**Message Authentication:**

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. It is intended against the attacks like content modification, sequence modification, timing modification and repudiation. For repudiation, concept of digital signatures is used to counter it.

There are three classes by which different types of functions that may be used to produce an authenticator. They are:

*Message encryption*–the ciphertext serves as authenticator

*Message authentication code (MAC)–*a public function of the message and a secret key producing a fixed-length value to serve as authenticator. This does not provide a digital signature because A and B share the same key.

*Hash function*–a public function mapping an arbitrary length message into a fixed-length hash value to serve as authenticator. This does not provide a digital signature because there is no key.

**Message Encryption:**

Message encryption by itself can provide a measure of authentication. The analysis differs for conventional and public-key encryption schemes. The message must have come from the sender itself, because the ciphertext can be decrypted using his (secret or public) key. Also, none of the bits in the message have been altered because an opponent does not know how to manipulate the bits of the ciphertext to induce meaningful changes to the plaintext. Often one needs alternative authentication schemes than just encrypting the message.

1. Sometimes one needs to avoid encryption of full messages due to legal requirements.

2. Encryption and authentication may be separated in the system architecture.

The different ways in which message encryption can provide authentication, confidentiality in both symmetric and asymmetric encryption techniques is explained with the table below:



**Message Authentication Code :**

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as cryptographic checksum or MAC, which is appended to the message. This technique assumes that both the communicating parties say A and B share a common secret key K. When A has a message to send to B, it calculates MAC as a function C of key and message given as: **MAC=Ck(M)** The message and the MAC are transmitted to the intended recipient, who upon receiving performs the same calculation on the received message, using the same secret key to generate a new MAC. The received MAC is compared to the calculated MAC and only if they match, then:

1. The receiver is assured that the message has not been altered: Any alternations been done the MAC’s do not match.

2. The receiver is assured that the message is from the alleged sender: No one except the sender has the secret key and could prepare a message with a proper MAC.

3. If the message includes a sequence number, then receiver is assured of proper sequence as an attacker cannot successfully alter the sequence number.

Basic uses of Message Authentication Code (MAC) are shown in the figure:



There are three different situations where use of a MAC is desirable:

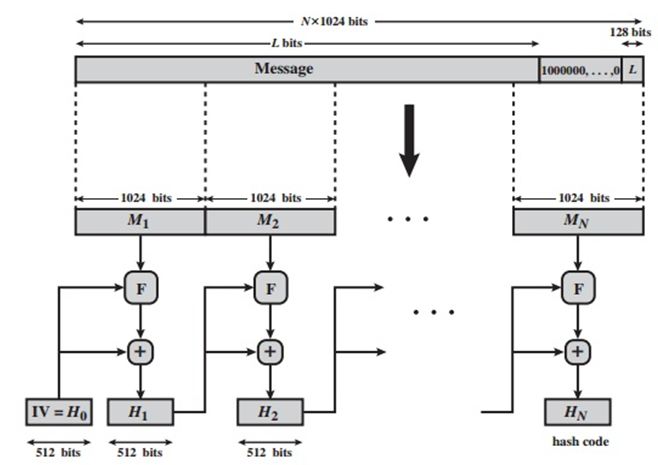
1.If a message is broadcast to several destinations in a network (such as a military control center), then it is cheaper and more reliable to have just one node responsible to evaluate the authenticity –message will be sent in plain with an attached authenticator.

2.If one side has a heavy load, it cannot afford to decrypt all messages –it will just check the authenticity of some randomly selected messages.

3.Authentication of computer programs in plaintext is very attractive service as they need not be decrypted every time wasting of processor resources. Integrity of the program can always be checked by MAC.

**SHA :**

The secure hash algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST). SHA-1 is the best established of the existing SHA hash functions, and is employed in several widely used security applications and protocols. The algorithm takes as input a message with a maximum length of less than 264 bits and produces as output a 160-bit message digest.



The input is processed in 512-bit blocks. The overall processing of a message follows the structure of MD5 with block length of 512 bits and a hash length and chaining variable length of 160 bits. The processing consists of following steps:

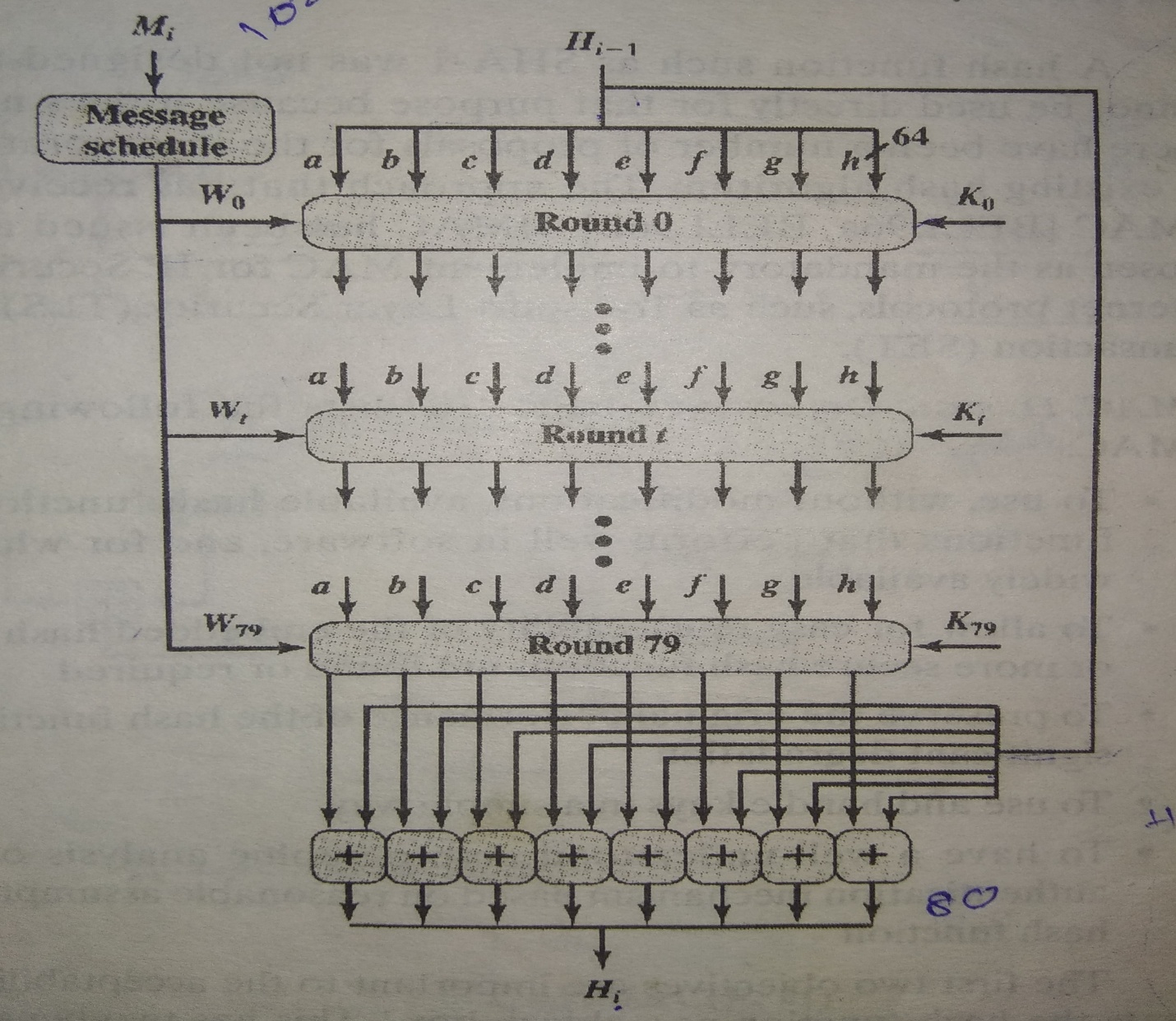
1.) ***Append Padding Bits:*** The message is padded so that length is congruent to 448 modulo 512; padding always added –one bit 1 followed by the necessary number of 0 bits.

2.) ***Append Length:*** a block of 64 bits containing the length of the original message is added.

3.) ***Initialize MD buffer*:** A 160-bit buffer is used to hold intermediate and final results on the hash function. This is formed by 32-bit registers A,B,C,D,E. Initial values: A=0x67452301, B=0xEFCDAB89, C=0x98BADCFE, D=0x10325476, E=C3D2E1F0. Stores in big-endian format i.e. the most significant bit in low address.

4.) ***Process message in blocks 512-bit (16-word) blocks*:** The processing of a single 512-bit block is shown above. It consists of four rounds of processing of 20 steps each. These four rounds have similar structure, but uses a different primitive logical function, which we refer to as f1, f2, f3 and f4. Each round takes as input the current 512-bit block being processed and the 160-bit buffer value ABCDE and updates the contents of the buffer. Each round also makes use of four distinct additive constants Kt. The output of the fourth round i.e. eightieth step is added to the input to the first round to produce CVq+1.

5.) *Output:* After all L 512-bit blocks have been processed, the output from the Lth stage is the 160-bit message digest.

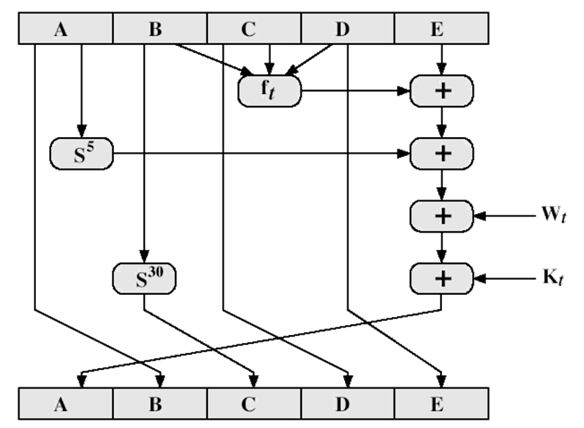


The behavior of SHA-1 is as follows: **CV0 = IV CVq+1 = SUM32(CVq, ABCDEq) MD = CVL** Where, IV = initial value of ABCDE buffer ABCDEq = output of last round of processing of qth message block L = number of blocks in the message SUM32 = Addition modulo 232 MD = final message digest value.

***SHA-1 Compression Function:***

Each round has 20 steps which replaces the 5 buffer words. The logic present in each one of the 80 rounds present is given as **(A,B,C,D,E) <- (E + f(t,B,C,D) + S5(A)+ Wt+** **Kt),A,S30(B),C,D** Where, A, B, C, D, E = the five words of the buffer t = step number; 0< t

1. 79 f(t,B,C,D) = primitive logical function for step t Sk = circular left shift of the 32-bit argument by k bits Wt **=** a 32-bit word derived from current 512-bit input block. Kt **=** an additive constant; four distinct values are used + = modulo addition.



SHA shares much in common with MD4/5, but with 20 instead of 16 steps in each of the 4 rounds. Note the 4 constants are based on sqrt(2,3,5,10). Note also that instead of just splitting the input block into 32-bit words and using them directly, SHA-1 shuffles and mixes them using rotates & XOR’s to form a more complex input, and greatly increases the difficulty of finding collisions. A sequence of logical functions f0, f1,..., f79 is used in the SHA-1. Each f**t**, 0<=t<=79, operates on three 32-bit words B, C, D and produces a 32-bit word as output. ft(B,C,D) is defined as follows: for words B, C, D, **ft(B,C,D) = (B AND C) OR** **((NOT B) AND D) ( 0 <= t <= 19) ft(B,C,D) = B XOR C XOR D (20 <= t <= 39) ft(B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59) ft(B,C,D) = B XOR C XOR D (60 <= t <= 79).**

1. Define Message Authentication Code.?List three approaches to Message Authentication.
2. What are the requirements of Authentication
3. Give a neat sketch to explain the concept of Secured Hash Algorithm (SHA).