Chapter 12

Cryptographic Hash functions

Iterated Hash function

All cryptographic hash functions need to create a fixed-size digest out of a variable-size message.

Creating such a function is best accomplished using iteration.

Instead of using a hash function with variable size input, a function with fixed-size input is created and is used a necessary number of times.

The fixed-size input function is referred to as a compression function.

It compresses an n-bit string to create an m-bit string where ~~n is~~ $n > m$.

This scheme is reffered to as an iterated cryptographic hash function.

Merkle - Damgaard Scheme :-

The Merokle - Damgard scheme is an iterated hash function.

This hash function is collision resistent, if the compression function is collision resistent.

The scheme uses the following steps:-

(1) The necessary padding is appended to the message so that message can be devided into a blocks of n-bits.

(2) Then the message divided into ~~n-bits~~ t blocks of n-bits. We call these block $m_1, m_2, m_3 \dots m_t$.

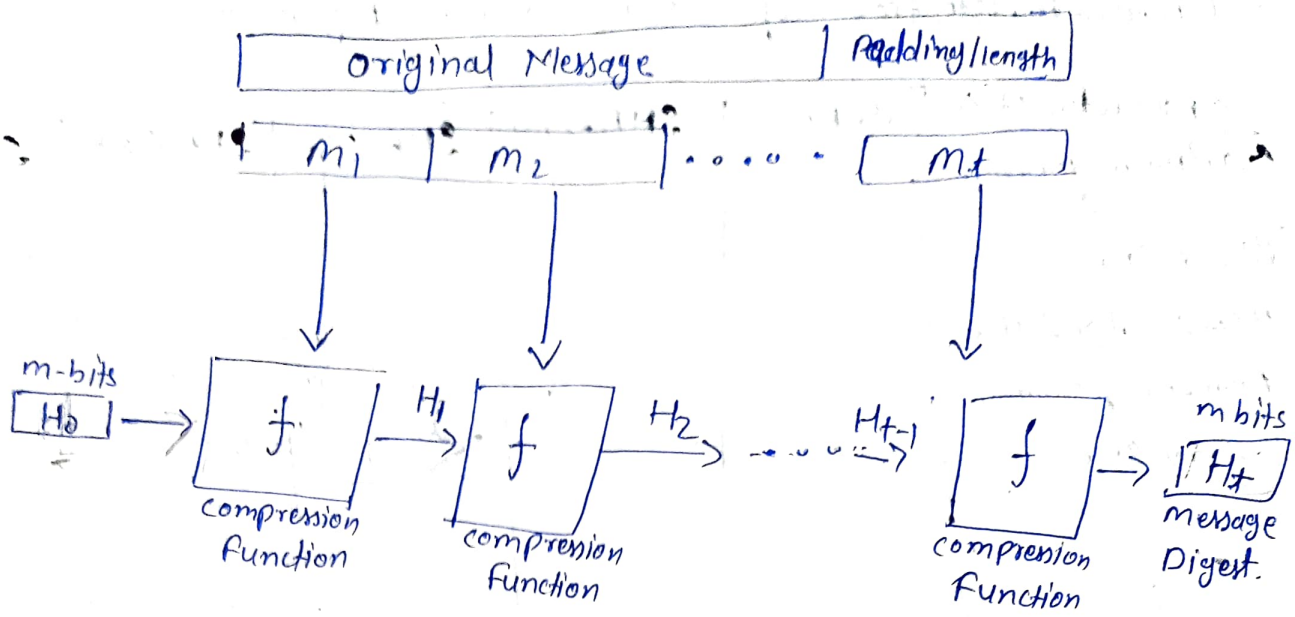
(3) The digest created at t iterations $H_1, H_2 \dots H_t$

(3) Before starting the iteration, the digest H_0 is set with fixed value.

(4) The compression function at each iteration operates on H_{i-1} & M_i to create new H_i.

In other words, we have $H_i = f(H_{i-1}, M_i)$

(5) H_i is cryptographic hash function of the original message, that is $h(M)$.



Hash function Made from Scratch :-

These compression functions are specifically designed for the purpose they serve.

Message Digest (MD) :- Several hash algorithms were designed by Ron Rivest.
These are reffered to as MD2, MD4, MD5.
MD stands for message digest.
The last version MD5 that divides the message into blocks of 512 bits and creates a 128-bit digest.

Secure Hash Algorithm (SHA) :- It is sometime referred to as Secure Hash Standard (SHS)

This is Based on MD5.
which includes :- SHA-1, SHA-224, SHA-256, SHA-384
SHA-512.

Characteristics of Secure Hash Algorithms

Characteristics	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Maximum message size	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	$2^{128}-1$
Blocke size	512	512	512	1024	1024
message digest size	160	224	256	384	512
Number of rounds	80	64	64	80	80
word size	32	32	32	64	64

All of versions has a same structure.

~~# SHA-512~~

SHA-512 :- SHA-512 is the version of SHA with a 512-bit message digest. It is based on the Merkle-Damgard Scheme. ~~We have chosen~~ It is latest version of this family. It has a more complex ~~Stru~~ Structure then the others. & It's message digest is the longest.

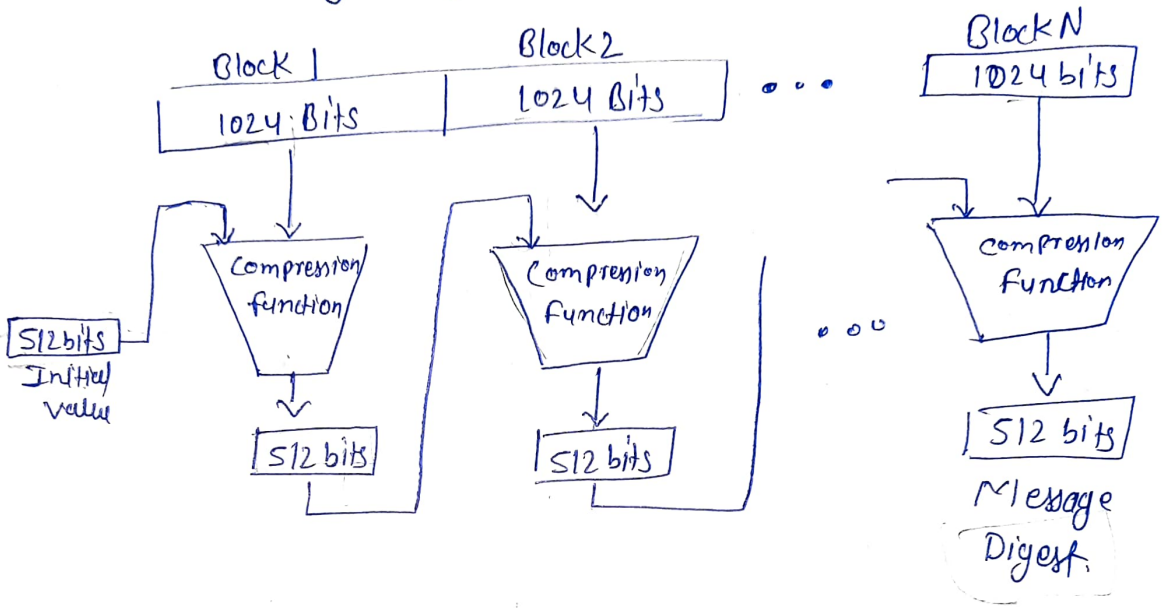
It creates a digest of 512 bits from a multiple-block message. Each block is 1024 bit in length.

Digest creation :- The digest is initialized with the predetermined value of 512 bits.

we give this initial value & Block 1 data to compression function to create the first intermediate message digest of 512 bits.

This message digest & Block 2 data again give to the compression function to create second intermediate digest.

finally the $(N-1)^{th}$ digest is, & N^{th} block is given to the compression function to create final Message Digest for entire message.



Message Preparation ~~Before encryption~~ :-

SHA-512 insists that the length of the original message be less then 2^{128} bits. This means that message of length equals or greater than 2^{128} is Not be processed by SHA-512

This is Not possible because 2^{128} bits is Probably larger than the total storage capacity of any system.

NOTE :- A communications network that can send 2^{64} bits per second is not yet available. Even it would take many years to send this message. This tells us that we do not need to worry about the SHA-512 message length restriction.

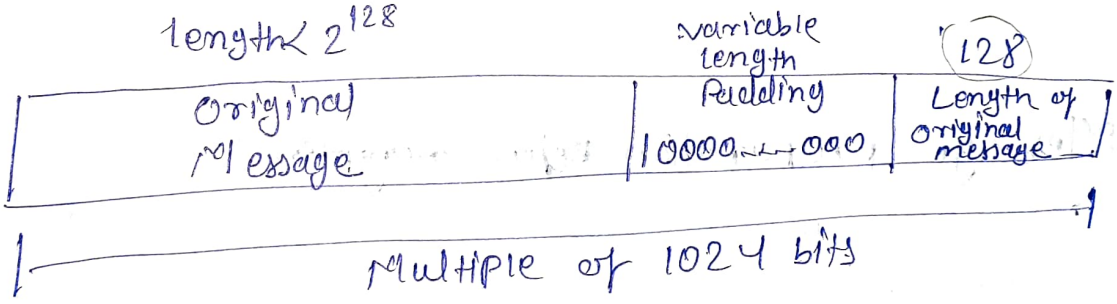
Length field and padding :-

Before the message digest can be created, SHA-512 requires the addition of a 128-bit to the message that defines the length of the message in bits.

~~The len~~

NOTE :- The length field defines the length of the original message before adding the length field or padding.

Before the addition of length field, we need to pad message to make the length a multiple of 1024.



The length of padding field can be calculated as follows.

Let $|M|$ be the length of original message & $|P|$ be the length of the padding field

then $(|M| + |P| + 128) = 0 \bmod 1024$

or $|P| = (-|M| - 128) \bmod 1024$

The formate of padding is one 1 followed by necessary numbers of 0s.

Ex:- What is the number of padding bits if the length of the original message is 2590 bits?

we can calculate padding as follows:-

$$|P| = (-2590 - 128) \bmod 1024$$
$$|P| = (-2718) \bmod 1024$$
$$|P| = 354$$

The padding consists of one 1 followed by 353 0's

Ex:- Do we need padding if the length of the original message is already a multiple of 1024 bits?

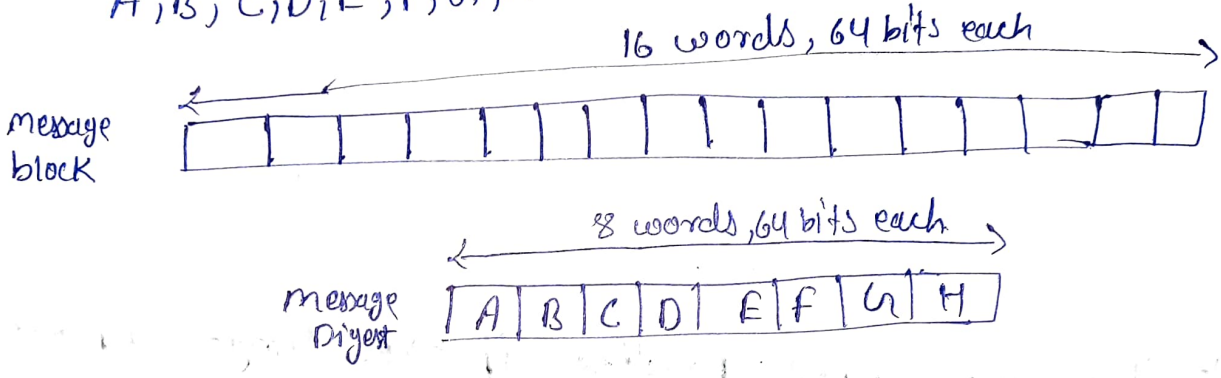
Yes we do, because need ~~the~~ to add the length field. So padding is needed to make the new block a multiple of 1024 bits.

$$|M| = 1024$$
$$|P| = (-1024 - 128) \bmod 1024$$
$$\boxed{|P| = 896}$$

NOTE:- The minimum length padding is 0 & maximum length padding is 1023.

Words :- SHA-512 operates on words, It is word oriented. A word is defined as 64 bits.

This means that after padding and the length field are added to the message, each block of the message consists of sixteen 64-bit words. The message digest is also made of eight 64-bit words named as A, B, C, D, E, F, G, H.



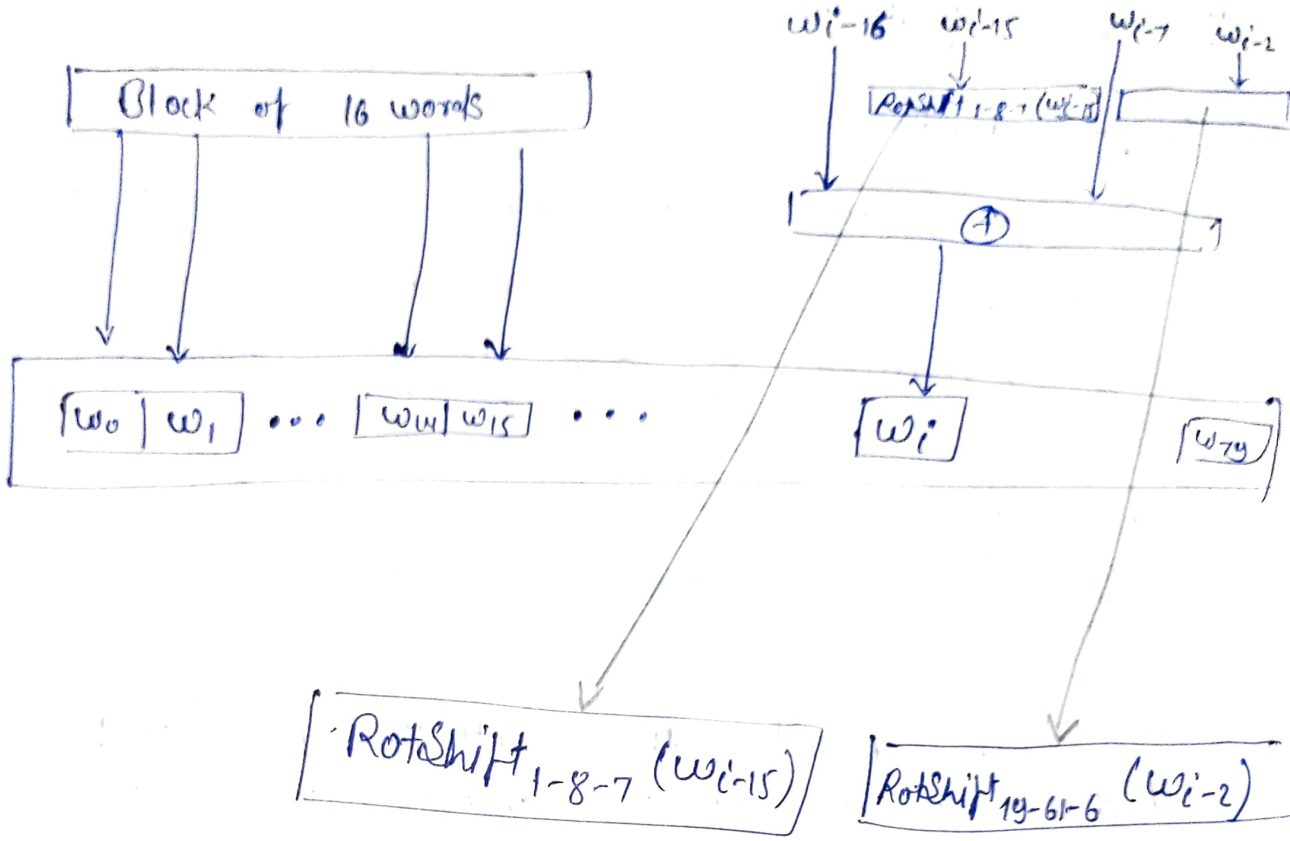
Word Expansion :-

Before Processing, each message block must be expanded. A block is made of 1024 bits, or Sixteen 64-bit words, we need 80 words in the Processing phase.

So that 16 words block needs to expanded to 80 words, from w_0 to w_{79}

First 16 words (w_0 to w_{15}) are from original message block (1024 bits)

Rest are Depends on the following operation.



$RotShift_{k-m-n}(x) = RotR_k(x) \oplus RotR_m(x) \oplus ShL_n(x)$

$RotR_i(x)$ = Right-rotation of the argument x by i bits

$ShL_i(x)$ = Shift-Left of the argumend x by i bits and padding the left by 0's.

Ex:-

$$w_{60} = w_{60-16} \oplus RotShift_{1-8-7}(w_{45}) \oplus w_{53} \oplus RotShift_{19-61-6}(w_{58})$$

~~Message Digest initialization:~~

Compression function:

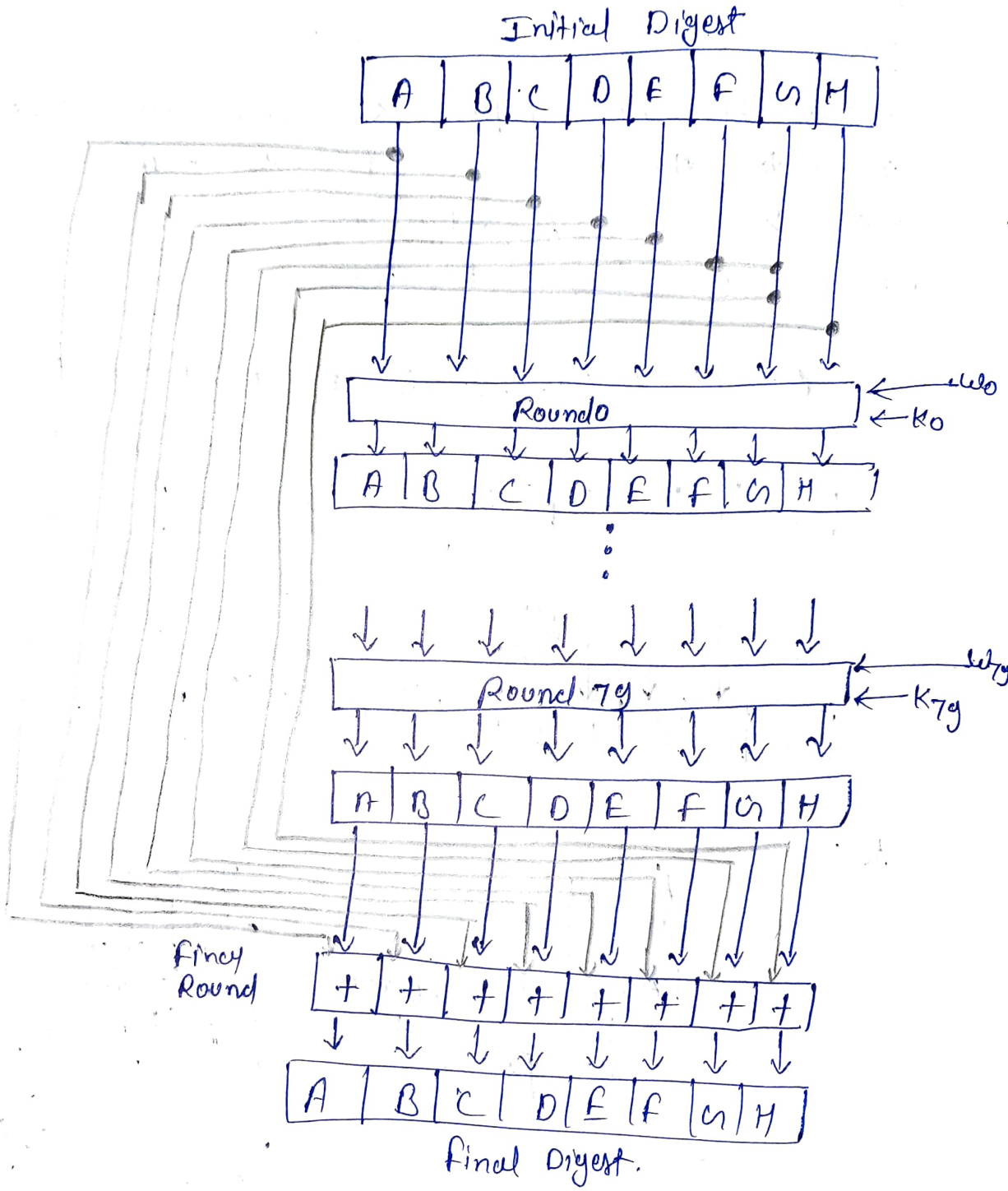
SHA-512 creates a 512 bit message digest from a multiple-block message where each block is 1024 bits

The processing of one Block of data in SHA-512 involves 80 Rounds.

The Algorithm uses eight constants for message digest initialization, we Named this as A, B, C, D, E, F G, H.

In each Round the contents of eight previous buffers, are operated with expanded word wi & 64-bit constant (Ki) to create a new eight new word buffer.

At the beginning of processing the value of the eight buffers are saved into eight temporary variables. , & added to the result of last Round.

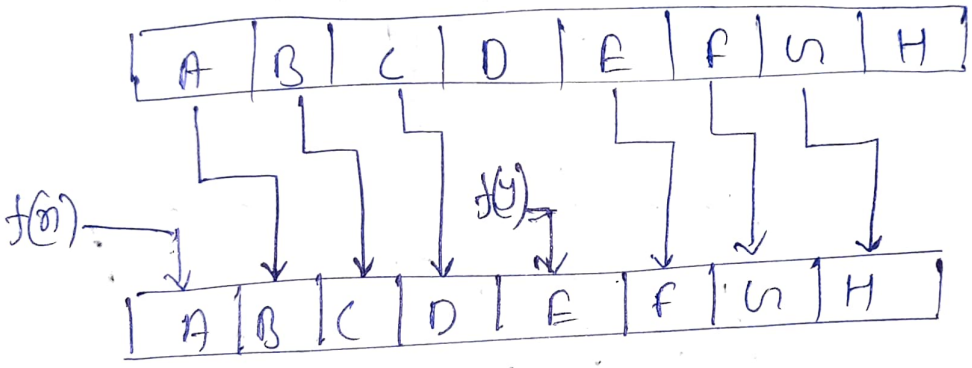

Initial Digest
A B C D E F G H
Round 0
K0
A B C D E F G H
Round 79
K79
A B C D E F G H
Final Round
+ + + + + + + +
A B C D E F G H
final Digest.

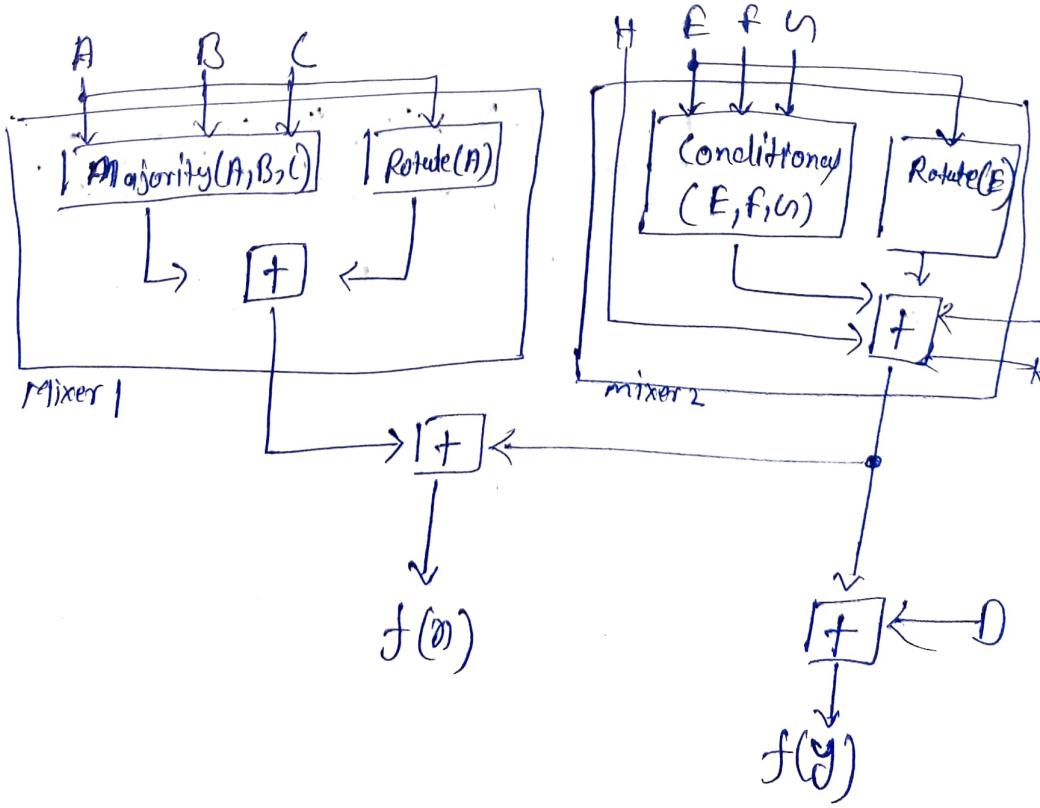
Structure of Each Round :-

In each Round eight new words are created from previous words & W_i & K_i.

In this new buffer 6 words are exact copied to from the previous buffer. But 2 word A & E receives their input from some complex function.

$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $E \rightarrow F$, $F \rightarrow G$, $G \rightarrow H$.





Majority (x,y,z) = $(x$ AND $y) \oplus (y$ AND $z) \oplus (z$ AND $x)$

Conditional (x,y,z) = $(x$ AND $y) \oplus$ (NOT x AND $z)$

Rotate (x) = $RotR_{28}(x) \boxplus RotR_{34}(x) \boxplus RotR_{39}(x)$

$\boxplus$ addition modulo 2^{64}

$RotR_i(x)$: Right-rotation of the argument x by i bits.

Example :- We apply the Majority function on buffers A, B, C. If leftmost digits of these buffers are 0x7, 0xA & 0xE respectively, what is the leftmost digit of the result?

Solution

The digits in Binary are

A = 0111
B = 1010
C = 1110

The first bit are 0, 1, 1

then majority

$(0$ AND $1) \oplus (1$ AND $1) \oplus (1$ AND $0) =$ ~~$0 \oplus 1 \oplus 1$~~

$0 \oplus 1 \oplus 0 = 1$

Similarly majority (1,0,1) = 1
majority (1,1,1) = 1
majority (1,0,0) = 0

The result is 1110 = 0xE

Ex:-

Apply condition function on ~~E, F, H~~ ~~with left~~ : 0x9, 0xA, 0xF

Binary value = 1001
1010
1111

condition (1,1,1) = (1 AND 1) ⊕ (NOT 1 AND 1) =

= 1 ⊕ 0 = 1

condition (0,0,1) = 0 ⊕ 1 = 1

condition (0,1,1) = 0 ⊕ 1 = 1

condition (1,0,1) = 0 ⊕ 0 = 0