Chapter Five

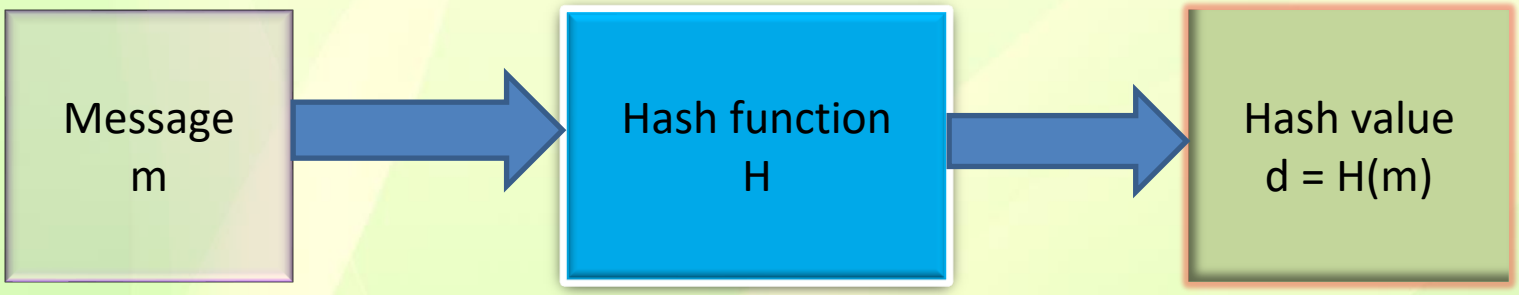
Cryptographic Hash Function

Cryptographic Hash Function

A cryptographic hash function, H is a function that transforms a string (value) of any length to a fixed length string (short value).

Input string is an original text or message and output is a hash value.

Message to hash value



The hash value is also termed as ***message digest***, ***digest***, ***checksum*** or ***digital fingerprint***. If *m* be a message and H be a hash function, the hash value, d = H (*m*).

The properties of Hash function

A Cryptographic hash function should have the following properties:

1. *m* can be of any length
2. H (*m*) is relatively easy to compute for any given *m*.
3. H( ) is one-way.
4. H( ) should be collision-free.
5. The output of H( ) will be fixed length.

Properties of hash function

One way function: Given a hash value *d*, it is **extremely difficult** to find the input value *m from hash value d*, where *H (m)* = *d*, then *H* is called a one-way hash function.

Collision free: Given an input m_1, and another input m_2 (where $m_1 \neq m_2$). Now we compute

d_1 = H(m1) and d_2 = H(m2).

1) If $d_1 \neq d_2$, it means the function, H is collision free.
2) If $d_1 = d_2$, it means that the function, H is not collision free.

Cryptographic hash function

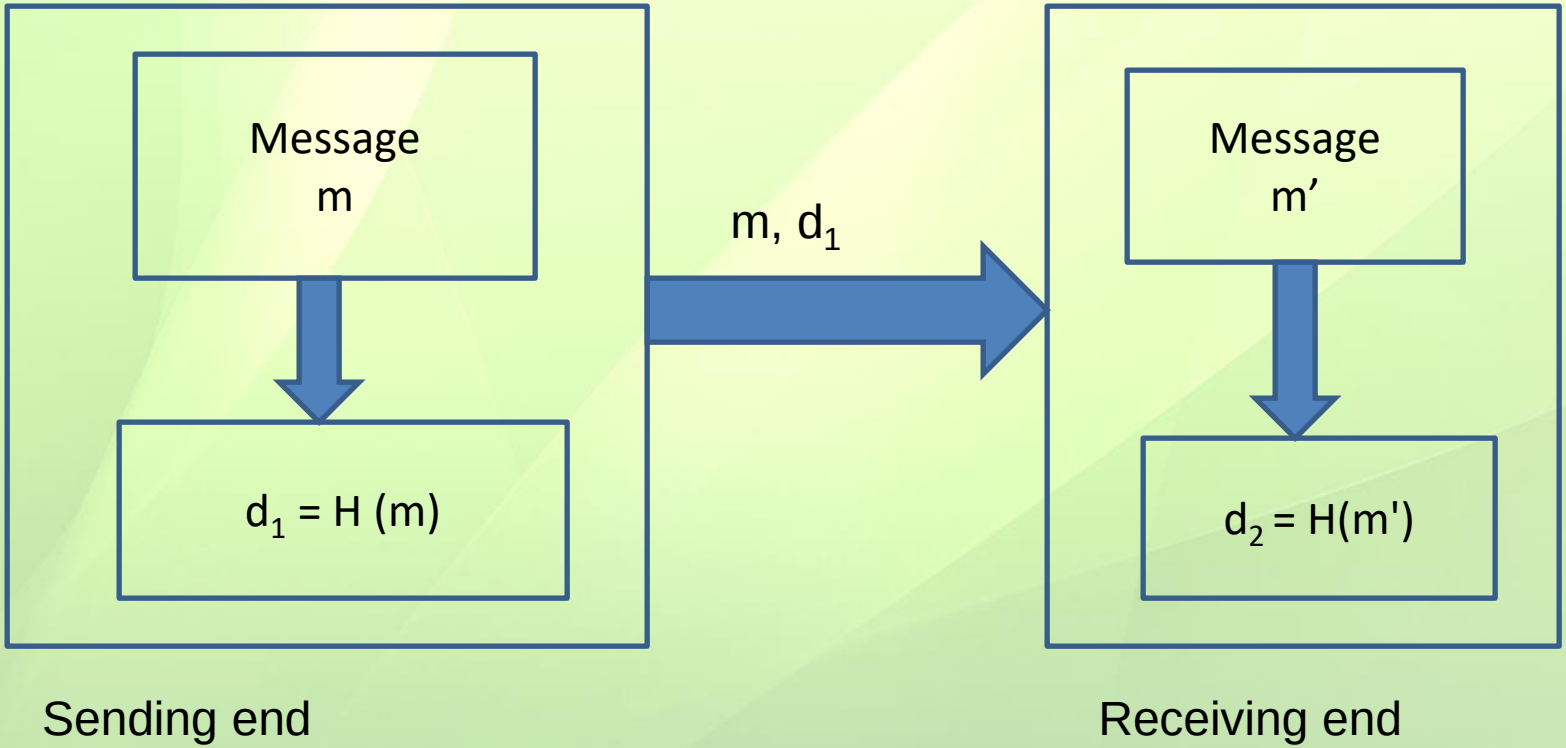
A cryptography hash function is considered *insecure* if either of the following is computationally feasible:

1. To find an input string from a hash value.

2. To find “collision” (the case where two different messages give the same hash value).

Application of Hash function

Integrity check:



Integrity check

If $d_1 = d_2$, then m = m'.

So, the message has been received in original form.

Otherwise the message has been modified.

Application of Hash function

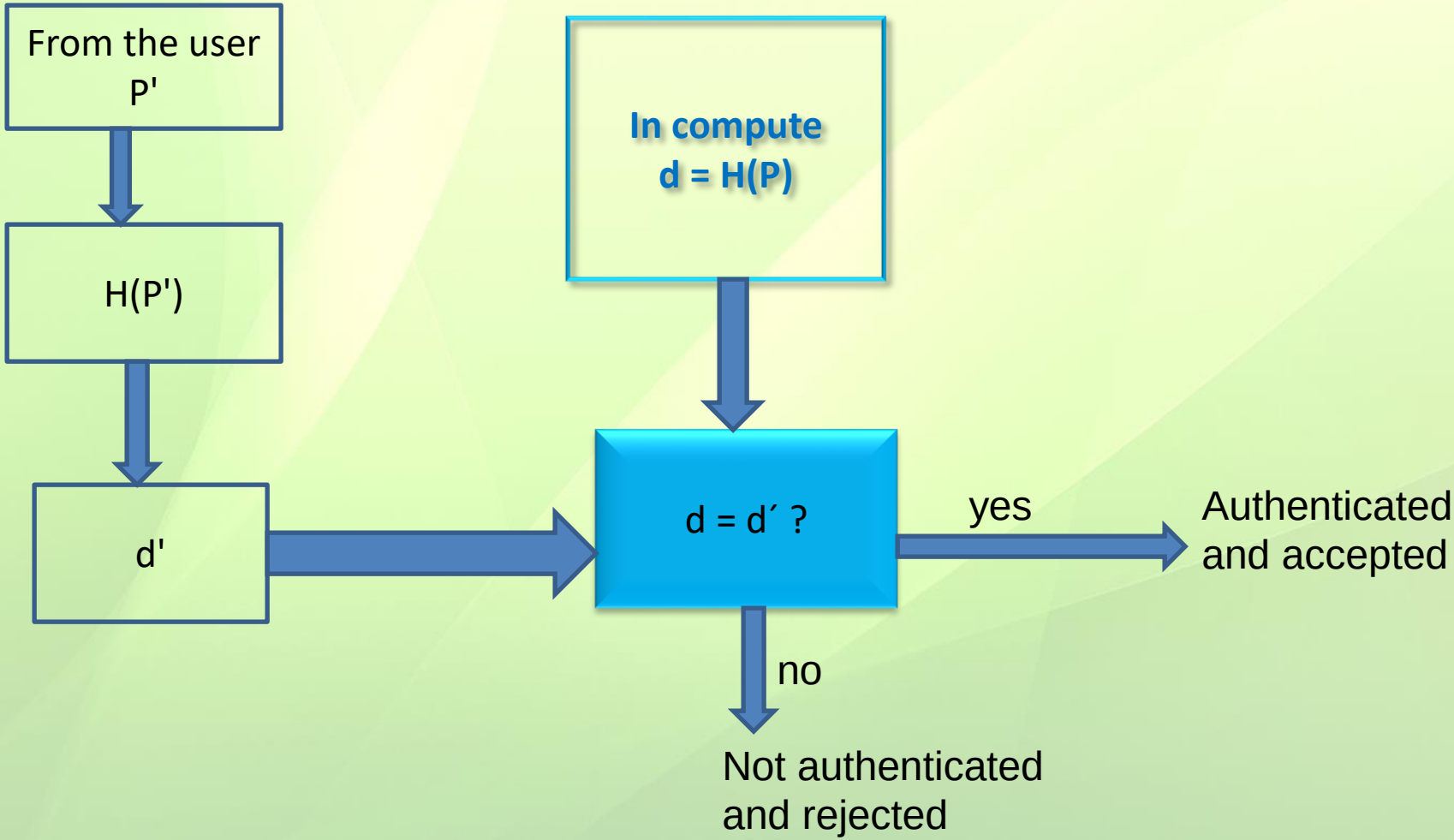
A related application of a cryptographic hash function is ***password* authentication or *verification*.**

Passwords are usually not stored in plaintext, but in encrypted or digest form.

If the passwords are stored in **digest forms**, to authenticate users.

The password of the user is hashed and compared with the stored hash or digest. If the results are same, the user has been authenticated. The above process sometimes referred to as *one-way encryption*.

Password authentication



Application of hash function in digital signature

A sends signed message to *B*.

A computes:

1. d = H(m)
2. s = $E(K_{p,\,A}, d)$

A sends (m, s) to *B*.

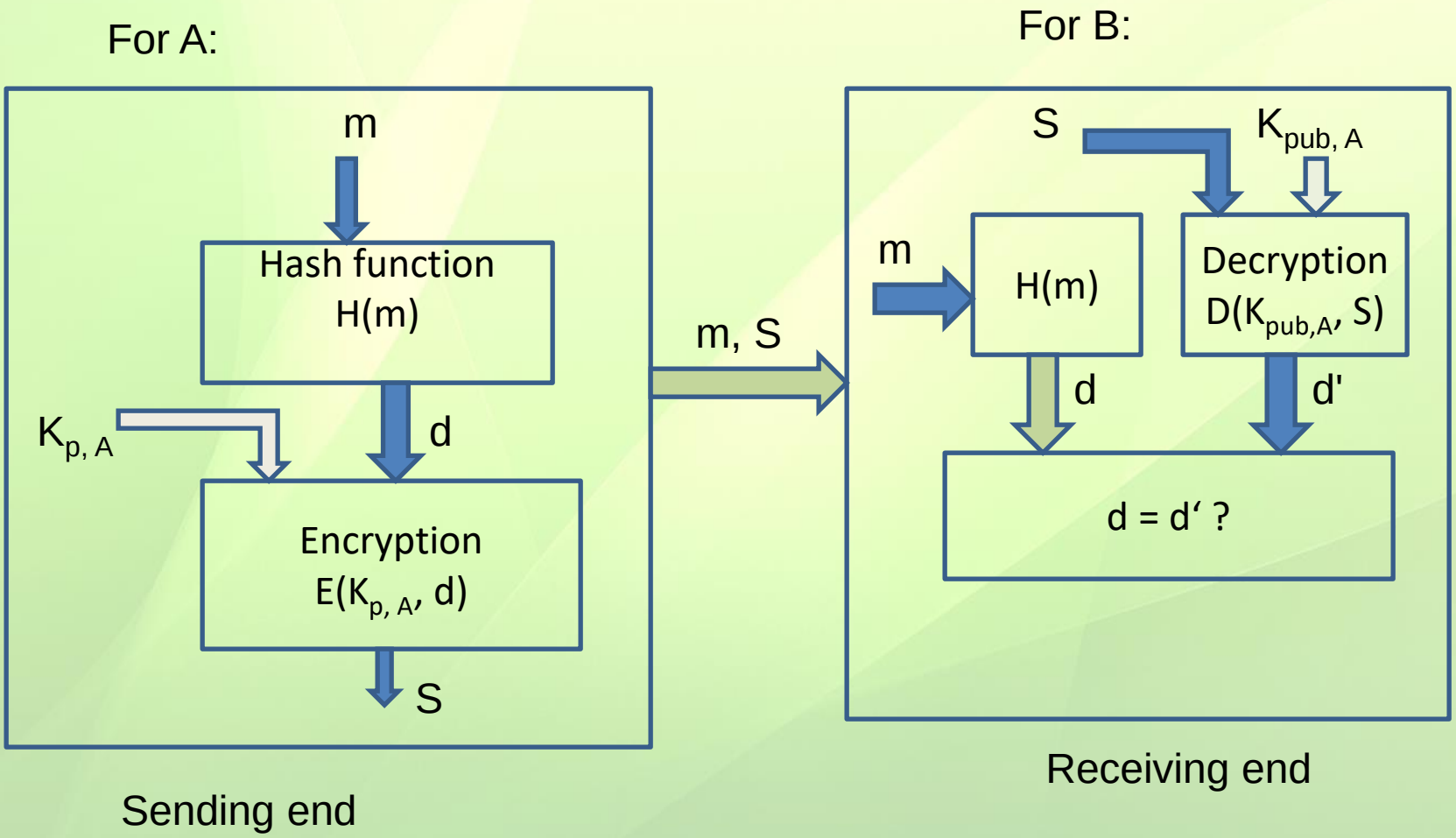
B receives (m, s) and computes:

1. d = H(m)
2. d' = $D(K_{pub,\,A}, s)$

Verification:

If d = d', so s is the signature of *A*. In other words, *A* signed the message m.

Application of Hash function in digital signature



Integrity and authenticity verification

If **d = d',** then m is received by B in original form and m is signed using S by A. Thus the integrity of the message is verified and S is the signature of A means the message is come from authentic source.

Hash function in Practice

Some well-known and widely used functions are briefly described below:

SHA-1 (Secure Hash Algorithm) SHA-1 is a cryptographic hash algorithm published by the United States Government. It takes an arbitrary length string as an input and produces a **160 bit hash value**. SHA-1 is still considered as adequately safe for practical use.

Hash function in Practice

The stronger versions of **SHA-1** are **SHA-256**, **SHA-384**, and **SHA-512**, which produce 256-bit, 384-bit and 512-bit hash values, respectively.

MD5 (Message Digest Algorithm 5):

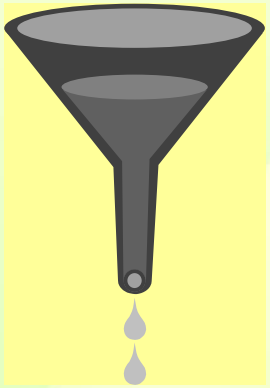
It is a cryptographic hash algorithm developed at **RSA Laboratories** in USA. The previous algorithms similar to MD5 are MD2 and MD4 have been broken.

Hash function in practice

Even **there are some concerns about the safety of MD5** as well and as a precaution the use of MD5 is not recommended in new applicaitons. **MD5 is commonly used to verify the integrity of files.**

MD5: Message Digest Version 5

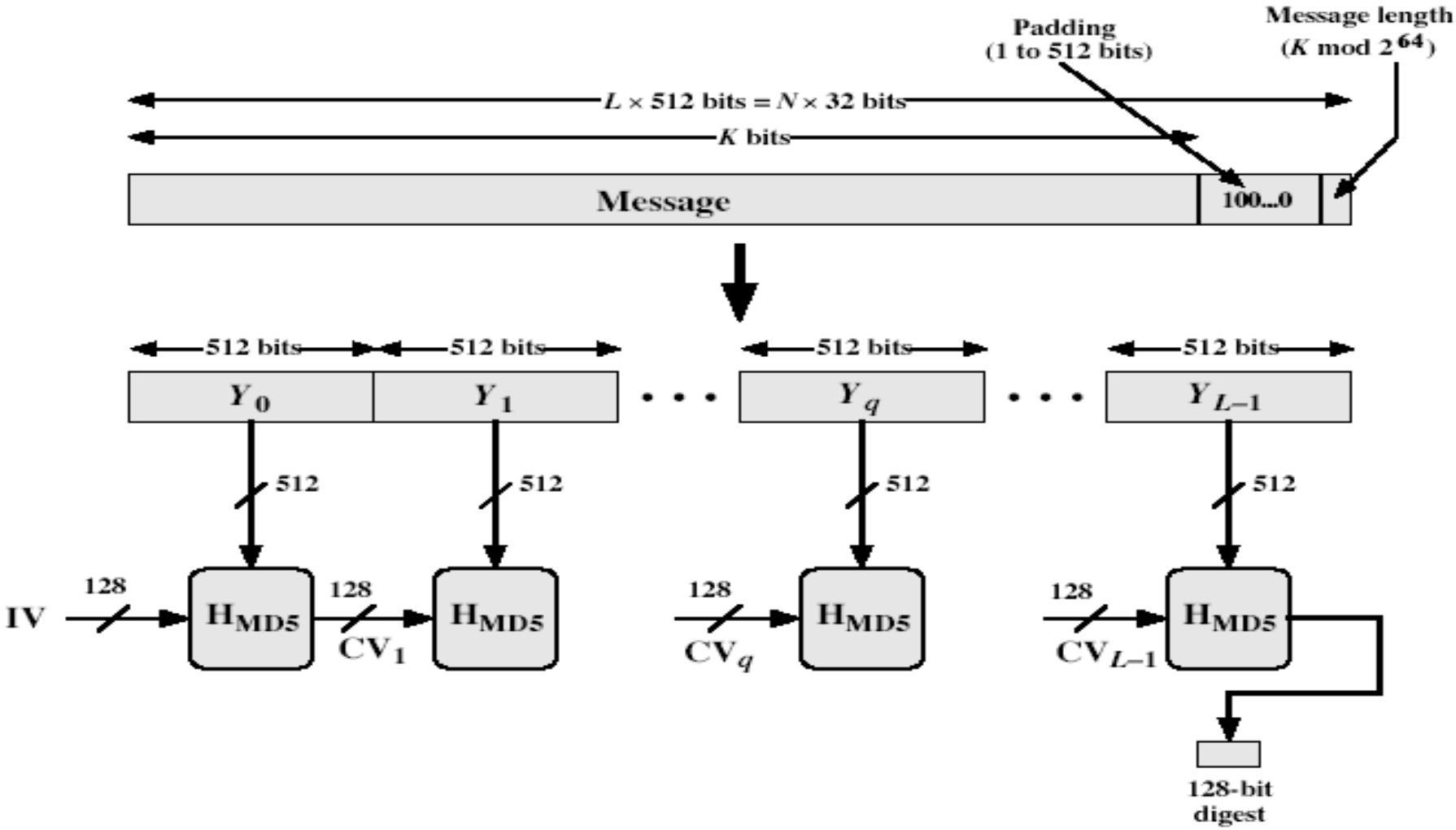
input Message



Output 128 bits Digest

- Until recently the most widely used hash algorithm

MD5 Overview



MD5 Overview

1. Pad message so its length mod 512 = 448)
2. Append a 64-bit original length value to message
3. Initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. Process message in 16-word (512-bit) blocks:
   - Using 4 rounds of 16 bit operations on message block & buffer
   - Add output to buffer input to form new buffer value
5. Output hash value is the final buffer value

Padding Twist

- Given original message M, add padding bits "10^*" such that resulting length is 64 bits less than a multiple of 512 bits.
- Append (*original length in bits* mod 2^{64}), represented in 64 bits to the padded message
- Final message is chopped 512 bits a block

MD5 Process

- As many stages as the number of 512-bit blocks in the final padded message
- Digest: four 32-bit words: MD=A|B|C|D [4x32 = 128]
- Every message block contains sixteen 32-bit words: $m_0|m_1|m_2\ldots|m_{15}$ [16x32 = 512]

MD 5

- The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted *A*, *B*, *C*, and *D*. These are initialized to **certain fixed constants**.

Initialization:

A=01234567,

B=89abcdef,

C=fedcba98,

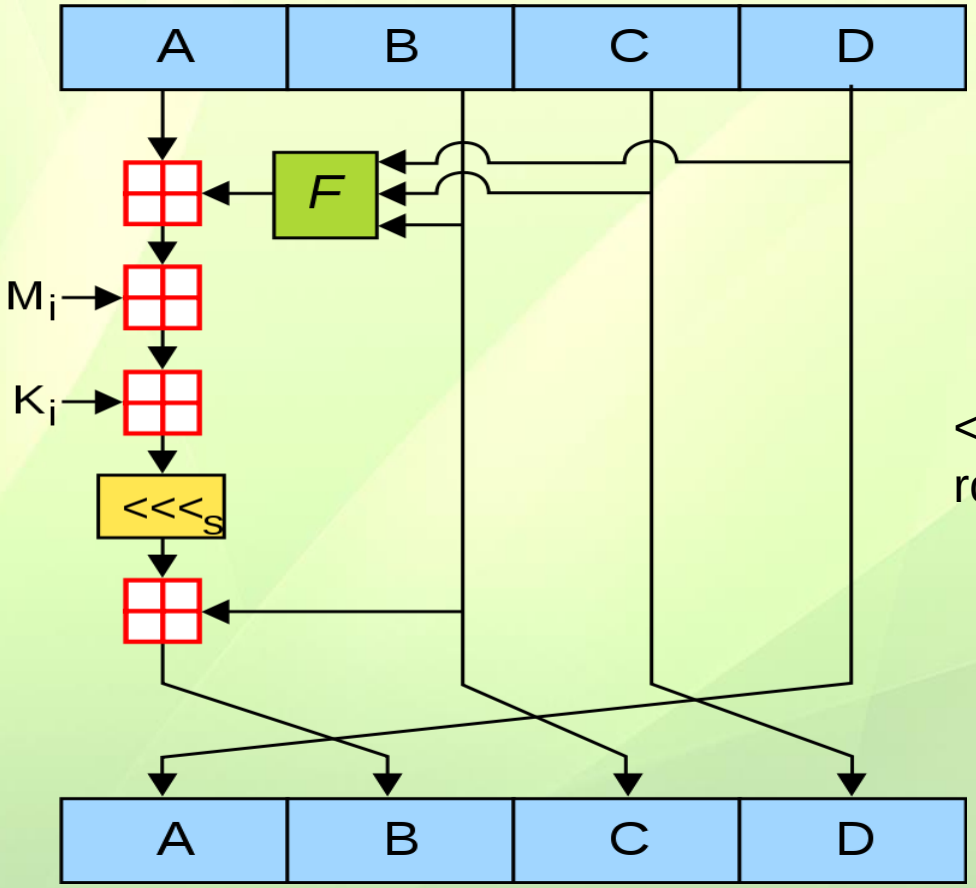
D=76543210

MD 5

The main algorithm then uses each 512-bit message block in turn to modify the state. The processing of a message block consists of four similar stages, termed *rounds*; each round is composed of 16 similar operations based on a non-linear function *F*, modular addition, and left rotation.

Operation of MD 5

M_i 32 bit word of message
K_i 32 bit word of constant.



$<<<_s$ denotes left bit rotation by s places.

Fig. 1: One MD 5 operation

MD 5

Figure 1 illustrates one operation within a round. There are four possible functions *F*; a different one is used in each round:

F (B, C, D) = (B & C)|| (!B & D)

G (B, C, D) = (B & D) || (C & !D)

H (B, C, D) = (B xor C xor D)

I(B, C, D) = C xor (B || !D)

Thank You.