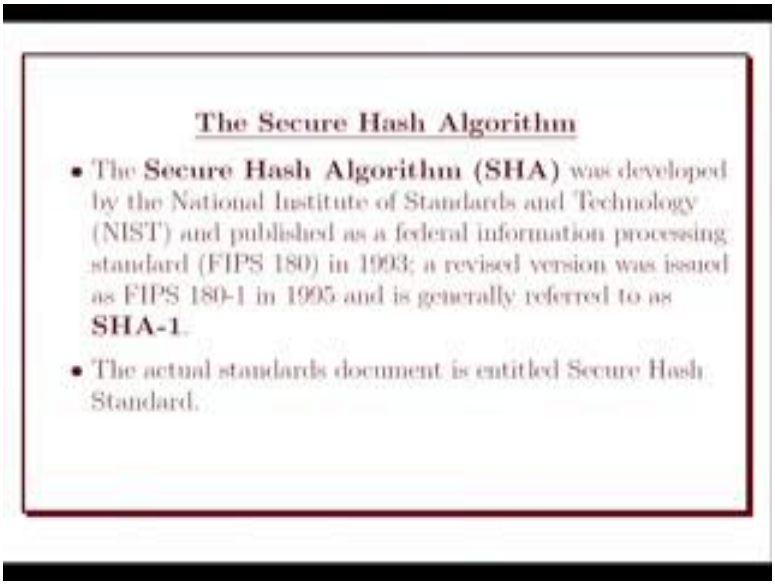
Internetwork Security
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture- 43
Secure Hash Algorithm (SHA)

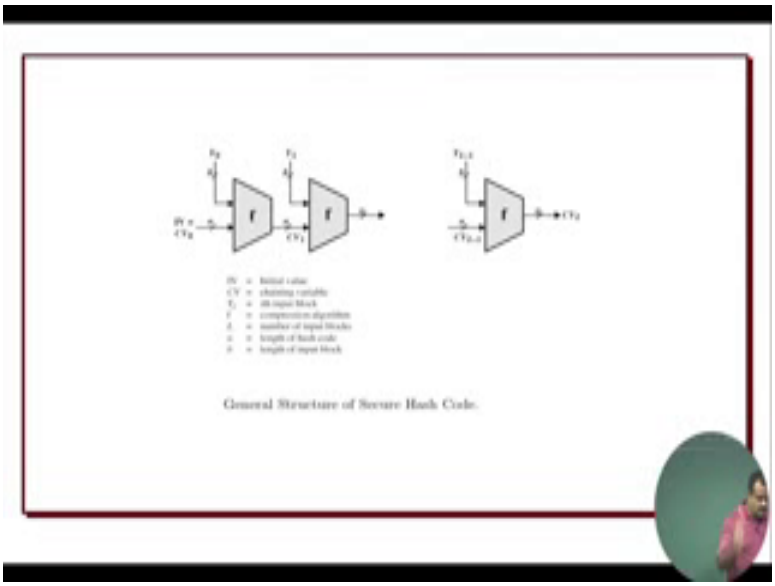
So, we will talk about secured hash algorithm - SHA, we will learn SHA-1 the hash function which is taking a arbitral line input and the length of the input could be 2 to the power 26 bit.

(Refer Slide Time: 00:33)



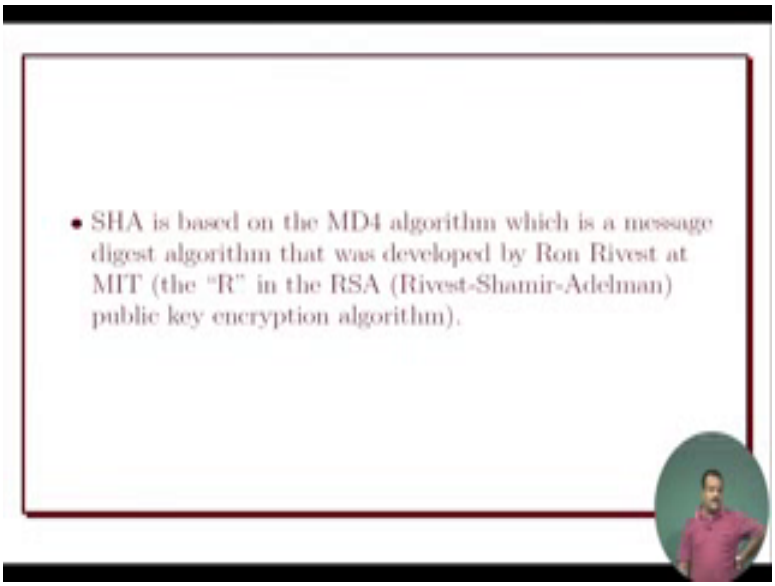
And so here is the can you please go to the slide. So, here is the SHA-1. So, SHA was developed by NIST in 1993; and it was replaced by SHA-1. And this was the standard during that, but now there is a recent competition was going on, so that is to have a standardization of this SHA.

(Refer Slide Time: 01:04)



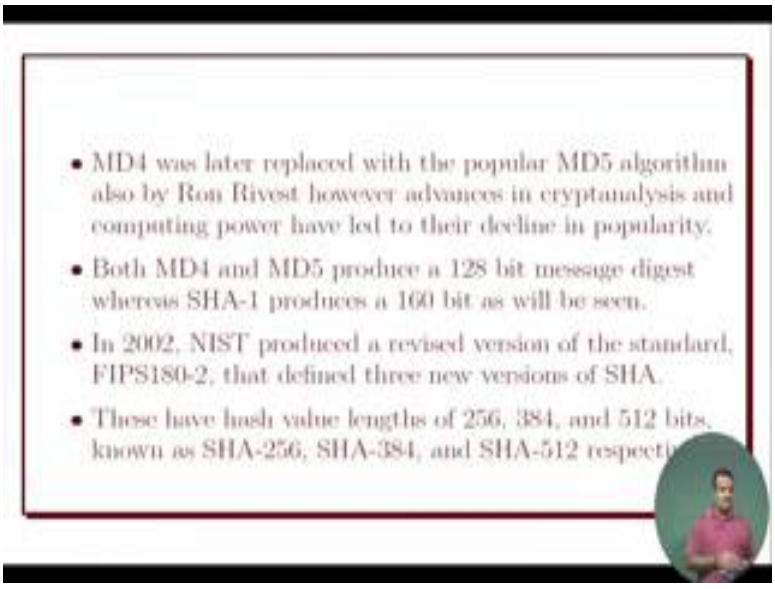
So, just come back to the general structure of a secure hash code. So, any hash function has this structure how we can decompose the message into a fixed length message. So, here is we have arbitrary length message and we partition this message into y 1, y 2, y l minus 1. So, suppose there are l blocks, and size of each block is b bits. And we have a function f which is called compression function or compression algorithm which takes n bit and b bit input and which is giving n bit output. And this is the initialization value or initialization texture or starting value this is c v 0; and all the c v is are the intermediate value. And then finally, c v l is our hash code. So, any hash function is having this compression function f.

(Refer Slide Time: 02:10)



So, we will talk about SHA-1, what is the compression function. So, for SHA-1 is based on the compression function MD4 algorithm which is a compression function which is their design by Rivest at MIT is one of the inventor for RSA.

(Refer Slide Time: 02:27)

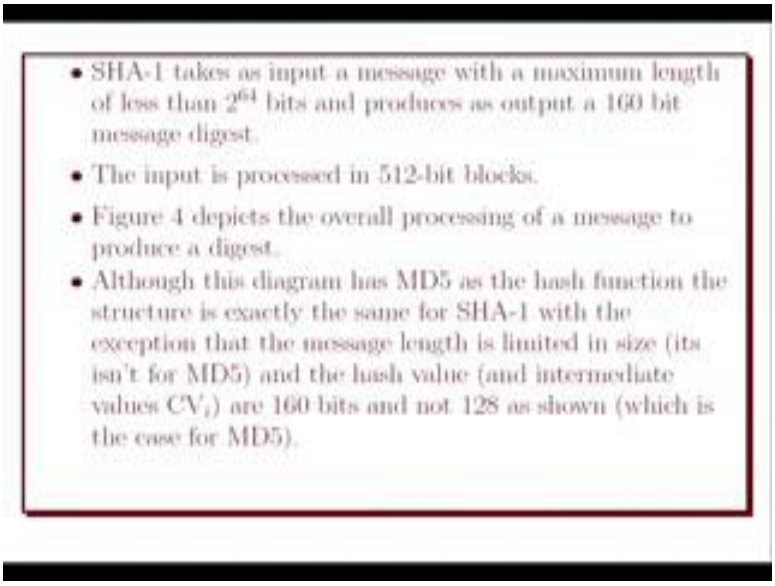


And then someone is later on MD4 is replaced by MD5 algorithm by Rivest, but MD4 an MD5 both are the function which is taking say one input is 512 bit, another input is 128 bit.

(Refer Slide Time: 02:42)

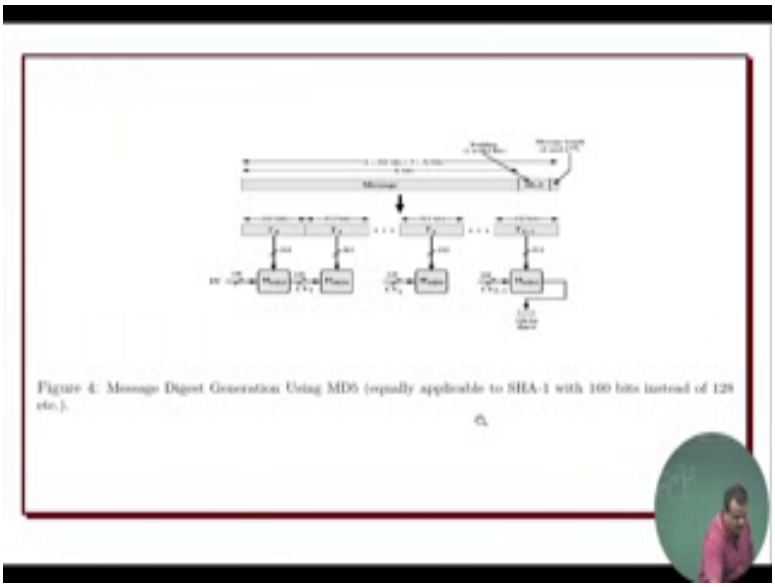
So, this is MD4 or MD5, and it is giving us 128 bit digest. So, if we use this just the MD5, it will give us ultimately the hash code is 128 bit. So, go to the slide please. So, this is, but we want this to be extend to make it 160 bit because our SHA-1 the hash code of the SHA-1 is 160 bit it takes a input of arbitrary length and it digests to the 160 bit output. So, we have to revise this MD5. So, there are some length of the value of this there are some version of SHA-1, SHA for 256, SHA-384 SHA-1, SHA-512 like this.

(Refer Slide Time: 03:44)



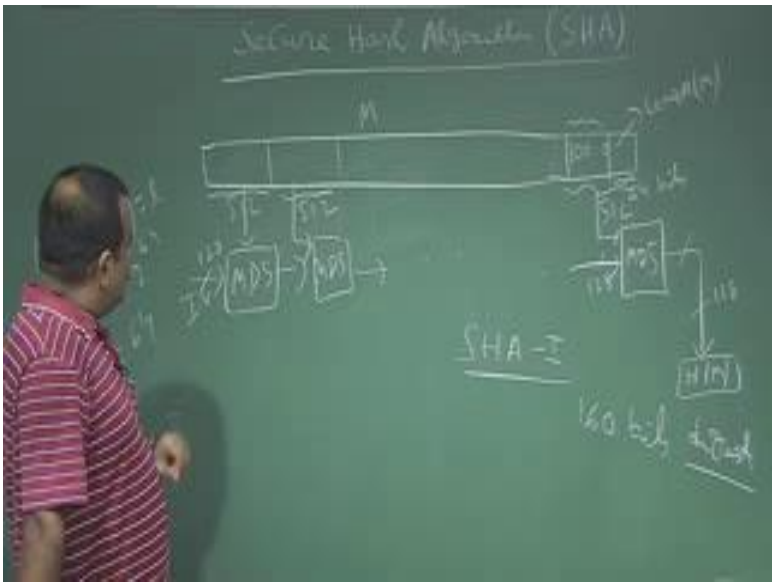
So, let us talk about this SHA 512 bit which is taking the algorithm MD5 as a compression function, but we have to modify this little bit to make it 160 bit because MD5 is 128 bit.

(Refer Slide Time: 04:03)



So, if you just apply the MD5, so what we are doing here in SHA-1, so this is our MD5, but in SHA-1 this should be 160 bit. So, we will come to that function - compression function which is basically using the MD5, but it is some modification on it because MD5 is 128 bit. So, what we are doing basically here, we are taking a message m and length of the message we are allowing is 2 to the power 64. So, each message must be less than 2 to the power 64 bit, so length of the message, so it is huge. And that length, so if length is 2 to the power 64 maximum, so that length can be store in 64 bit and that 64 length we are appending in the end of the message.

(Refer Slide Time: 05:04)

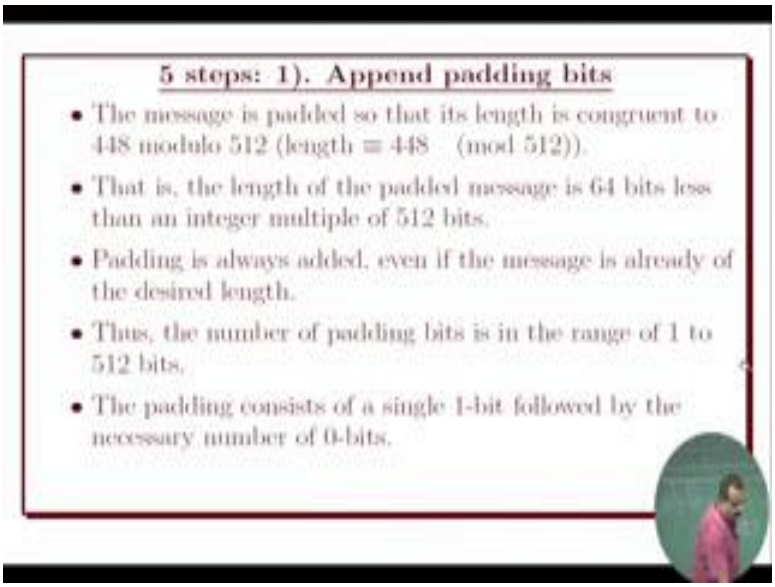


So, this is how we are; so this is we have a message in. And at the M, so we are allowing this is the message in. So, length of the message, if the length of the message is l, we are allowing l to be maximum 2 to the power 64. So that means, length of l, I mean length of length of l is 64 bits. So, now we store this is the length of M, and this is 64 bit. So, we are allowing maximum message size to be 2 to the power 64 bit. And we break it into 512 bits. So, each of this is 512 bit blocks. So, last bit also should be, so this is the last bit also should be 512 bit. So, for that, so this should be total 512 bit. So, for that maybe we need to append some bits over here. So, if we have to append, it is always append function then we will append the constant like one followed by all the zeros this is the way we append the bit.

So, go to the slide please. So, this is the break up. So, we are breaking this message. So, this is our message and this we stored in the last 64 bit we store the length of the message. So, this to make it one 512 bit, so we may have to append some bits over here which is basically constant which is for one followed by the zero to make to make it 512 bits. So finally, we break the message along with this with 512 bit blocks. Suppose, there are l blocks starting from 0 to l minus 1, so these are the blocks. So, now, suppose we have the compression function which is basically MD5 which is taking input 512 and another input 128 bit, and it is giving the intermediate values like 128 bit; again it is taking the second input like this we continue finally, we are getting this 128 bit has digest. This is the hash function on this plaintext M.

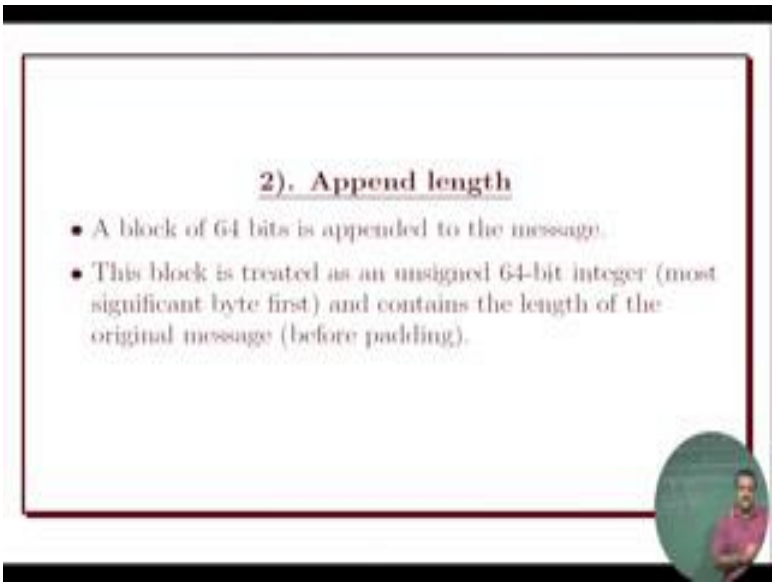
So, basically we are just compressing this MD5. So, we have the IV, this is 128 bit. So, this is also giving us 128 bit. So, another MD5 compression functions. So, like this, we continue. So, finally, so here is also you are applying MD5 and this is coming from the last before last block, and this is basically this is 128 bit, this is 512 bit, so this is 128. So, this is basically H of M. So, now, this is basically compressing into this hash function is digesting into a 128 bit. But in our SHA-1, SHA-1 is basically 160 bits digest. So, we have to modify this function, we will look at how this MD5 looks like for this 160 bit and what are this IV. So, those thing we will have a look.

(Refer Slide Time: 08:56)



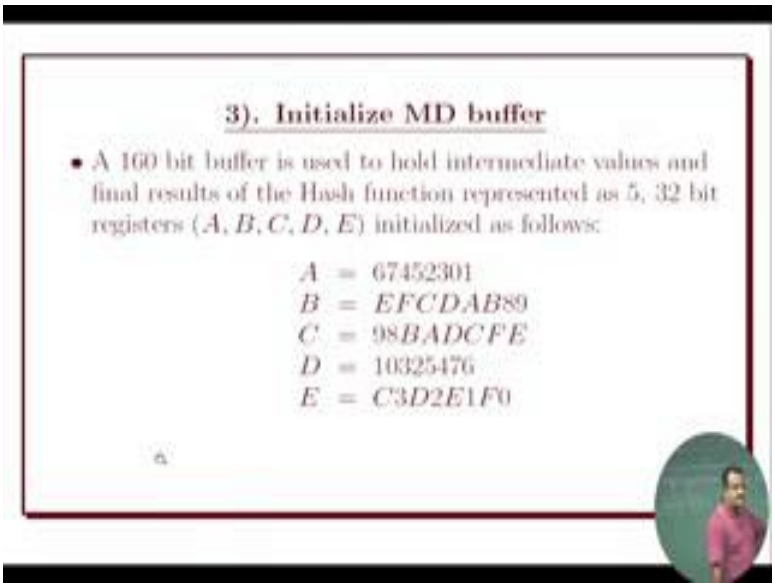
So, this is append the padding bits. So, last bits of 64 bits. So, maybe we need to append maximum 448 bits. So, 448 plus 64, it is 512. So, if you have to append then append will be one followed by all zeros. So, this we have already talked about.

(Refer Slide Time: 09:22)



And this is the length. So, length of the message we have to append. So, we are allowing the length to be maximum 64 bit. So, this 64 bit length, the integer we are adding to the end of the message.

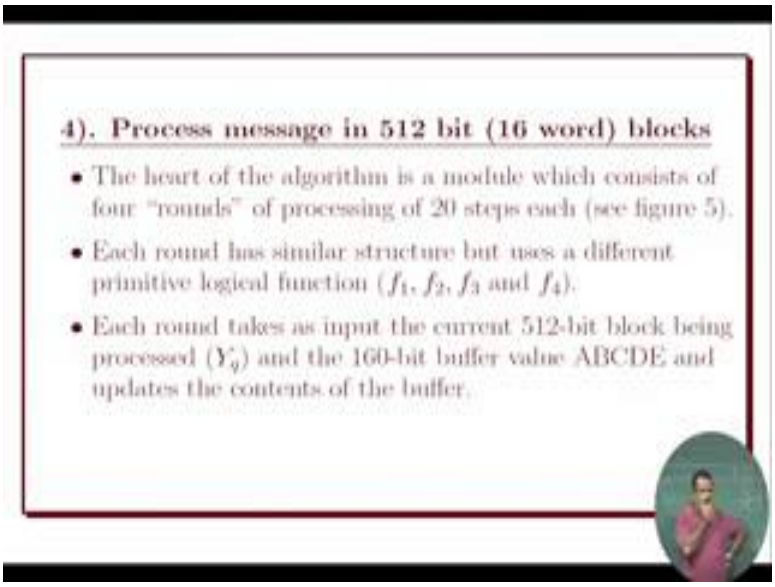
(Refer Slide Time: 09:35)



And this is the initialization vector - this IV. So, IV is basically so for SHA-1, this should be 160 bit. So, 160 bit if we block it if we break it into this five constant A, B, C, D, E then each is 32 bit, so 5 into 32 - 160 bit. So, each is 32 bit and 32 bit numbers can be written is in the 8 of 4 bit blocks. So, each of this is our hex representation, this is hex bit

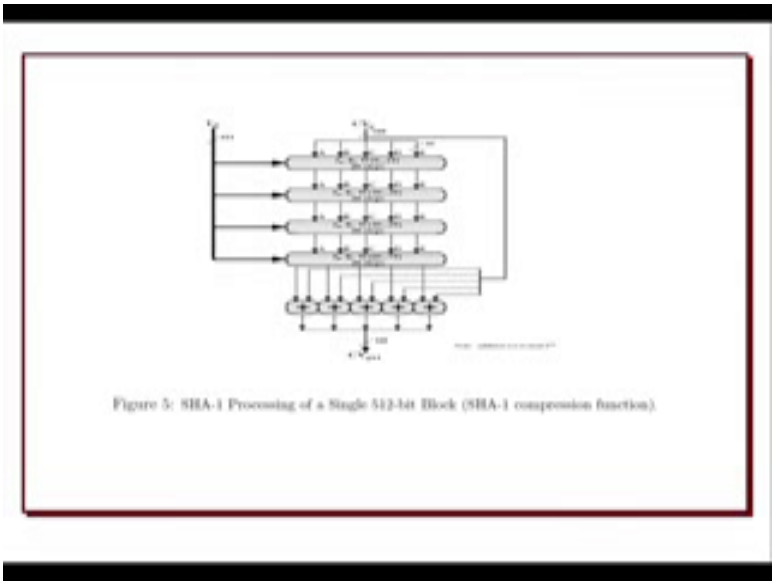
674, so this itself is a four bit each of this is a four bit, so total we have 4 into 8, so 32 bits. So, each of this constant is 32 bits. So, this is using the initialization vector for this SHA-1, I mean the modified version of MD5. So, this is 160, total is 160 bit. So, this is the IV which is 160 bit.

(Refer Slide Time: 10:46)

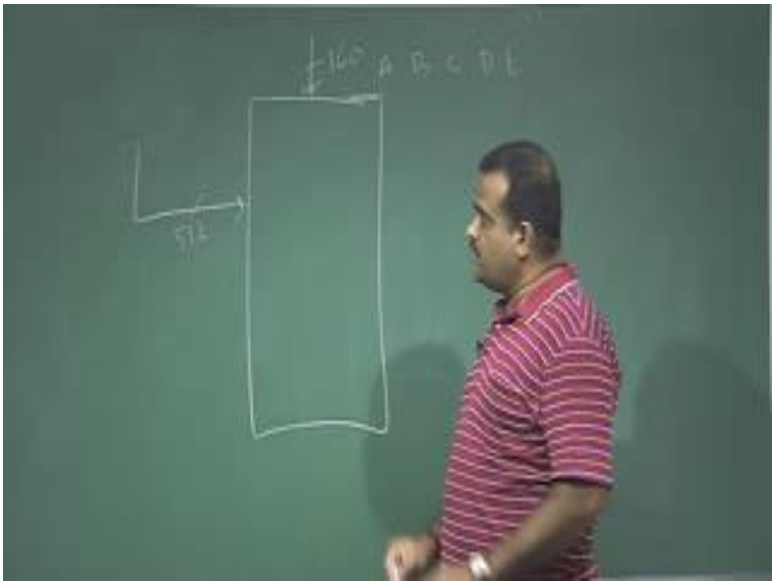


So, now we will come to the round of the MD5 I mean modified version of MD5.

(Refer Slide Time: 11:13)



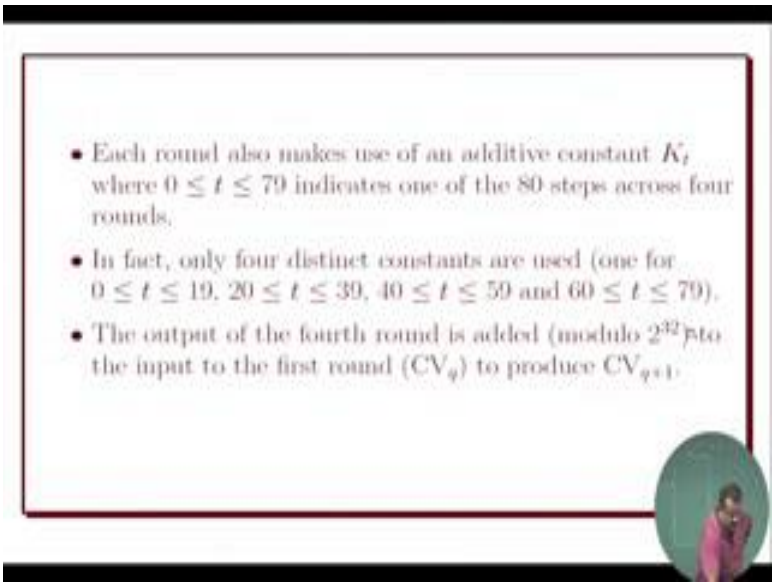
(Refer Slide Time: 11:15)



So, basically this is the let us come to the picture. So, this is the picture. So, it is basically taking two input, one is 160 bit and these are nothing but A, B, C, D, E; and another input is 512 bit. So, what we are doing. So, come to the slide please. So, what we are doing, we are taking this 512 bit and inside we have four round function we can say I mean four round function and in each of this round function there are 20 steps. So, 20 functions are, they are in each of this round function. And each of the round function is using a constant k and which is fixed for particular round. So, k is fixed for this round we use another constant k here for this round another constant k here for this round like this. So, each round is each round f 1, so f i has a input is 512 bit; and another input for first f 1, another input is this intermediate value A, B, C, D, E. And it has for 20-steps will come to that step how it is looks like it is basically the logical operation we are performing.

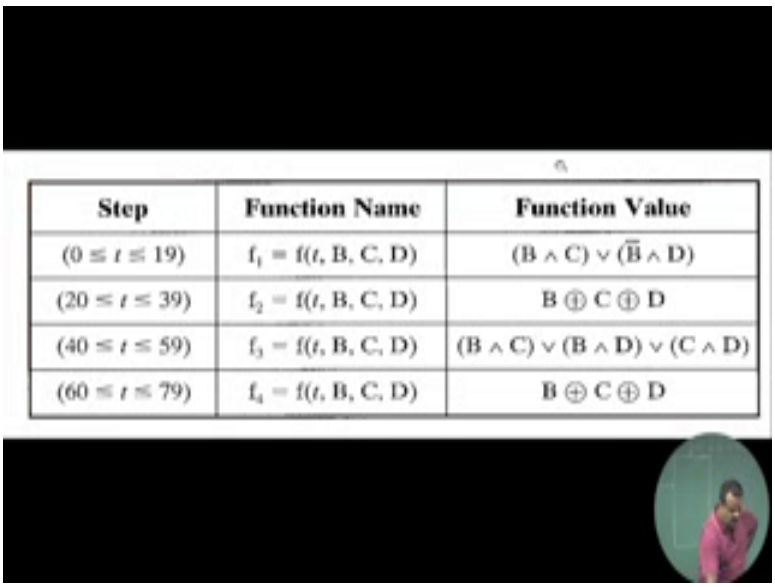
And then after that, after 20-steps, the output again will be the input for second round function which is called f 2. And there we use a different constant k and we use another input which is this 512 bit. So, like this we continue after the fourth round - f 4, we just take this intermediate value A, B, C, D, E and we take this initial value of this A, B, C, D, E and we will add to the addition, this is basically addition modulo 2 to the power 32. So, we will do the addition and then finally we get the 160 bit.

(Refer Slide Time: 13:29)



So, this is the addition operation. So, modulo 2 to the power 32 we just take 2 32 bit 32 bit and we add and then we take the modulus 2 to the power 32. And each round function this is the first round, this is the second round, the each round function is having 20 steps. And for each round function, we need a k and k fixed for each round, I mean individual round

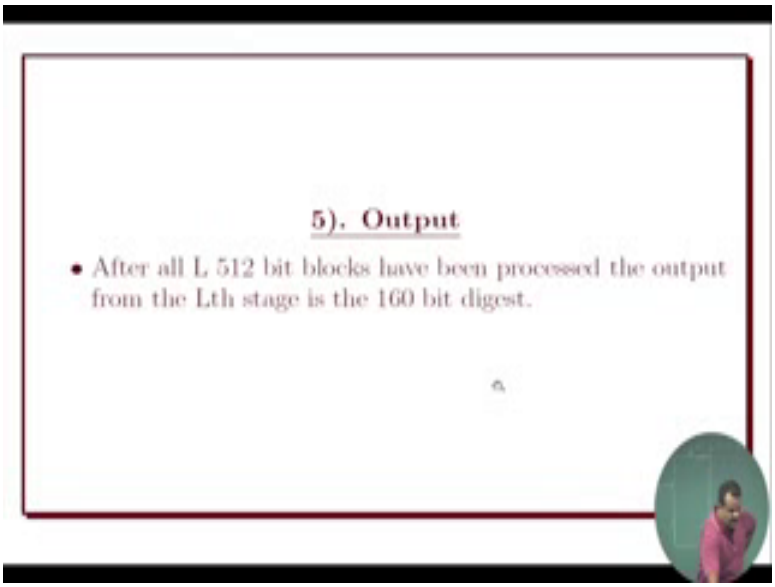
(Refer Slide Time: 13:58)



Step	Function Name	Function Value
$(0 \leq t \leq 19)$	$f_1 = f(t, B, C, D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$(20 \leq t \leq 39)$	$f_2 = f(t, B, C, D)$	$B \oplus C \oplus D$
$(40 \leq t \leq 59)$	$f_3 = f(t, B, C, D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$(60 \leq t \leq 79)$	$f_4 = f(t, B, C, D)$	$B \oplus C \oplus D$

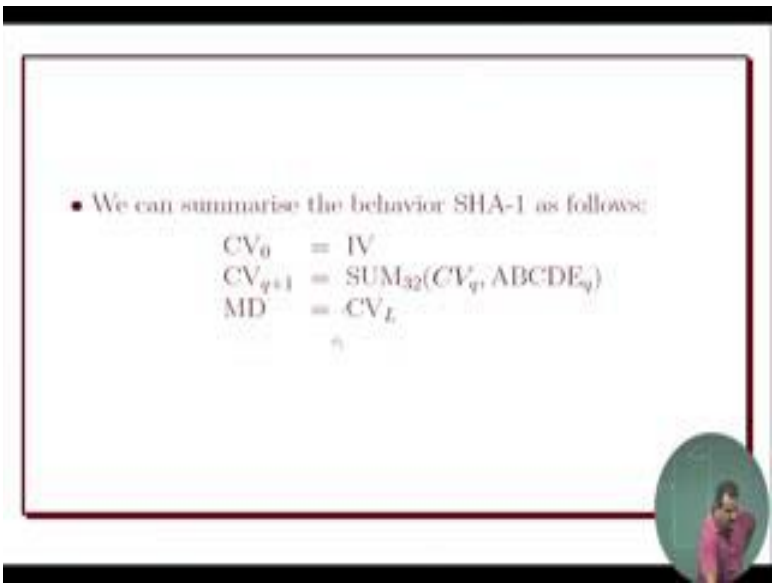
So, this is the round function. So, for first round we take this we use B, C, D, and we use this operation this is the logical operation. For the second round, we do this like this. So, these are the round function of this modified MD5.

(Refer Slide Time: 14:18)



And the finally, the output will be the after the Lth round output will be the; so this is all round. So, we continue like this after Lth round, it will be 160 bit digest.

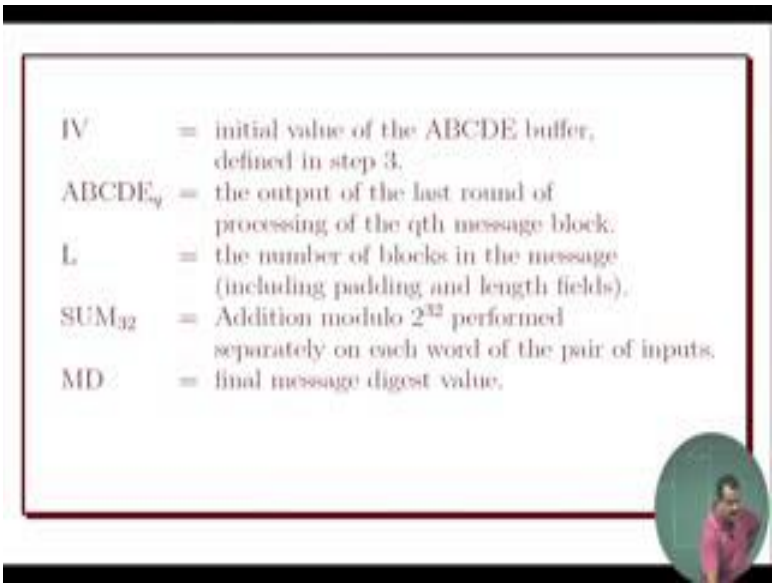
(Refer Slide Time: 14:30)



And this is the initialization. So, we can initialize the C V 0 by IV initialization vector which is given by this 32 bits and then final round is the basically this C V q plus 1. C V

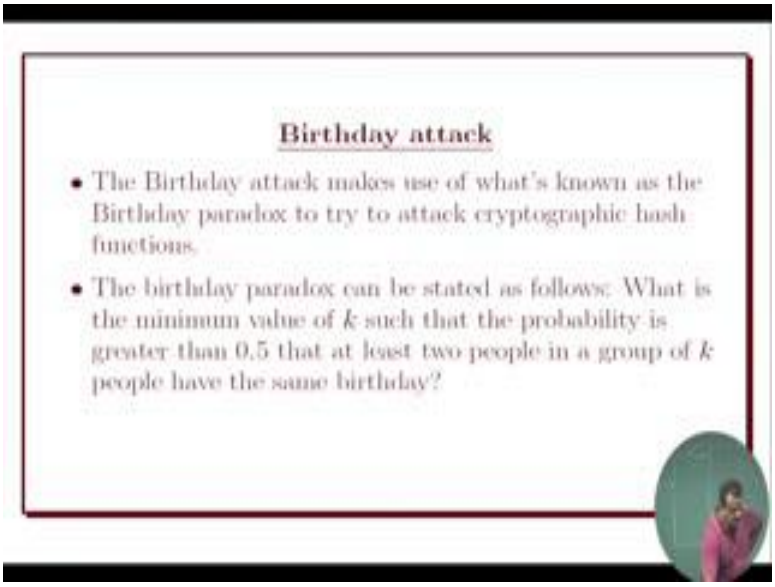
q plus 1 is the intermediate round which is taking this C V q and this and after this summing we are get this is our modified of the MD5. And then finally, modified MD5 is basically CV L.

(Refer Slide Time: 15:15)



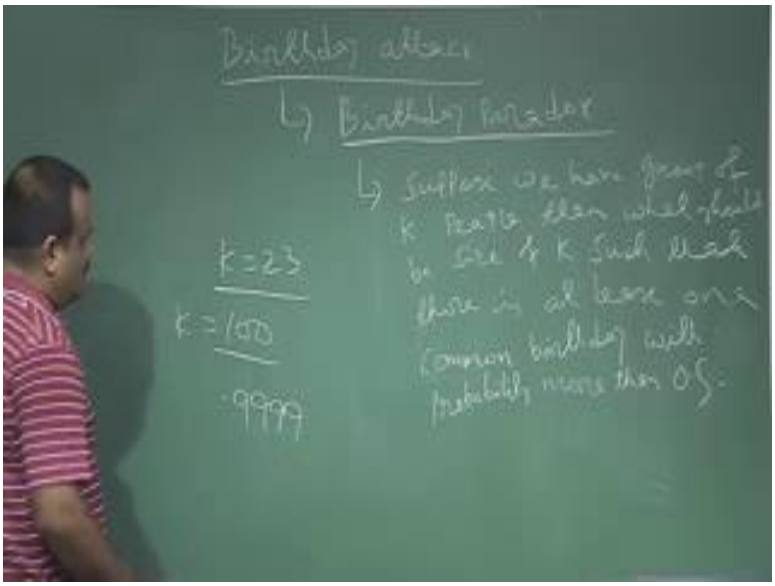
So, these are the intermediate value of this MD5 operation.

(Refer Slide Time: 15:25)



So, now we will talk about what is called birthday attack on hash function.

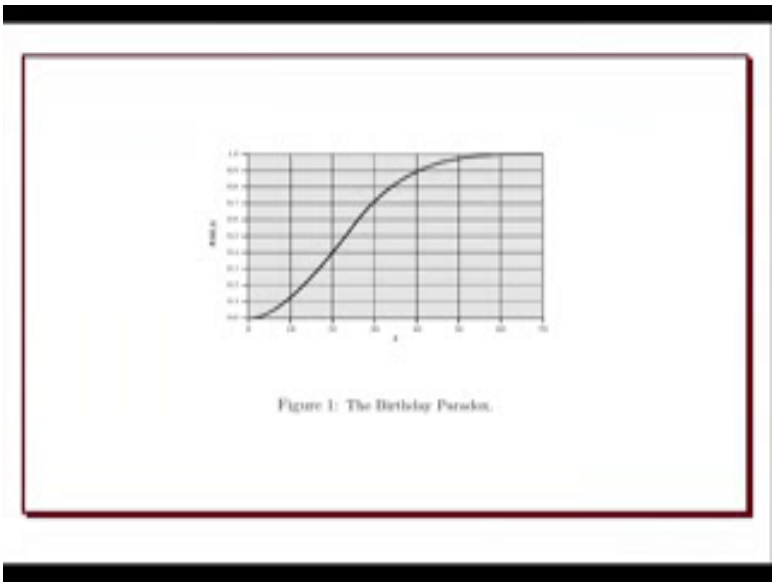
(Refer Slide Time: 15:37)



So, birthday attack is coming from what is called birthday paradox. So, what is the birthday paradox, the birthday paradox is suppose we have some collection of people then what should be the size of that collection, so that at least two people have a common birthday. So, let us state that suppose we have a group of k people or k persons then what should be the size of k such that there will be at least one common birthday with probability more than half. So, this is the birthday paradox.

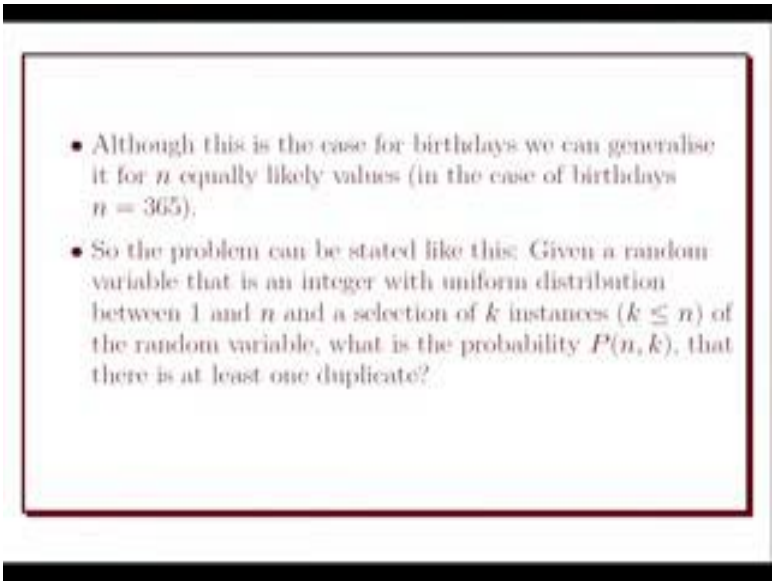
So, this is the question. So, what should be the value of k, we are gathering in a people suppose this is the class. So, we are just we are if we are say if k is 23, then we are done. Then if we have a 23 collections of people group of person then there is a chance and that chance is probability is more than half that two people at least two people is having a common birthday. So, this is called birthday paradox. Now, where from this is coming, now if k is say if we have 100 people 100 collection then this probability will be 0.9999 like this.

(Refer Slide Time: 18:22)



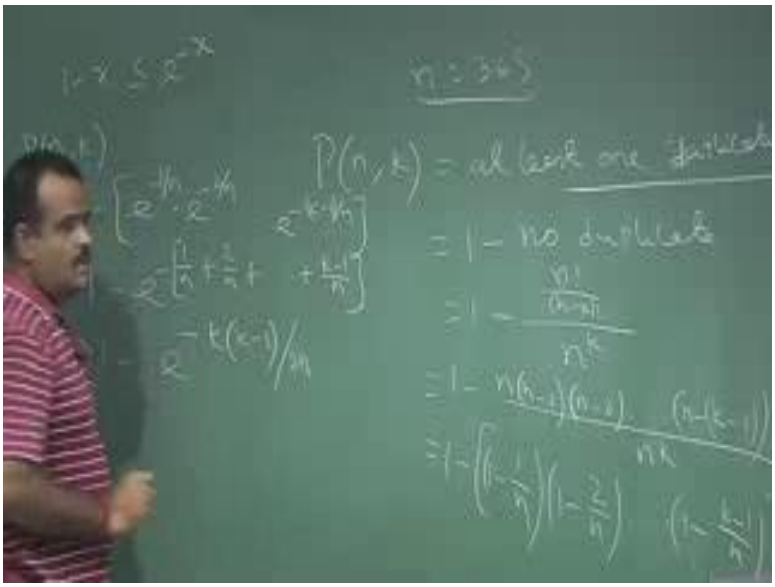
So, now we have a graph over here. So, this graph is telling like this. So, if we are number of group of people are like this, so this is the k and this is the probability, probability of that at least one collision at least one common birthday. So, this is the graph for that. So, if we have say around so this is probability is more than half means this is 23 and if we have more than 70 or in near about 60 people then it is almost close to the 1.

(Refer Slide Time: 19:09)



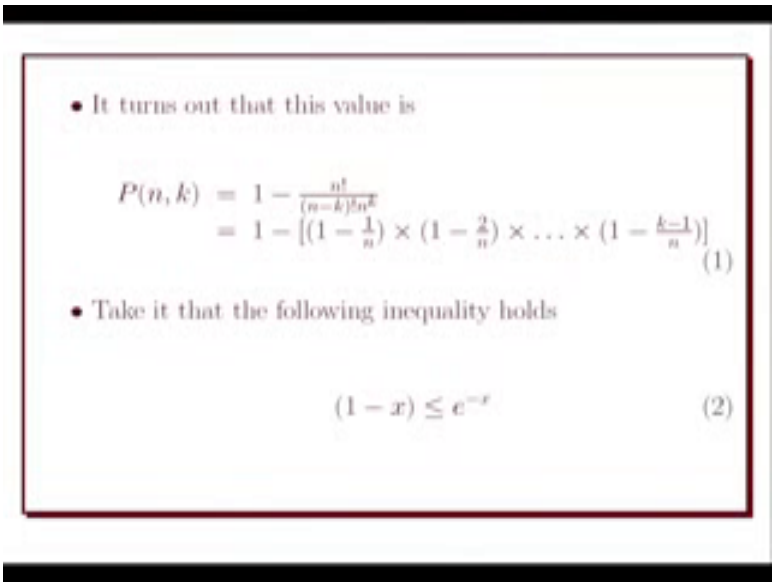
So now, we have to analyze this by probability calculation. So, now how we can how we are saying this is 23 and if it is more near about 70 then it is probability is more, so we have to do some basic probability stuff.

(Refer Slide Time: 19:29)



Now here so in a year how many days is this 365 days. So, among these 365 days, so we are collecting some random numbers. So, suppose our random numbers; so this is the problem. So, given a random variable that is an integer with uniform distribution between 1 to n and then we have a selection of k instant k is less than n the random variable then the probability P n, k that there is at least one duplicate. So, this probability we want to calculate. So, we have given n and we want to from this n we want to select k instance such that this probability n minus k is at least probability of at least one duplicate. So, this probability we want to calculate.

(Refer Slide Time: 20:48)



So, this probability can be calculated like this. So, we want to calculate this. So, this is basically 1 minus probability of no duplicate. So, this is we will do by classical definition. So, we are basically choosing k people and their birthday. So, this is each of these has n possibilities. So, total numbers of cases are n to the power k. So, among this now we want no collision. So, these are the k cell or k people. Now, among this now we want the favorable cases with no duplicate

So, how we can do that now this can be chosen and n ways and once we choose this n ways we do not duplicate. So, this, this, these are the option n minus 1 ways like this n minus 2 like this continue, so it is n minus k plus 1. So, this is basically factorial n by factorial n minus k. So, this is basically factorial n by; so this is basic classical definition of probability. So, this is basically can be written as, so you want to write this as, so this is 1 minus n into n minus 1 into n minus 2 continue n minus k minus n minus k minus 1 in the bracket, so divided by n to the power k. So, these we want to simplify the 1 minus 1 by n into 1 minus 2 by n 1 minus k minus 1 by n.

So, now we want to use some inequality, this is telling us 1 minus x is less than e to the power minus x. So, this we want to use here. So, if we use this inequality then we get. So, this is basically greater than, so p minus this is basically greater than; so each of this is 1 minus 1 by n each of this is less than this so minus of that. So, this must be greater than 1 minus e to the power minus 1 by n into e to the power minus 2 by n into dot dot

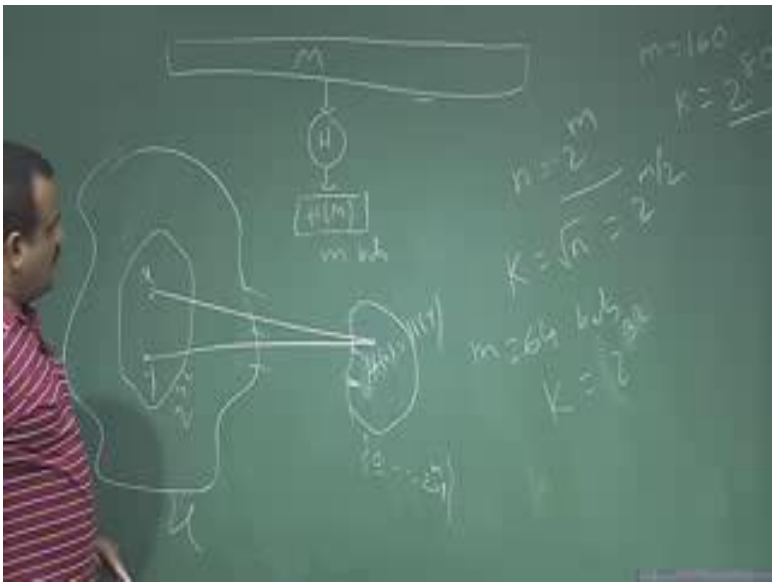
dot e to the power minus k minus 1 by n. So, these we want to further simplify, this is basically 1 minus e to the power minus 1 by n minus we take common then plus 2 by n plus k minus 1 by n. So, this is basically one minus e to the power minus k into k minus 1 by 2 n. So, now we want this probability should be greater than half.

(Refer Slide Time: 24:57)

So, we want this probability should be greater than half. So, for half is greater than 1 minus 2 n so that means, we take e this side k into k minus 1 by 2 n, we want this to be half. Now, this is basically this is the set greater than again half. Now, we take the log both sides. So, this is basically half. So, we take the log both side it will give us log 2 is equal to roughly is equal to 1 minus e to the power minus k into k minus 1 by sorry not, so log two is basically equal to k into k minus 1 by 2 n.

So, this is basically approximately k square by 2 n, this is basically k square by 2 n. So, k square is basically 2 ln 2 into n. So, k is basically so root over of this root over of this value is 1.18, you can calculate this square root of n. So, this is approximately root n. So, k is approximately root n. So, if there are if n is 365, then root n is basically near about 23. So, for birthday case n is 365, so k is 23, so that is the reason if we have a collection of 23 percent then there is a probability that there will be a duplicate birthday. So, there will be at least two people which have a common birthday. So, this we want to use for our hash function. So, how we can use this for our hash function, this birthday paradox?

(Refer Slide Time: 27:43)



So, suppose we have a hash function which is taking a message m arbitral in message m and it is giving us small m bits excuse me for small m bit output. So, this is the set of all message and this is the hash code set of all has hash code, this is very small. So, this is the hash function, this is the hash code, this is 0, 1, 2; this is m bit, so it is from 2 to the power n minus 1. So, this set is bigger, this set is smaller. So, there has to be collision. So, all the thing is we know the weak collision and strong collision, those properties are there for hash function. So, only thing is it is very difficult to get to two keys x, y, the weak collision is given x it is very difficult to find y such that they are colliding. So, this is the weak collision h of x h of y, because this set is very big in general this set is very big. So, it is very difficult to find y. So, you have to search for all y. And for strong collision given a so it is very difficult to find 2.65 such that they are colliding is it computationally hard to find two points they are colliding.

So, now, we want to use this birthday paradox. So, here size of n is 2 to the power m. This is the total power total number of possible hash code. Now if we have k which is root over of n, which is basically 2 to the power m by 2. If we have this many collection then if we search in that collection then we can get then it is ensured that by the birthday paradox that there will be a collision at least one duplicate will be there. So, for example, suppose we have this hash function, suppose m is 64 bit then what is k, k is 2 to the power 32 bit. So, 2 to the power 32 bit. So, it is basically m by 2 to the power m by 2. So, if we can collect a 2 to the power 32 messages, and if we search and we know that by

birthday paradox we know that there will be at least one duplicate that means, there will be at least 2 x, y such that they are colliding into the same function. This is the common birthday or they are colliding.

So, we are reducing our search phase by instead of 2 to the power 64, we are reducing it to two to the power 32. So, if we can search in 2 to the power 32 collection then there is a probability with half that will get this x, y. And searching to do in 2 to the power 32 is not a big deal in a modern computer. So, this is the attack one can mount for hash function, so that is why m should be reasonably good size. So, for SHA-1 m is 160 bits, so that means, k is 2 to the power 80, which is reasonably difficult to search. So, this is the way how we can mount the birthday attack on the hash function.