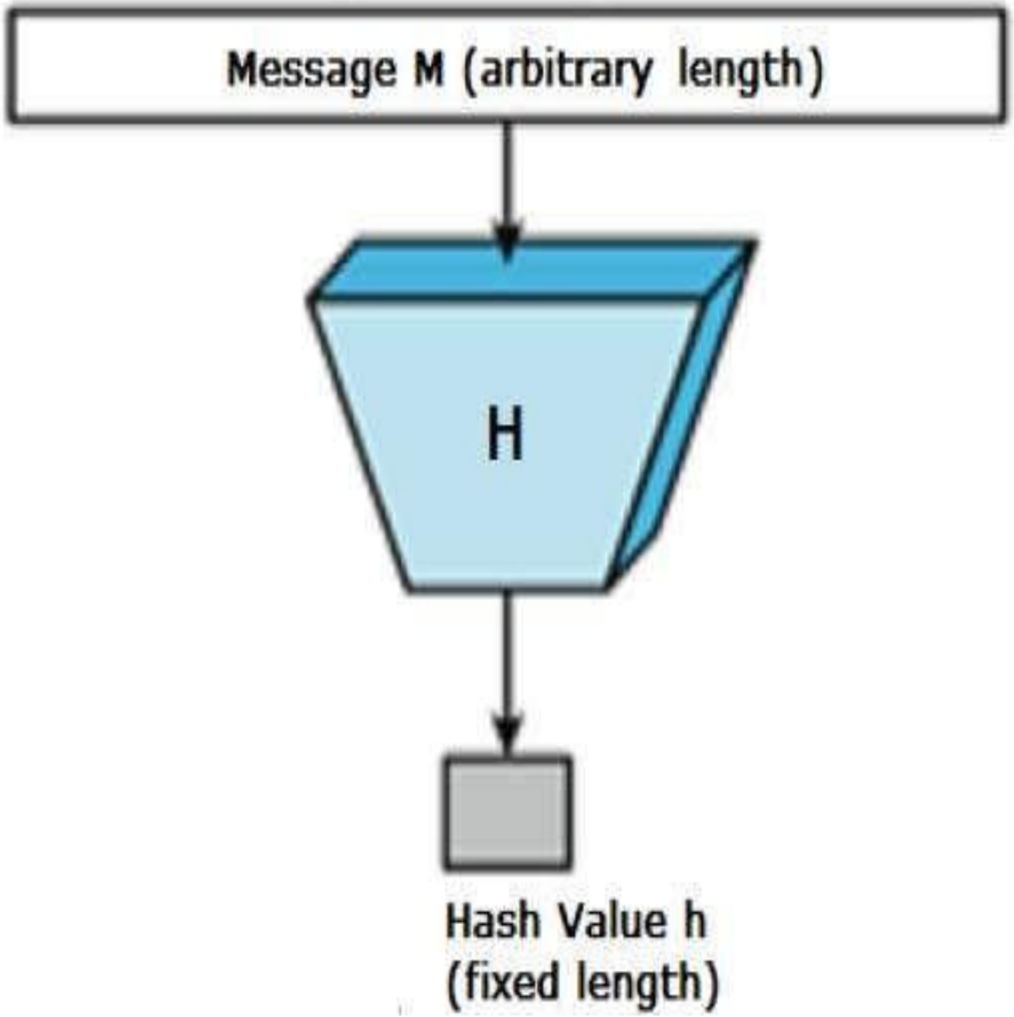
# UNIT 4

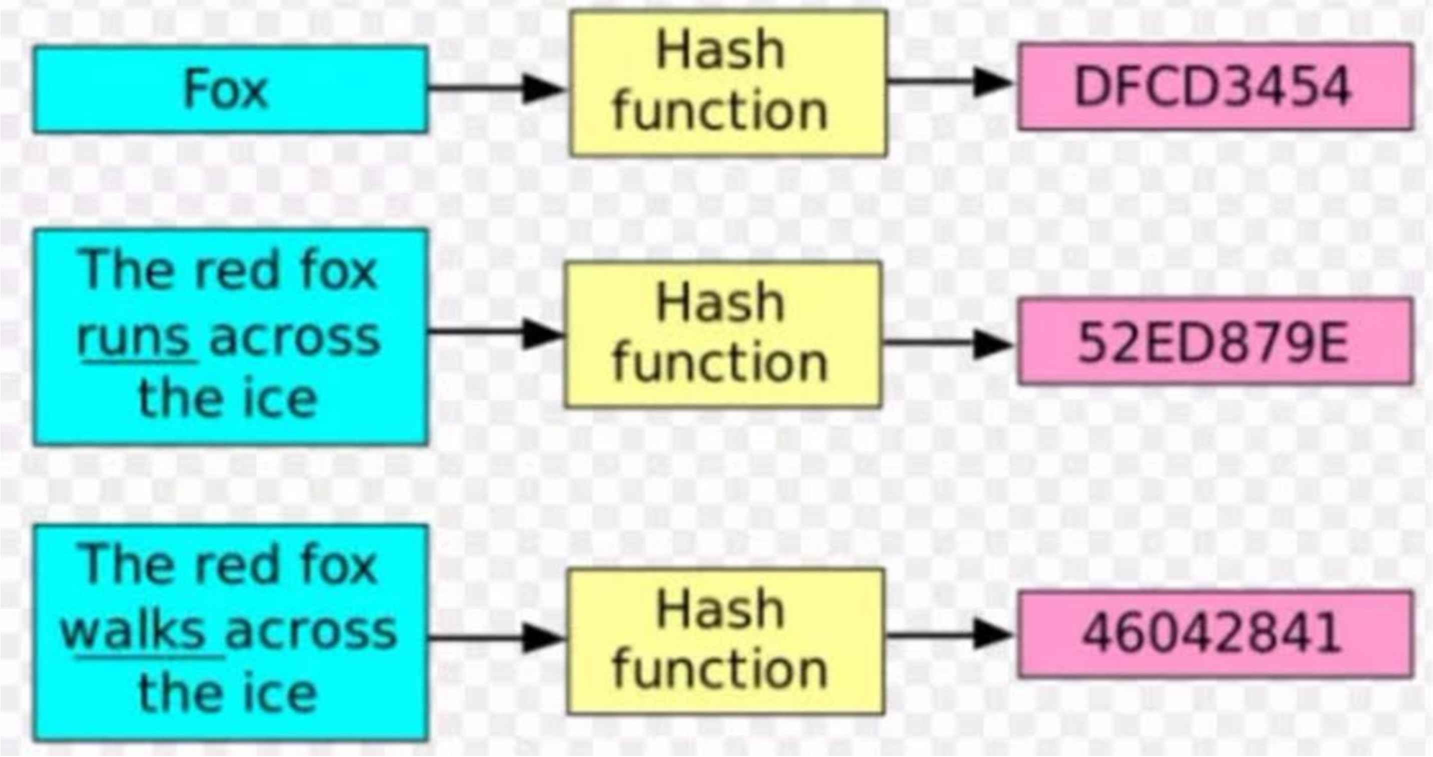
# Hash Function

- Hash function is a mathematical function that converts a numerical input value into another compressed numerical value.
- In hash function, H accept a Variable length block input data called as “M” and produce the fixed size hash value can be represented as

$$h = H(M)$$


Message M (arbitrary length)
H
Hash Value h
(fixed length)

- If any bit or bits changes in the data, then the whole hash function output data will also change.
- Cryptographic hash function is one-way function.
- We cannot get the original message from the hash value.



# SHA

- SHA stands for secure hash algorithm.
- It is the modified version of md5.
- The length of the output is 160 bits
- The purpose of SHA is authentication.

# Working of SHA

1. Padding – the process of adding data to a message before encryption.
2. Appending – adding padding bits to a message to make it a specific length.
3. Initializing

   Here, we should initialize 5 variables.(eg) A,B,C,D & E.

4. Processing blocks

   Changing variables into some corresponding variable.

   A = a, B=b, C=c, D=d and E=e

   Now we should calculate the 5 variables.

# Properties of Hash Function

**Compression :**

The output of the hash function is much smaller than the size of the input.

**Pre-image resistance :**

Difficult to find the input from the given hash function output.

h = H(M)

If h is given, it is difficult to find M.

## Weak Collision Resistance

Given message m1, weak collision resistance means that it is difficult to produce another message m3 such that H(m1) = H(m2).

(i.e) cannot generate the same hash value for 2 different messages.

## Strong Collision Resistance

It means that is difficult to find any two different message that has the same hash value.

# Characteristics of Hash Function

- It is quick to calculate hash value (h) for any given message.
- Hash Function (H) can be applied to variable length of data block.
- A small change in a message (M) should change the hash value.
- The hash function has one-way property.

# MAC

- The full form of MAC is ***Message Authentication Code.***
- Message Authentication Code (MAC) is a cryptographic technique used to verify the authenticity and integrity of a message or a piece of data.
- It is designed to ensure that a message has not been tampered with during transmission and that it was indeed sent by the expected sender.
- MACs use authentication cryptography to verify the legitimacy of data sent through a network or transferred from one person to another.

# How MAC does work

- Here in MAC, sender and receiver share same key.
- Where sender generates a fixed size output called Cryptographic checksum or Message Authentication code and appends it to the original message.
- On receiver's side, receiver also generates the code and compares it with what the received thus ensuring the originality of the message.

These are components:

1. Message
2. Key
3. MAC algorithm
4. MAC value

Let A -> Sender

B -> Receiver

when A sends a message to B, it calculates the MAC as a function of the message and the key.

**MAC = C ( K , M )**

Where

M = Input message

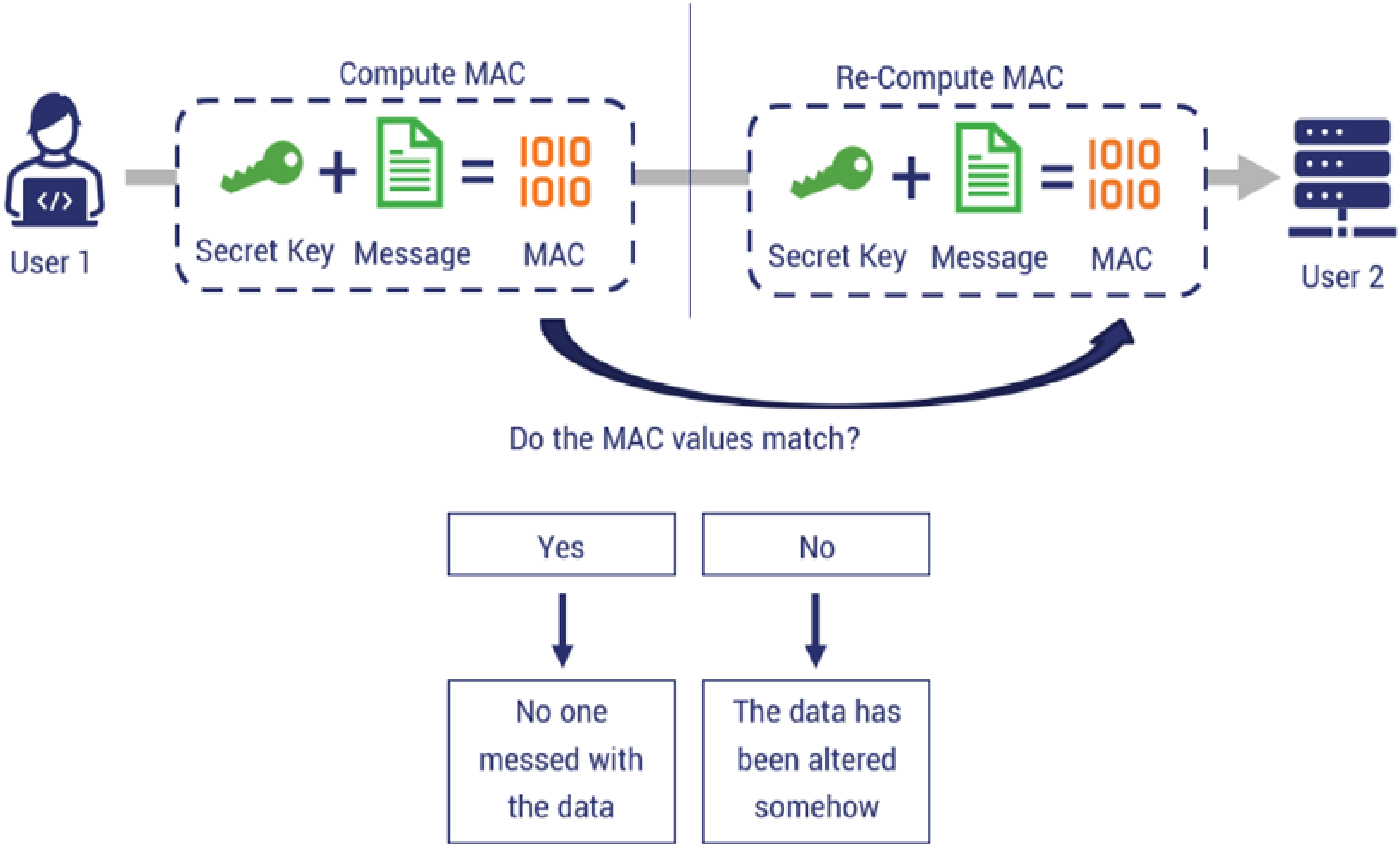
C = MAC function

K = Shared Secret key

Types of MAC

1. One-time MAC
2. Carter-Wegman MAC
3. HMAC
4. CMAC

# How Message Authentication Codes Work



Compute MAC

Secret Key + Message = MAC

Re-Compute MAC

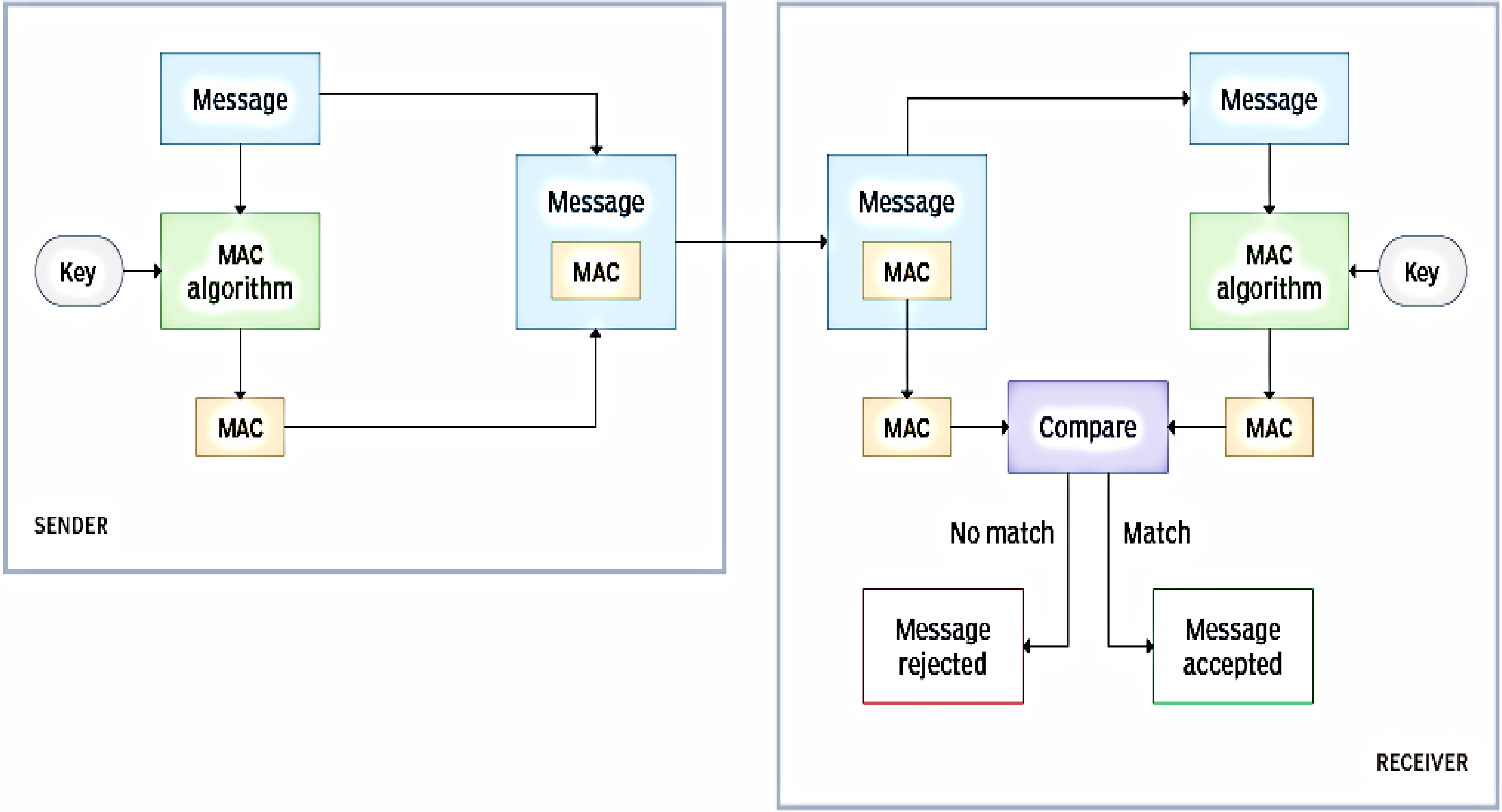
Secret Key + Message = MAC

Do the MAC values match?

Yes → No one messed with the data

No → The data has been altered somehow

# Generating and verifying message authentication codes



# Introduction to MD5

- MD5 algorithm stands for the message-digest algorithm.
- MD5 is a cryptographic hash function algorithm that takes the message as input of any length and changes it into a fixed-length message of 16 bytes.
- MD5 was developed as an improvement of MD4, with advanced security purposes.
- The output of MD5 (Digest size) is always 128 bits.

# Algorithm

As we all know that MD5 produces an output of 128-bit hash value.

This encryption of input of any size into hash values undergoes 5 steps, and each step has its predefined task.

- **Step1: Append Padding Bits**
- **Step2: Append Length**
- **Step3: Initialize MD buffer**
- **Step4: Processing message in 16-word block**
- **Step5: Output**

# Working of MD5

- MD5 is quite fast than other versions of the message digest, which takes the plain text of 512-bit blocks, which is further divided into 16 blocks, each of 32 bit and produces the 128-bit message digest, which is a set of four blocks, each of 32 bits.
- MD5 produces the message digest through five steps, i.e. padding, append length, dividing the input into 512-bit blocks, initializing chaining variables a process blocks and 4 rounds, and using different constant it in each iteration.

# Padding

Suppose we are given a message of 1000 bits. Now we have to add padding bits to the original message. i.e. 64 bits less than an exact multiple of 512 (i.e. 512*3 = 1536). After adding the padding bits the size of the original message/output of the first step will be 1536 – 64 = 1472. 1472. we will add 472 padding bits to the original message.

# Appending

- In this step, we add the length bit in the output of the first step.
- Where, length bits = 64 and we add the length bit in the output of the first step. i.e. output of first step = 512 * n – 64. After adding both we will get **512 * n** i.e. the exact multiple of 512.

# PROCESS EACH 512-BIT BLOCK:

This is the most important step of the MD5 algorithm.

Here, a total of 64 operations are performed in 4 rounds each round consist of 16 operation.

# OUTPUT

After all rounds had performed, the buffer A, B, C, D,E contains the MD5 output starting with lower bit A and ending with higher bit D.

# SHA ALGORITHM

- SHA Stands for Secure Hashing Algorithm.
- SHA modified version of MD5 and used for hashing information and certificates.
- A hashing algorithm shortens the input data into a smaller form that cannot be understood by using bitwise operations, modular additions, and compression functions.
- The output of SHA is a message digest of 160 bits in length.
- This is designed to be computationally infeasible.

# Introduction

- The Secure Hash Algorithm (SHA) was invented by the National Security Agency (NSA) and published in 1993 through the National Institute of Standard and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS).
- It has following versions

SHA-0

SHA-1

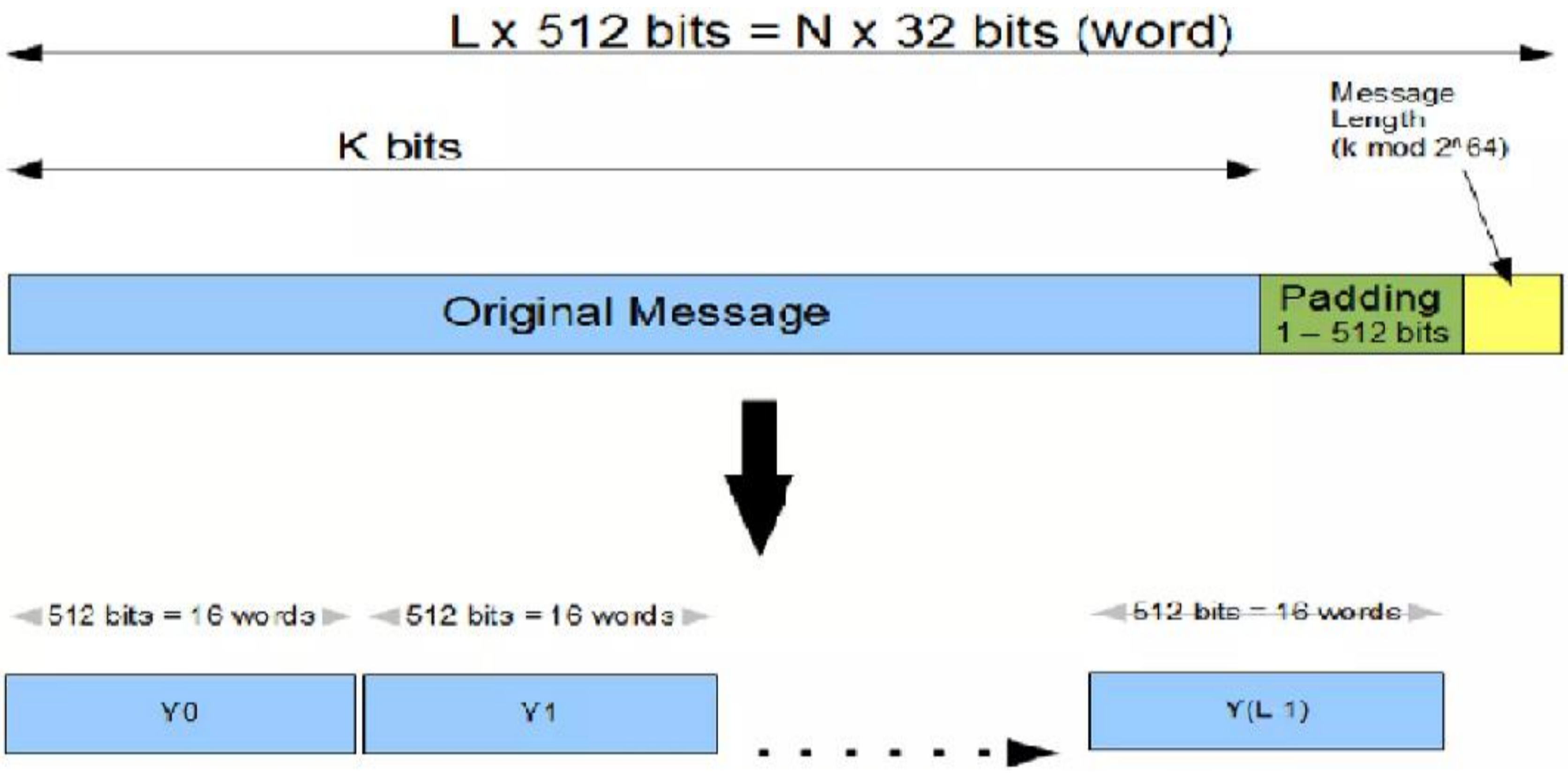
SHA-224

SHA-256

SHA-512

# There are 5 steps in this Algorithm:

- Step 1: Padding of bits
- Step 2: Append Length
- Step 3: Divide the input into 512-bit blocks
- Step 4: Initialize
- Step 5: Process Blocks.



A=a B=b C=c D=d E=e

160 bits 160 bits 160 bits 160 bits 160 bits

## 1) Padding:

- Original message + Padding = Length
- So that total length is 64 bit less than exact multiple of 512

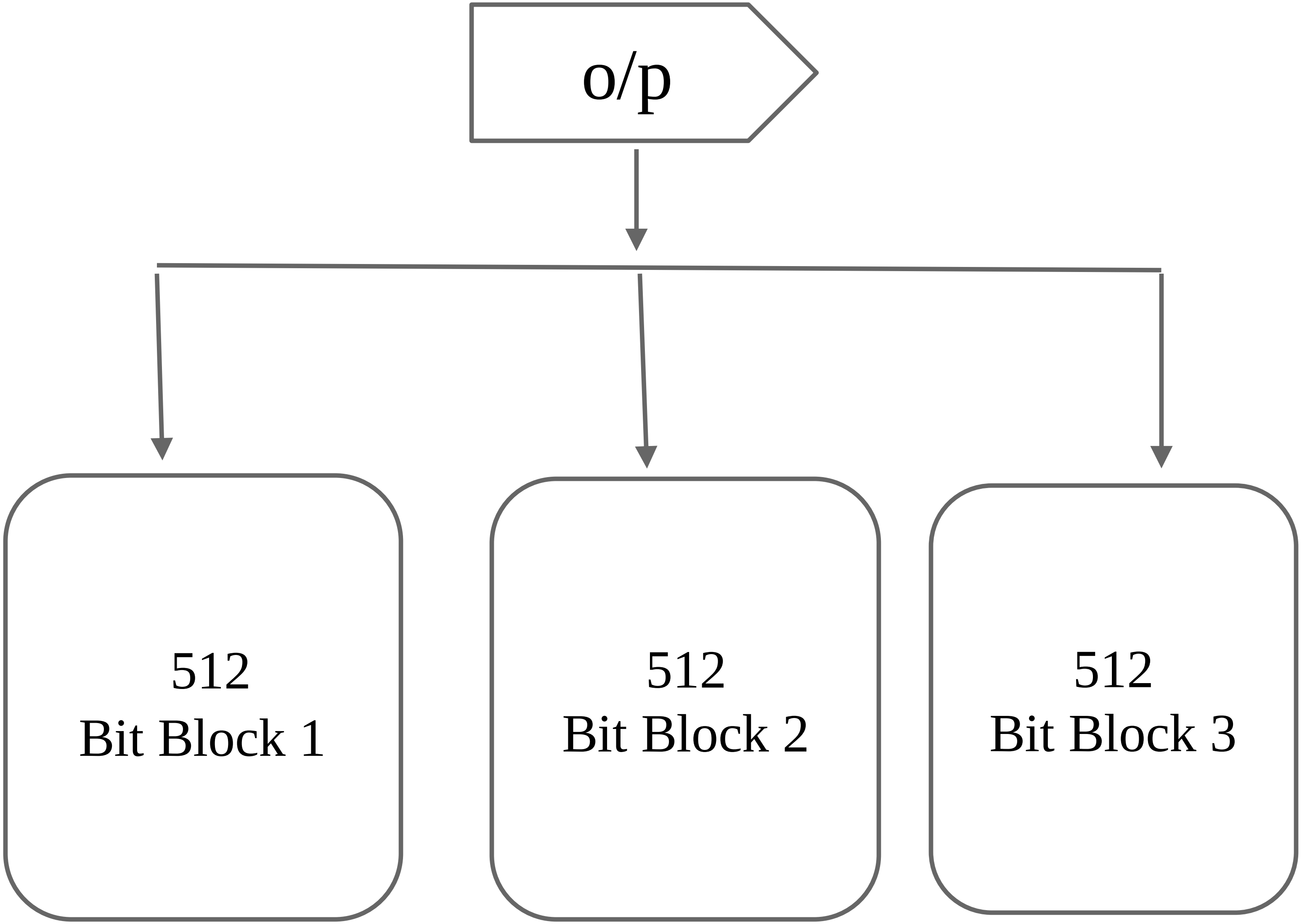
Example:

- Original message =1000bits
- 512x1 = 512 bits
- 512x2= 1024 bits
- 512x3=1536 bits.
- 1536 -64 1472
- 1000bits+472 bits =1472 bits.

# (2) Appending:

- Append the original length before padding.
- Calculate length. In most of the cases, 64bits is obtained as answer(append 64 bits).
- So it again becomes multiple of 512

# (3)Dividing



o/p

512
Bit Block 1

512
Bit Block 2

512
Bit Block 3

**(4)** Initialize

- Initialize 5 variables. (A,B,C,D and E)

**(5)** Process Blocks-Now the actual algorithm begins.

- Copy responding variables
- A=a, B=b, C=C, D=d, E=e
- Divide into no. of 512 bit blocks.
- Four Rounds (each round=20steps).

# Application

- Secure password hashing
- Secure Socket Layer(SSL) security protocol
- Digital signature

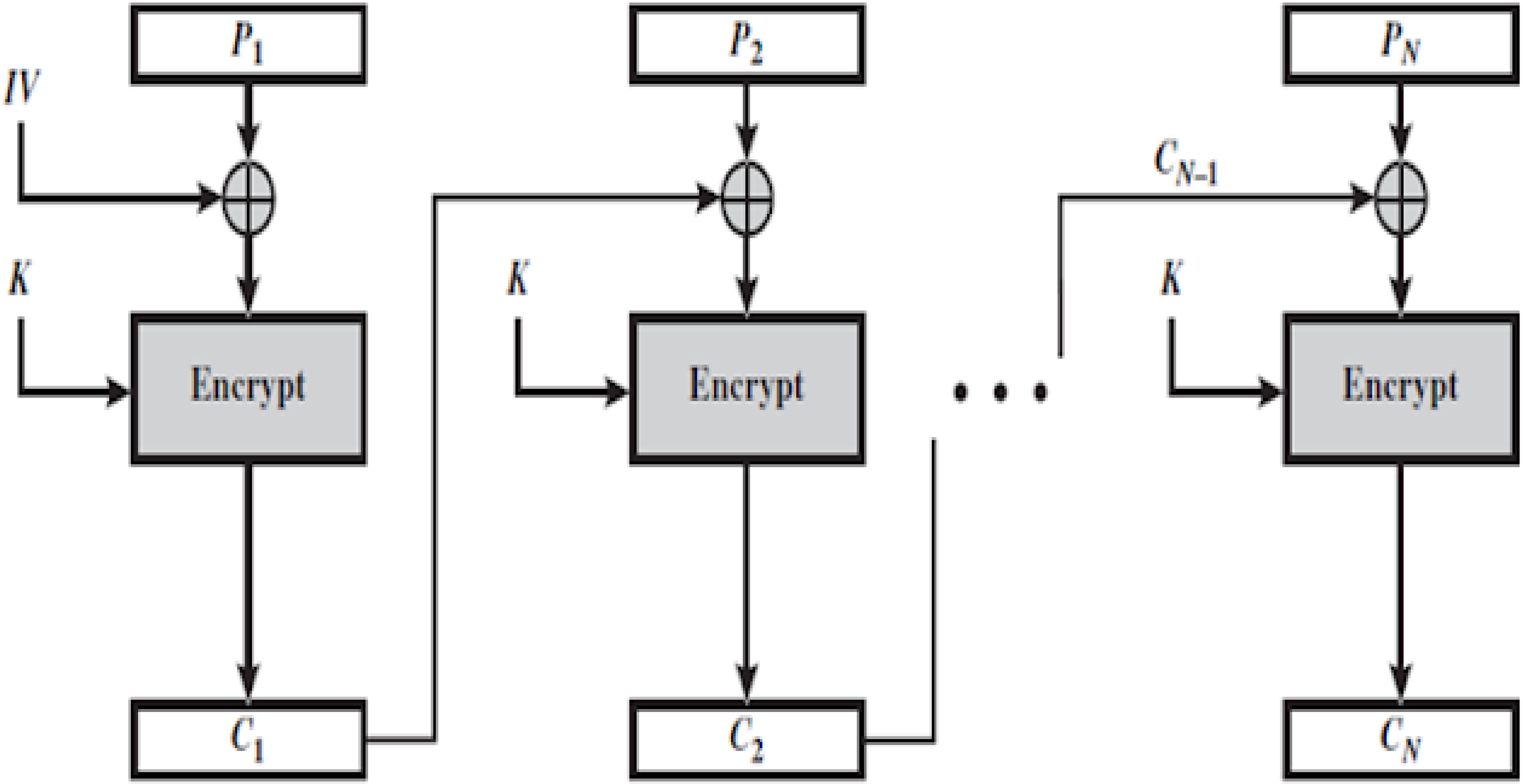
# Hash Function Based On Block Cipher Chaining

## Block cipher:

- Block cipher is a popular encryption and decryption primitives.
- The block cipher processes fixed-size blocks simultaneously,  as opposed to a stream cipher, which encrypts data one bit at a time.

To encrypt, the block cipher accepts a key K and a plain text block P as input and produces a cipher text block

$$C = E(K, P), \text{ also written as } C = E_k(P)$$



- The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each.

- If necessary,  the final block is padded to bits.

- The final block also includes the value of the total length of the input to the hash function.

- The inclution of the length makes the job of the opponent more difficult.

# MAC algorithms

1. **Hash-based Message Authentication Code.** HMAC is a cryptographic authentication technique that uses a secret key in conjunction with a hash function approved by the Federal Information Processing Standards (FIPS). Because different hash functions can be used, there are multiple implementations of HMAC, such as HMAC-SHA256 and HMAC-SHA3-256. Multiple communication and transfer protocols use HMAC, including Hypertext Transfer Protocol Secure (HTTPS), File Transfer Protocol Secure (FTPS) and Secure File Transfer Protocol (SFTP).

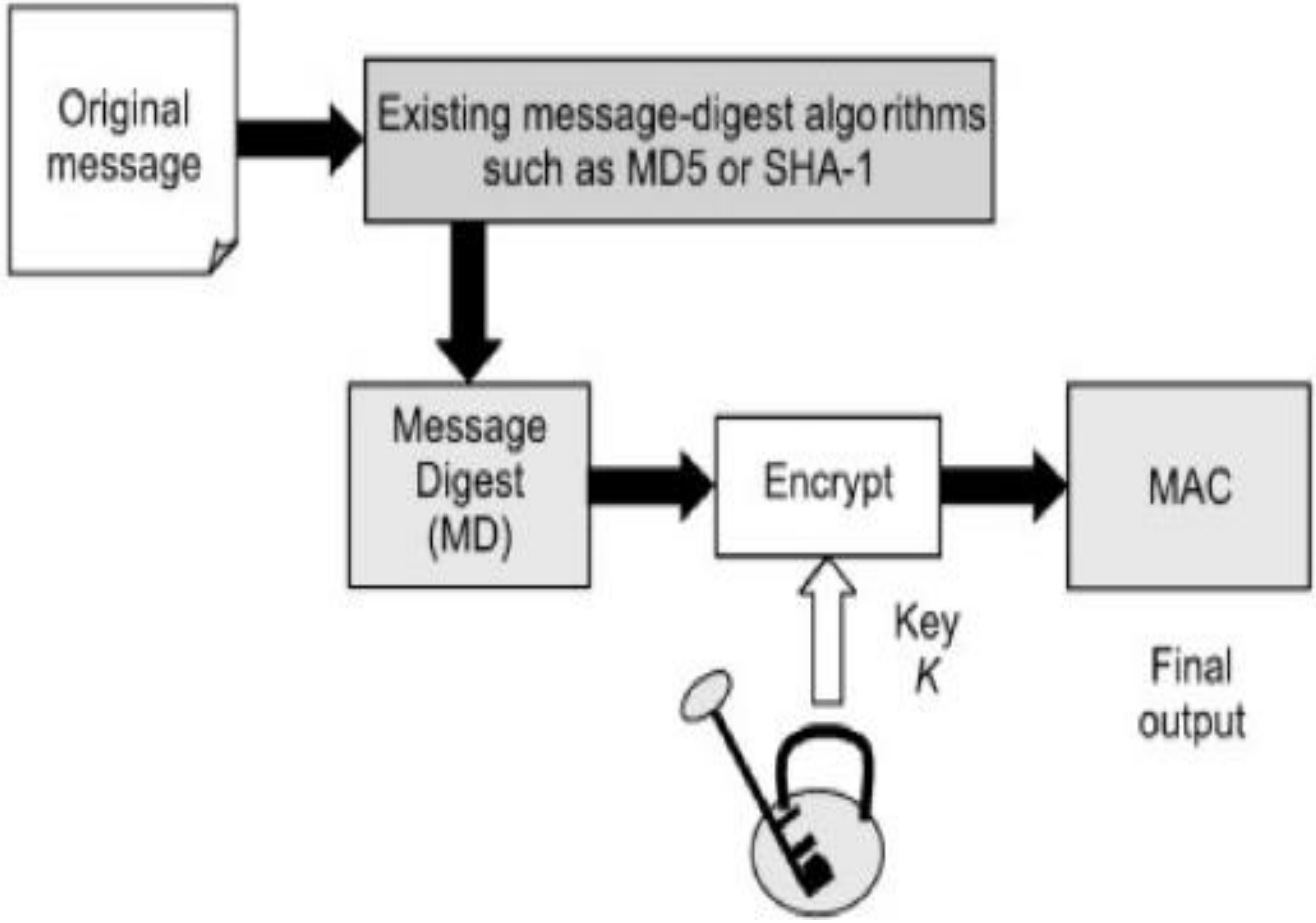
2. **Cipher-based Message Authentication Code**. CMAC standard defines a block cipher-based MAC algorithm for ensuring authenticity and integrity. According to NIST, CMAC can be considered a mode of operation of the block cipher, providing an "algorithm for the cryptographic transformation of data that features a symmetric key block cipher."

# HMAC - Hash Message Authentication Code

# HMAC

- HMAC stands for - Hash Message Authentication Code.
- Mandatory for security implementation for Internet Protocol Security.
- HMAC is to reuse existing message digest algorithms (such as MD5,SHA-1...)
- Uses shared symmetric key to encrypt message digest.

# HMAC CONCEPT



# WORKING OF HMAC

## Variables used in HMAC

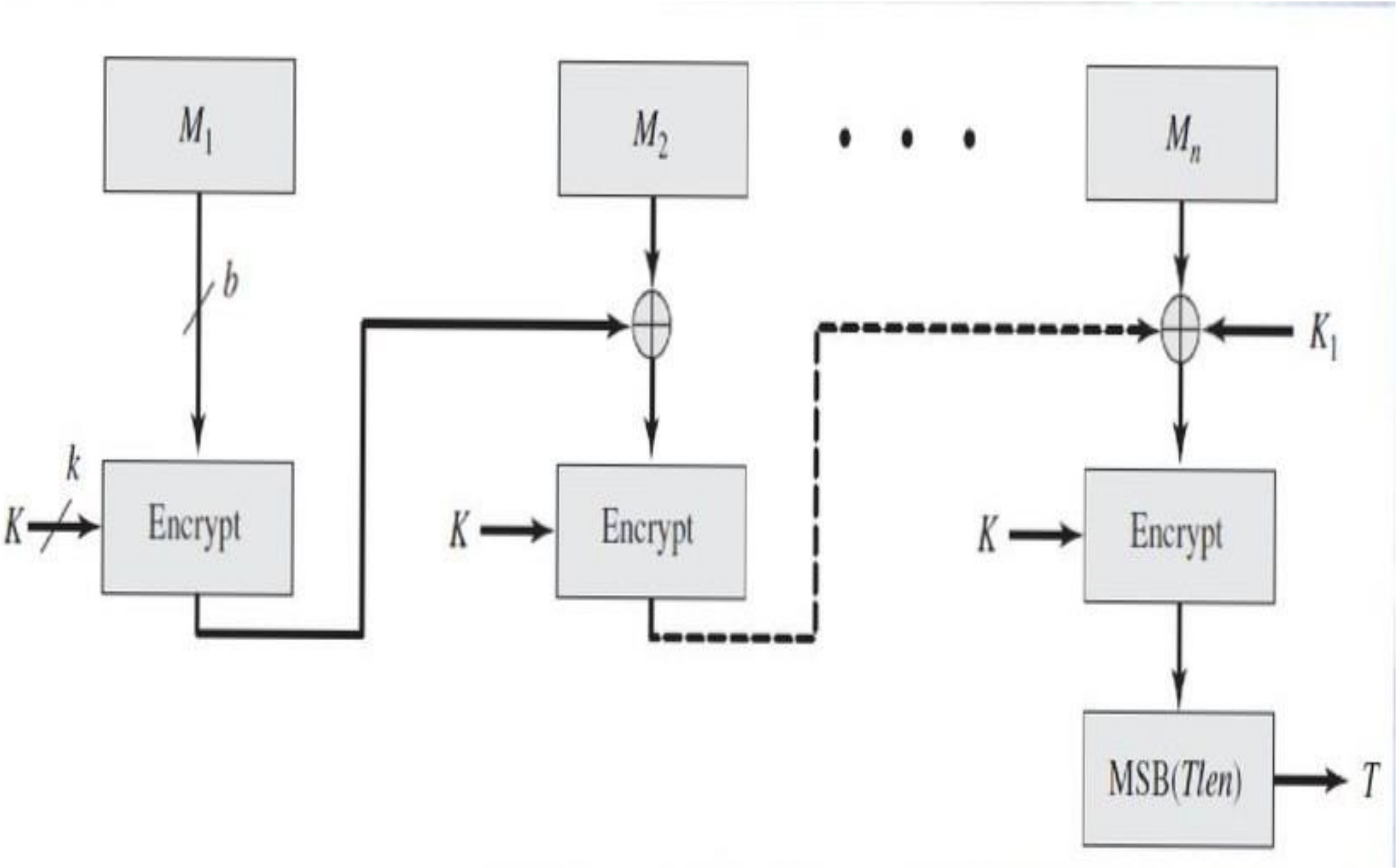
- MD - The message digest hash function used (eg, MD5)
- M - the input message whose MAC is to be calculated.
- L - the Length of blocks in the message M.
- K - the shared symmetric key to be used in HMAC.
- B - the numbers of bits in each block.
- Input Padding (IPad) - a string repeated by b/8 times.
- Output Padding (Opad) - a string repeated by b/8 times.

# CMAC-
# Cipher Based MAC

# Cipher-Based Message Authentication code (CMAC)

- CMAC is a message authentication code based on AES and triple DES.
- Message length is integer multiple of block size :
- First, let us consider the operation of CMAC when the message is an integer multiple n of the cipher block length b.
- The message is divided into n blocks (M1,M2,...,Mn).

# Diagrammatic Representation Of CMAC



# Calculation Of CMAC

Step 1: C1 = E(K,M1)

Step 2: C2= E(K,[M2(+)C1])

Step 3: C3=E(K,[M3(+)C2])

. . . .

. . . .

. . . .

Step n: Cn=E(K,[Mn(+)Cn-1)

T=MSBtlen(Cn)

C = Cipher Text
E = Encryption
M = Message
K = Key generation
T = to Measure Length

# KEY WRAPPING

# What is key wrapping?

- Key wrapping is a cryptographic technique used to protect and securely transmit cryptographic key material. It involves encrypting a cryptographic key with another key or keys, known as the wrapping key or keys.
- Key wrapping may rely on either symmetric or asymmetric cryptography, depending on the context.
- The primary purpose of key wrapping is to safeguard the confidentiality, integrity, and authenticity of cryptographic keys, especially when they are transmitted or stored in potentially insecure environments.

# Key Wrapping Procedure

**Step 1** - Select Wrapping Key: Start with the cryptographic key that you want to protect. This could be a symmetric encryption key, asymmetric private key, or any other sensitive cryptographic key.

**Step 2** - Select or generate a Key Encryption Key (KEK). This is a separate cryptographic key used specifically for encrypting and decrypting other keys. The KEK can be symmetric or asymmetric, depending on the security requirements and the encryption algorithm being used.

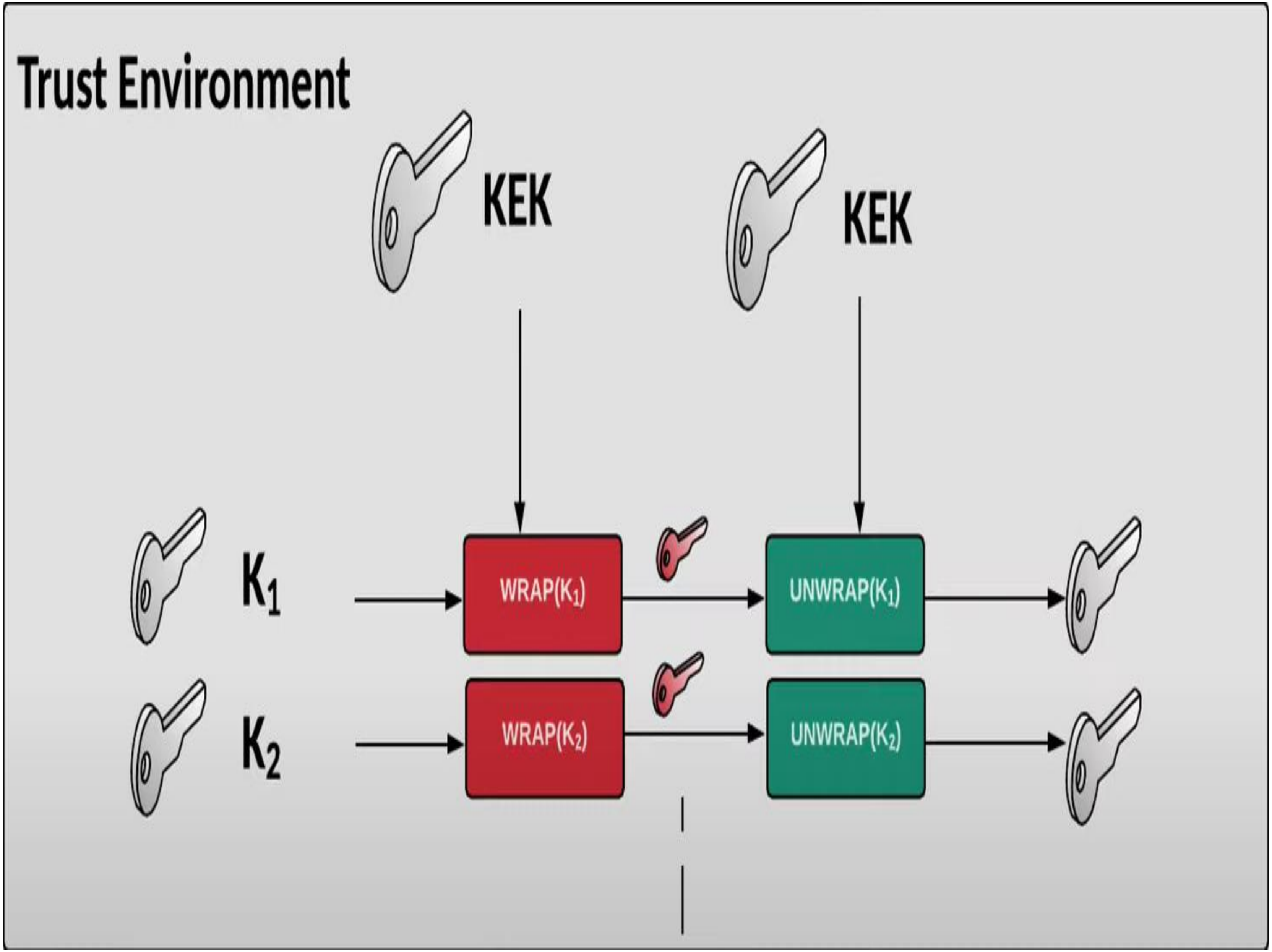
**Step 3** - Use the KEK to encrypt the key that you want to protect. This process involves applying a cryptographic algorithm (such as AES Key Wrap or RSA-OAEP) to securely encapsulate the key with the KEK. The result is a wrapped (encrypted) version of the original key.

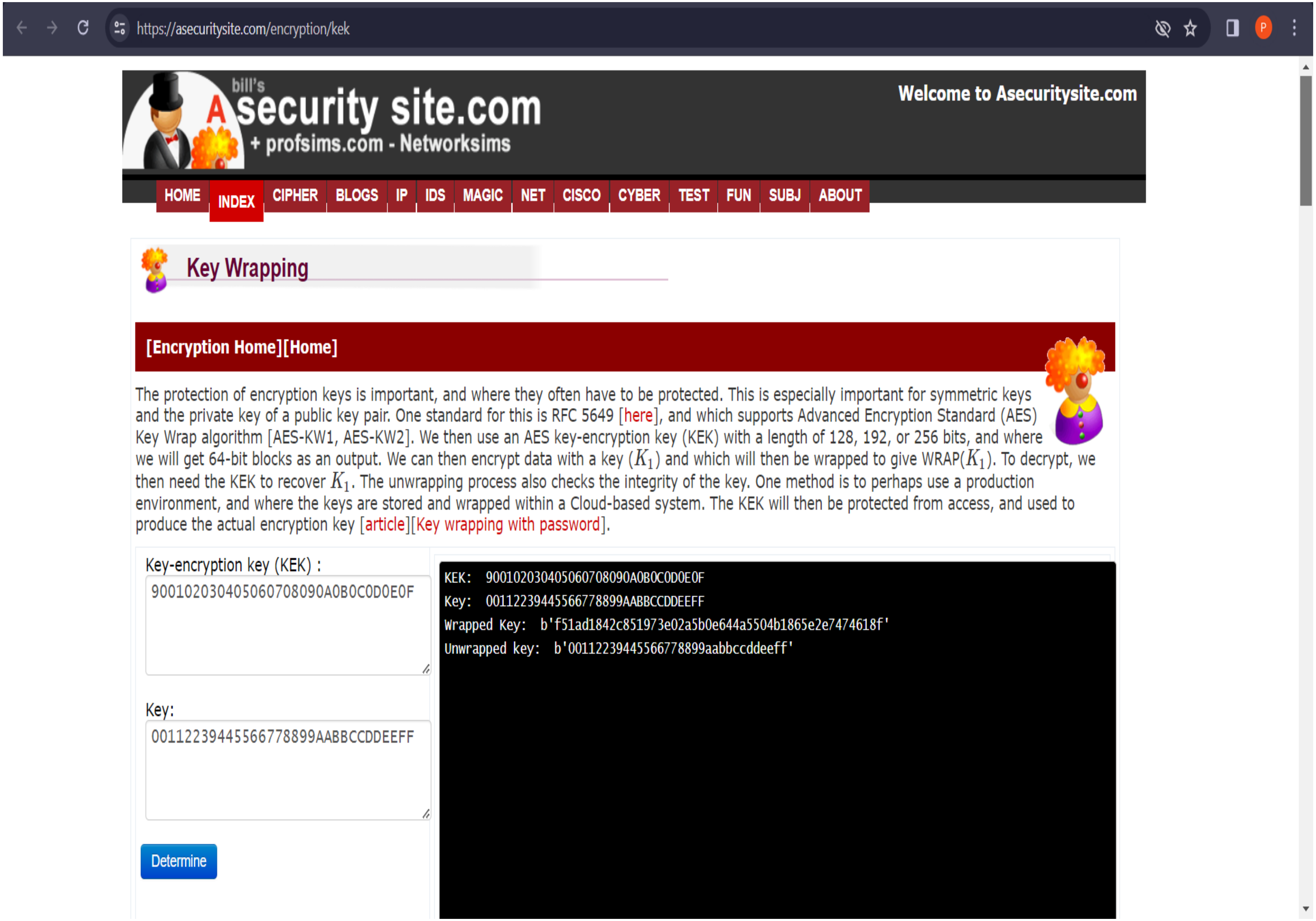
# Key Wrapping Procedure

**Step 4** - Safely store or transmit the wrapped key as needed. Since it is encrypted with the KEK, the wrapped key is protected and can only be decrypted by someone who has access to the corresponding KEK.

**Step 5** - When the wrapped key needs to be used, it can be decrypted (unwrapped) using the appropriate decryption key (KEK). This process involves applying the inverse cryptographic algorithm to securely recover the original key from the Wrapped version..

**Step 6** - Once the wrapped key is successfully decrypted, you have the original key that can be used for cryptographic operations, such as data encryption or decryption.


Trust Environment
KEK
KEK
$K_1$
WRAP($K_1$)
UNWRAP($K_1$)
$K_2$
WRAP($K_2$)
UNWRAP($K_2$)


https://asecuritysite.com/encryption/kek
bill's
A security site.com
+ profsims.com - Networksims
Welcome to Asecuritysite.com
HOME
INDEX
CIPHER
BLOGS
IP
IDS
MAGIC
NET
CISCO
CYBER
TEST
FUN
SUBJ
ABOUT

## Key Wrapping

[Encryption Home][Home]

The protection of encryption keys is important, and where they often have to be protected. This is especially important for symmetric keys and the private key of a public key pair. One standard for this is RFC 5649 [here], and which supports Advanced Encryption Standard (AES) Key Wrap algorithm [AES-KW1, AES-KW2]. We then use an AES key-encryption key (KEK) with a length of 128, 192, or 256 bits, and where we will get 64-bit blocks as an output. We can then encrypt data with a key ($K_1$) and which will then be wrapped to give WRAP($K_1$). To decrypt, we then need the KEK to recover $K_1$. The unwrapping process also checks the integrity of the key. One method is to perhaps use a production environment, and where the keys are stored and wrapped within a Cloud-based system. The KEK will then be protected from access, and used to produce the actual encryption key [article][Key wrapping with password].

Key-encryption key (KEK) :

900102030405060708090A0B0C0D0E0F

Key:

00112239445566778899AABBCCDDEEFF

```
KEK:  900102030405060708090A0B0C0D0E0F
Key:  00112239445566778899AABBCCDDEEFF
Wrapped Key:  b'f51ad1842c851973e02a5b0e644a5504b1865e2e7474618f'
Unwrapped key:  b'00112239445566778899aabbccddeeff'
```