# *Steps involved in creating hash function:*

-> In this notes, I have explained about the SHA512 algorithm's logic with explanation and examples

**HASH FUNCTION BASIC CONDITIONS:**

->Hash function should convert a variable size string into fixed sized hexadecimal output

->Small changes to string should have a larger impact on the output.

->Two different strings should not have same hash value (Collision resistant)

**BREAKING OF STRING:**

-> As a first step, we have to break the string(password) into pieces of equal sizes based on output size requirement by converting ASCII values into hexadecimal form along with padding(concatinating of extra bits which is explained later).

Example:

if password="abc"

97 98 99(ASCII) -> converted to hexadecimal values 61 62 63

616263(totally 24 bit size) but we want 512 bits size hexadecimal output

so we are going to pad extra bits to make it compatible for the next processes.

**PADDING DEFINITION:**

Padding is the process of adding extra bits to the hexadecimal string to make it compatible for the next process. It involves the following process:

1.Concatination of "100000....so on"( Up to 896 bits-len(password in bits) ) in hex i.e 1000 0000->8 0 ... so on(1000 -> 8) + length of original string in hex in 128 bits size with password.

2.It is done so that the total length (After padding) becomes a multiple of hash function output size 512 bits.

Eg:Here password size is 24 bits length = 18 in hex which is converted to "...000018 (length 32)"(128 bits) in hex to obtained result such that the total length becomes multiple of 512 bits after padding. Eg: like 1024,2048,..etc

why do 1024 bits after padding and not 512?

It's because of making more processes to involve inside the hash function to avoid collision.

**MESSAGE FORMATION:**

->The next step after padding is to divide those bits obtained into equal pieces of output size.

Eg In SHA512, size of hash output is 512 bits and hence break 1024 bits into pieces of 512 bits i.e 512,512 or in case of obtaining 2048 bits after padding, divide it like 512,512,512,512.

**INITIALIZATION VECTORS:**

Now we have to initialize some variables with some constant hexadecimal values, each of size 64 bits such that the total number of variables created should have a total size equal to hash output size.

Eg: In SHA512 ,we should create 8 variables with some constant hex values in it (Known as Initialization Variable),such as a,b,c,d,e,f,g,h each of size 64 bits and total of 64x8=512 bits (which is the size of hash output required)

->We have to use each of the subdivisions obtained from padding and process it with all the initialization variables such that the values inside variables get changed and also process all the other subs in sequence so that at the end, the variables "a-h" will hold the required hash value constant size of 512 bits.

**WORD FORMATION:**

->Before the process we have to further make 80 words of 64 bits size from each 512 bit subdivision to make more processes to get involved and make our hash more resistive to collisions.This involves the following steps

1.From 512 bits subdivision from padding,the first 16 words can be directly obtained by dividing 512 by 64.

2.For the remaining 64 words ,there is a formula to derive new words from the previously obtained words.

Wn=Wn-16+ s(Wn-15) + Wn-7 +s(Wn-2)

Eg1: W17=W1 + s(W2) + W10 + s(W15) , where s(x)= rr1(x) XOR rr8(x) XOR rl7(x)

rr-rotate right rl- rotate left

(Use >> and << operator and ^ for XOR in java)

Eg2: rr1(1001) ->9 in decimal= 1100->12 in decimal,where bits are cyclically rotated to right by 1 time

3.Derive the remaining words w7-w80 using the above formula.

4.With this formula all the other words (Totally 80 words) are derived from the 512 bit subdivision.

5.Now we have to process all these 80 words w1-w80 with variables a,b,c,d,e,f,g,h previously created

**WORD PROCESSES AND MESSAGE DIGEST:**

The processing of the words w1-w80 with variables involves the following actions:

1. a=T1+T2, b=a, c=b, d=c, e=d+T1, f=e, g=f,h=g , where T1 and T2 are functions which involves bit wise shift, XOR operations with nearby variables (h for variable a and d for variable e respectively )

2.Use these newly obtained a-h variables for next round or next word W2 and so on until 80 words are used for process, so there will be totally 80 rounds of the process happens.

3.Finally obtained result on the a-h variables will be highly scrambled.

4 .Repeat these processes for the next 80 words from the message subdivision left from the padded messages using these scrambled a-h variables itself.

5.This process is known as "MESSSAGE DIGEST".

**FINAL HASH VALUE:**

->At the end, the variables a-h will hold some hexadecimal values which can be concatenated together to obtain the required hash value at the end.