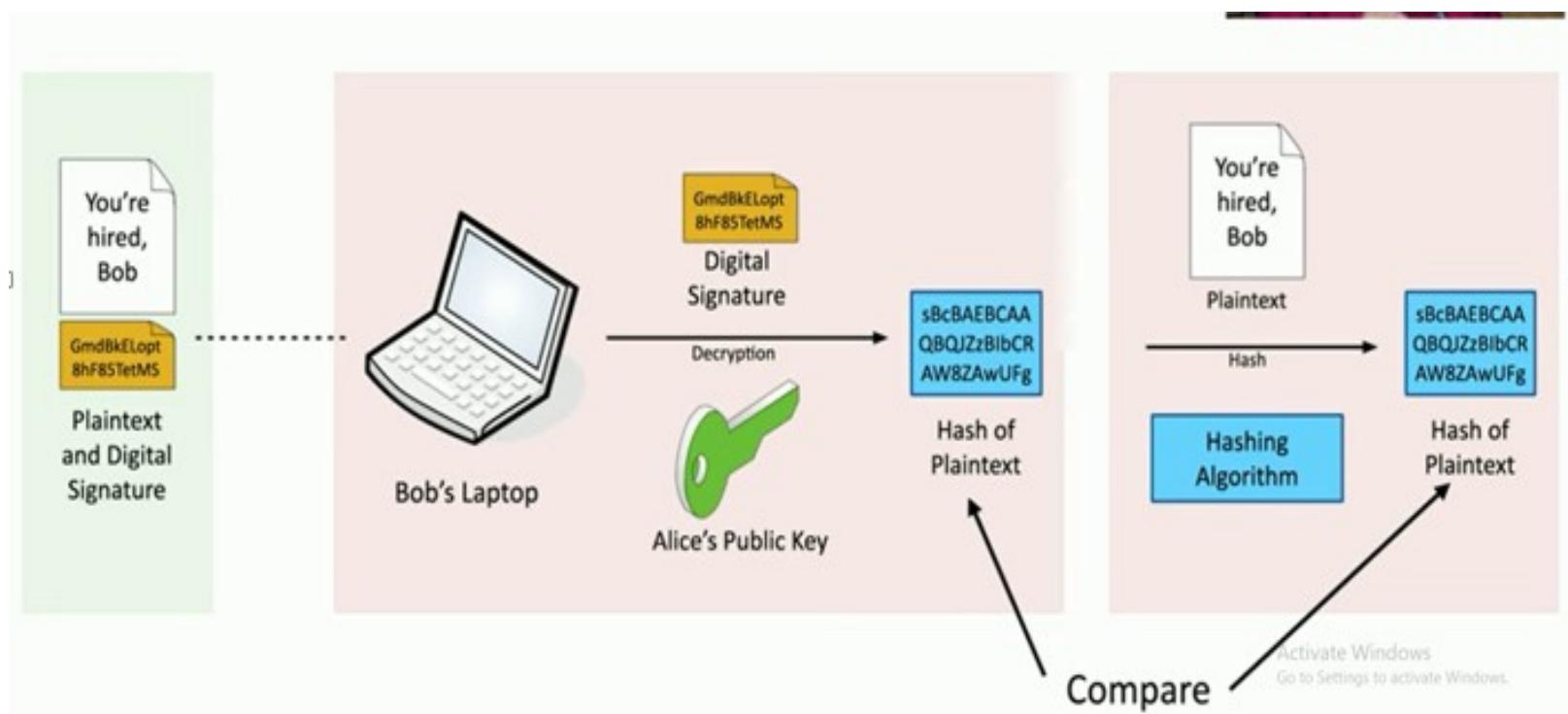
Digital Signatures

- We have looked at message authentication
- But does not address issues of lack of trust
- Digital signatures provide the ability to:
 - Verify author, date and time of signature
 - Authenticate message contents
 - Be verified by third parties to resolve disputes
- Hence include authentication function with Additional capabilities

Digital Signatures



Digital Signatures - Verify the Signature



Digital Signature Requirements

- Must depend on the message signed
- Must use information unique to sender
- To prevent both forgery and denial
- Must be relatively easy to produce
- Must be relatively easy to recognize and verify
- Be computationally infeasible to forge
 - With new message for existing digital signature
 - With fraudulent digital signature for given message
- Be practical to save digital signature in storage

THANK YOU