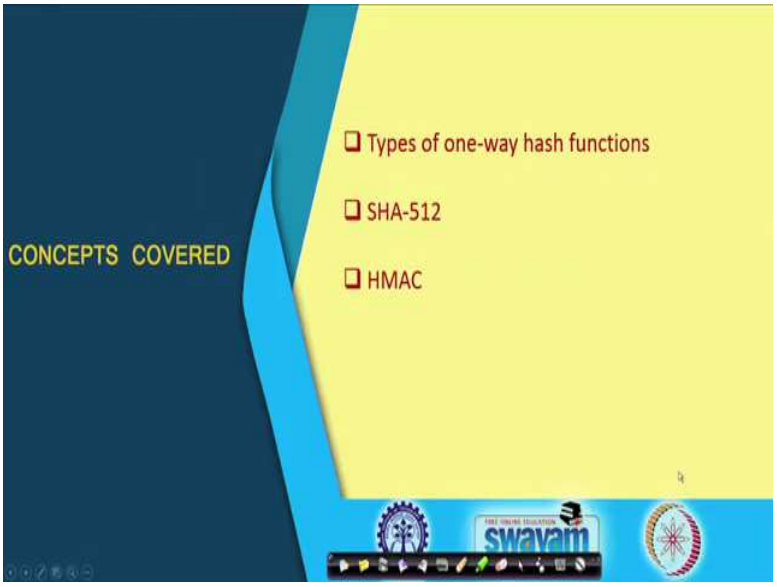
Ethical Hacking
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 32
Cryptographic Hash Functions (Part II)

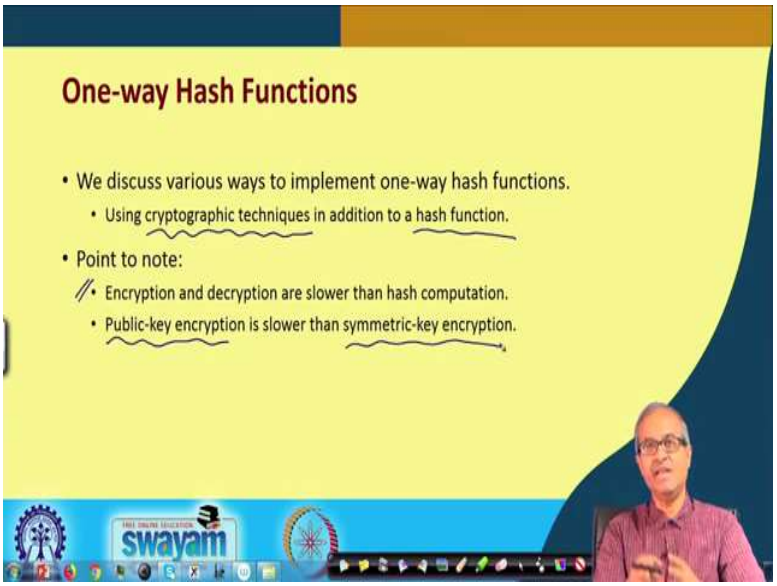
Now, we continue with our discussion on Cryptographic Hash Functions. If you recall in the previous lecture we had basically talked about the basic properties and requirements of a hash function. In particular the hash functions which will be required for security or cryptographic applications. We shall be continuing with our discussion in this lecture, this is the part II of the lecture.

(Refer Slide Time: 00:38)



So, in this lecture we shall be first talking about several ways this one-way hash functions can be used and specifically we shall be looking at two different hash functions: one is the SHA-512; other is HMAC. Let us see.

(Refer Slide Time: 00:58)



Well talking about one-way hash function, if you recall, we mentioned because a hash function maps a larger set into a smaller set, there is a many to one mapping. You cannot have a unique reverse mapping from the hash value to the message. That is why we call it a one-way hash function. Now, in computer systems for example, when you store the passwords there also this kind of one-way hash functions are used.

So, when you create a password, the passwords are not stored in the computer, because if you store the password in plain text, well anyone who gets access to the system can steal the passwords.

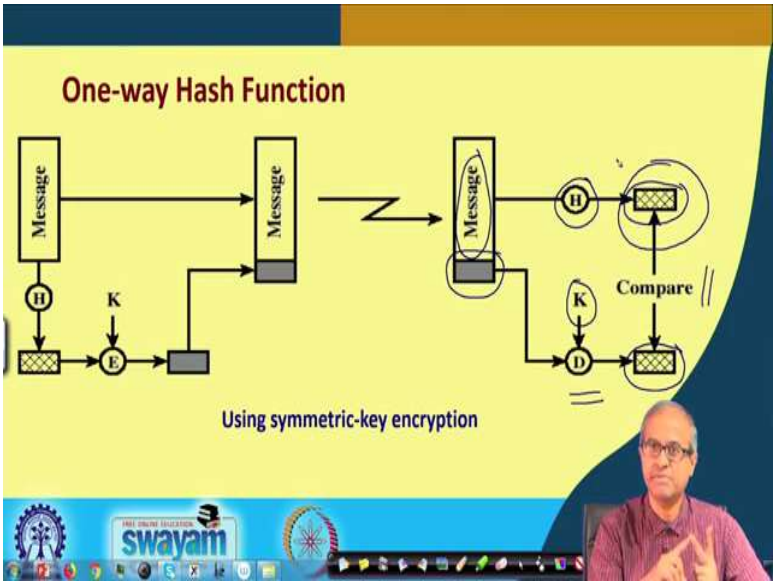
So, what is done? Some kind of one-way hash function is done and the hash value is stored in some table. So, whenever you are logging in you are typing the password, again that one, our hash function is applied on your password and then you compare against the table whether it is matching with any one of the hash values. That is how it is used. Well, talking about two one-way hash functions. Here now we shall first look at different ways to implement it. We said that we can use some encryption or decryption algorithms to implement a, implement authentication.

So, we shall look at that and then even without encryption/decryption methods. So, we will be using some cryptographic techniques like symmetric-key or public-key encryption in addition to hash functions, but the point to note is that you should clearly understand that why you were looking at so many varieties and alternatives, because the bottom line

is that we need a hash function which will be efficient to compute both in software and possibly also in hardware, if you think of a hardware implementation.

Now, this encryption and decryption which possibly can also be used for authentication is, they are much slower than hash value computation. This is the first thing you should remember and the second thing which also we mentioned earlier public-key encryption is much slower than symmetric-key encryption. So, public-key encryption is the slowest, then symmetric-key encryption, then hash function computation which is the fastest. Let us see, ok.

(Refer Slide Time: 03:34)



So, first in this diagram we try to explain how we can implement a one-way hash function using symmetric-key encryption? Let us see. Now here you have a message. So, what we do; we have a hash function H here. So, we apply a hash function right and we get a hash digest out here.

So, we have a hash function, we apply to a message, we get a hash value, but after that we use another step of encryption, we have an encryption step where you use a private key K and a private key or symmetric-key encryption algorithm to encrypt the hash value. We are not encrypting the message. We are encrypting this small hash value to generate an encrypted hash.

So, what we are transmitting along with the message is the encrypted hash. So, the advantage is that any intruder who is getting hold of it, because it does not know the value of K; so, the encrypted hash he or she cannot access or process. At the receiving site something similar is done. So, the message along with this encrypted hash is received.

So, with the message part again the hash function is applied, the hash digest is obtained and with the encrypted hash part we apply a decryption algorithm with this same key value K. So, we get back the hash value and you compare whether these two are matching or not. Well, if this two are matching, we get two things together; one is of course, now means, we can identify the sender, because only the secret key K was shared with the sender.

So, because we are able to verify, so, it must be the case that the same value of K was used. So, the sender has been authenticated. Now, in addition, another thing also we will, because we achieve here is that we also verify the integrity of the message that the message was not modified during transmission, because if there is a modification in the message, then this hash value would have changed and so, the comparison would have failed, ok.

So, two things were achieving together; authentication and as well as message integrity ok. Let us see. Similar thing we can do using public-key encryption, because we had said earlier that one drawback of symmetric encryption was that both the parties must be sharing the same key. So, the key must be sent to the other party through some, means, that is an initial step.

So, one way hash function is very similar here. In this case with the message you again apply the hash function, you get the hash digest. Well, you again do encryption, but this time you use a public-key encryption technique like RSA where you use the private key of the sender to encrypt. I use my private key to encrypt the hash value. This will be the encrypted hash value which is appended to the message and is transmitted.

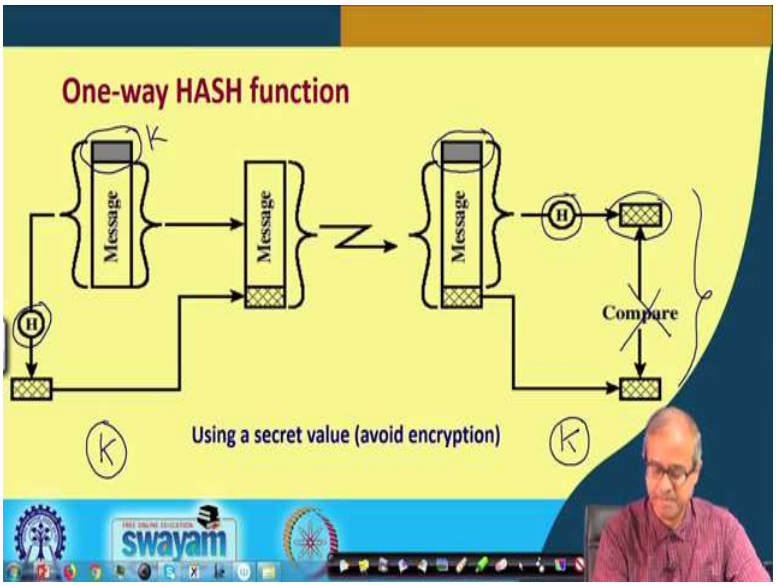
When you see sometimes when someone encrypts with the private key, well using public encryption which sometimes say that it is a signature process, because I am only having my private key, encrypting the private key with the private key can be done only by myself as if it is carrying my signature something similar to that.

At the receiving end something similar is happening. Firstly, with the message part the same hash functions is applied and with the encrypted hash part a decryption algorithm is run with the public-key of the sender and again we do a comparison. So, if there is a match again just like the previous one and using symmetric-key encryption, here also we can verify two things together.

Firstly, that it must be signed by the intended sender, because, otherwise I could not have decrypted it using the public-key and secondly, the message is also not modified. There is no loss in integrity because of that I could get back the correct hash value here, ok.

These two methods look very fine, but the only downside is that I am having to use encryption and decryption algorithms at the two ends and as I said encryption and decryption are slower as compared to hash function computation. These methods are good alright, but they are becoming a little slower.

(Refer Slide Time: 09:12)



There is another approach we are trying to use here, where we are trying to avoid encryption. What you are saying is something like this, let us assume that we are having some kind of a secret value K like a key shared by the sender and receiver, but we are not using this value K for encryption and decryption. What we are doing; we are appending this K with the message, let us say this is, this K.

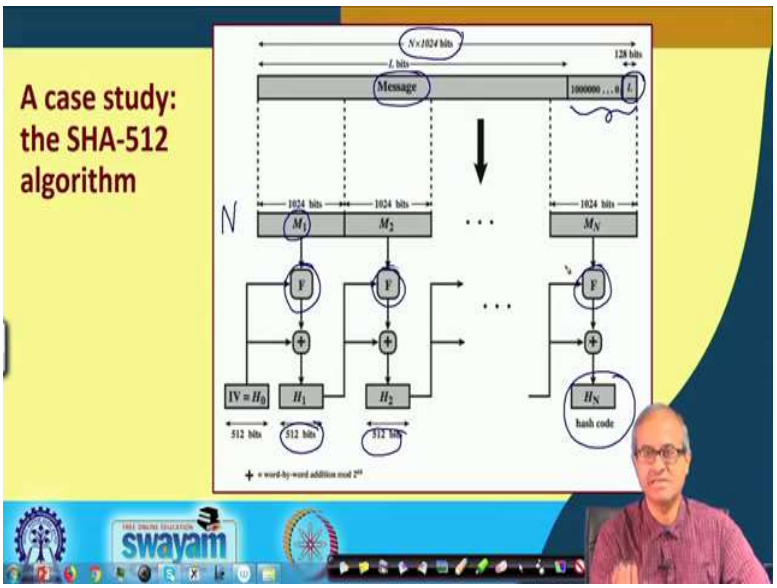
The message is there with us. I append that secret value with the message and for the whole thing I applied the hash function. I apply this hash value or hash function on this whole thing $message + K$. So, what hash function I get that also carries a flavor of K with it.

So, unless I have the current value of K, I cannot generate that hash function. So, I sent the message only the message not the K, but along with the hash function I compute here. At the other side something similar happens. The message is received, but the same value K was also present to the receiver. The receiver again appends that value K with the message and applies the hash value and gets a hash function, then it compares.

See here also we are achieving both the things, but without carrying out any encryption, because the key value is shared by the two parties. Both the sender and receiver know. So, if the comparison gives a success, it means that the message must be coming from the intended sender and also if there was some modification, the message must be, the hash value would have changed.

So, here there would be no match, ok. So, I can achieve both the things but here I am avoiding this step of encryption, ok.

(Refer Slide Time: 11:21)



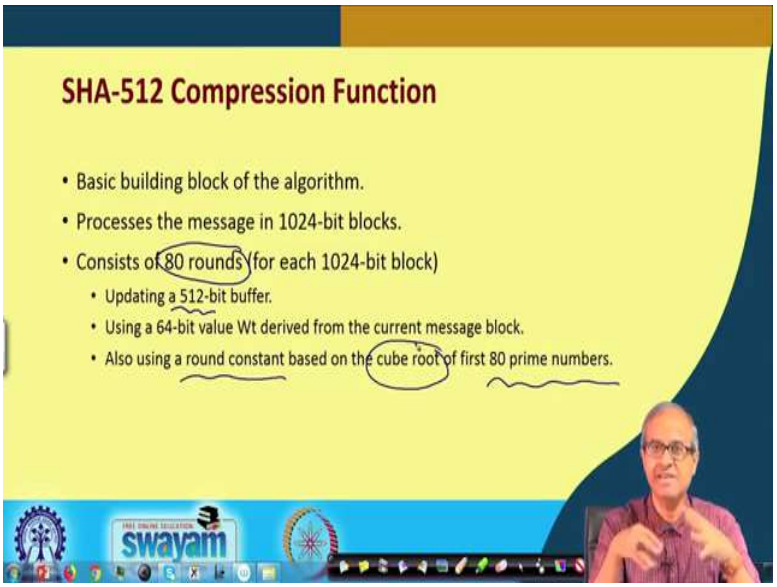
So, let us now look at a case study. I talked about the various hash families. SHA is one such very popular hash family and SHA-512 is one of the strongest member in that family. Now, very broadly speaking, so, you see I am not going into detail, ok. Just to look at it

what you are doing here, I have the message that I want to compute the hash value on. So, what I do? The first thing is that the total bit string that I am trying to hash, must be a multiple of 1024 bits that is a restriction with SHA-512. So, if it is not a multiple of 1012, then some additional pad bits are added.

What does the pad bit contains? It contains first the length of the message at the end, L. It contains the value of L. How many bits in the original message was there and then the pad consists of a single 1 followed by all 0s. You do it to make it the whole thing a multiple of 1024 fine. Then this each of these 1024 bit chunks, they are called, it is a M_1, M_2 to M_N. There are capital N such data blocks or chunks. Now, each of them undergo the SHA-512 algorithm and by virtue of this a 512 bit of data is generated.

But to see how it is done, the first is M_1 generates a 512 bit data, but this, this hash value, this hash value is fed as input to the second stage that creates another 512, this proceeds and finally, we get a single 512 hash value, the final value whatever comes that is taken as the 512 bit hash. So, you see this is a fairly complicated process and this function F, this itself is very complicated that is why it is not very easy to break this, this one-way hash function.

(Refer Slide Time: 14:05)

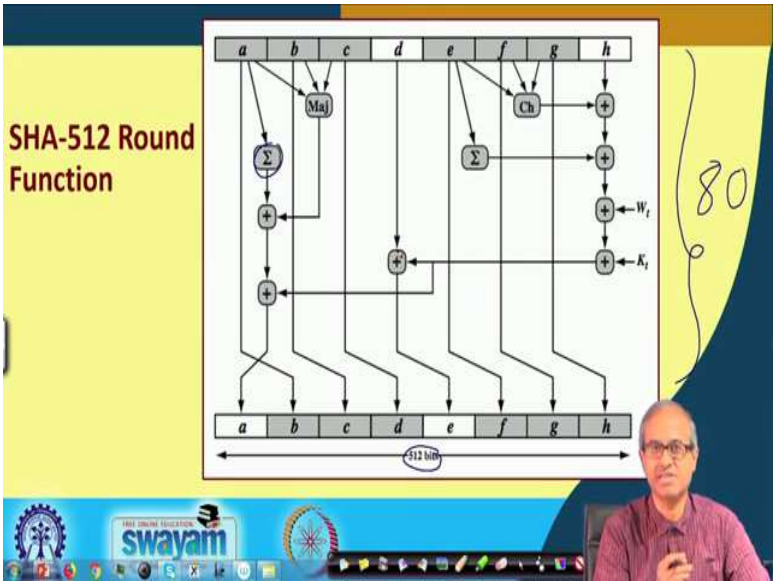


So, the compression function means this F function we talked about; this F function is the basic building block of SHA-512 and as you have seen in the previous diagram that function F takes a block of the message which is 1024 bits in size, ok, and inside that F

there are 80 iterations going on not 1, not 2 but 80 iterations or 80 rounds for each 1024 blocks and inside they maintain a 512 bit buffer which is updated at every step of these 80 iterations.

I am not going into the details of these 80 iterations. Details are available in a, any, you can say in any standard textbook or if you search in the internet you will also find and it uses a 64 bit value, derived from the message block and also use a round constant in each of these 80 rounds and these round constants are generated from the first 80 prime numbers. You compute the cube root of each of them and the result of that cube root whatever is obtained, those are taken as a round constants. So, you see the things are fairly complicated inside, ok.

(Refer Slide Time: 15:40)

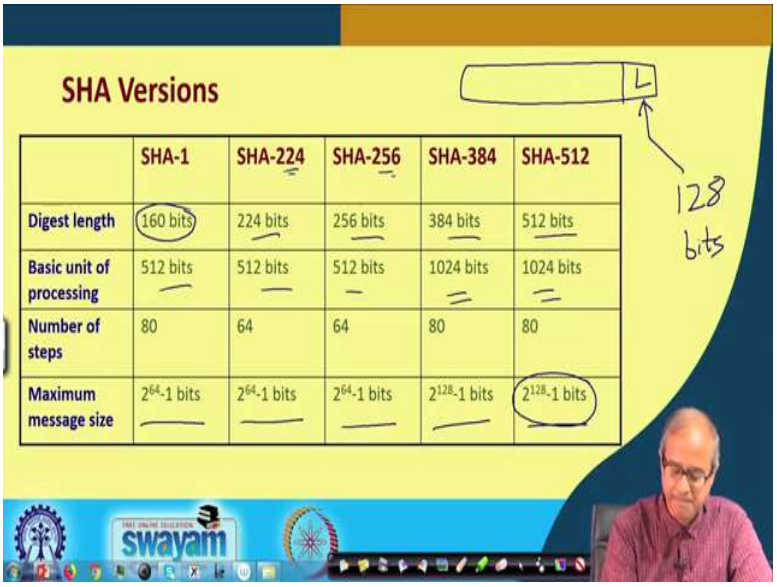


So, just bird's eye view of the round function. What is happening inside that function F. So, I am not going into the details of the explanation. This is the 512-bit buffer I was talking about. There are some changes or modification that are going on and this iteration is carried out for your, this 80 times. The lot of operations. There is a sigma operation. There is a majority operation.

There are some bitwise XOR up on this, plasmids bitwise XOR operations. There are many such operations going on inside and everything is iterated 80 times. Only after that, you get the final that 512 bit digest that you are obtained and that is going into the input of the

F for the next one, that is going to the input of the next one and finally, you get back a single 512-bit digest which you take as the final hash value, right.

(Refer Slide Time: 16:51)



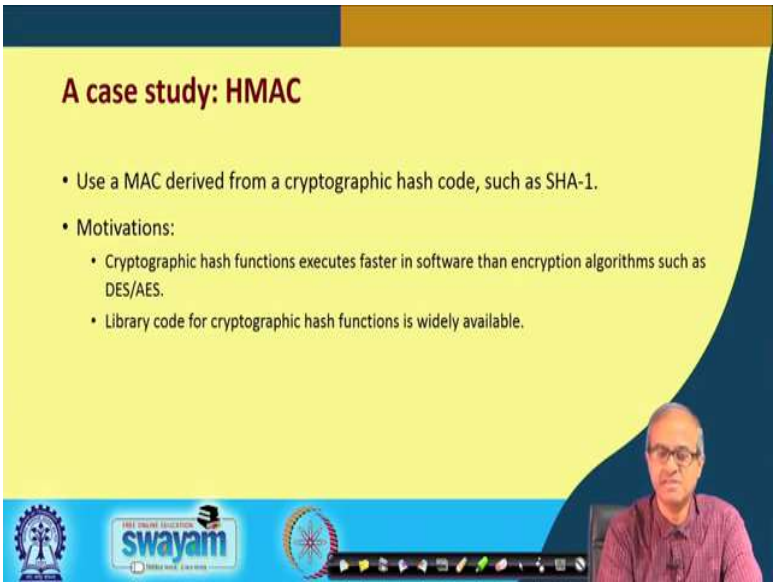
	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Digest length	160 bits	224 bits	256 bits	384 bits	512 bits
Basic unit of processing	512 bits	512 bits	512 bits	1024 bits	1024 bits
Number of steps	80	64	64	80	80
Maximum message size	2^{64}-1 bits	2^{64}-1 bits	2^{64}-1 bits	2^{128}-1 bits	2^{128}-1 bits

Now, talking about the different hash versions of SHA. In this table I have just summarized five different version, there are other versions also in between: 1, 224, 256, 384 and 512. They are distinguished primarily by the digest length, SHA-1 is the most basic version, it uses one; sorry, it uses 160 bits of hash value, final hash value, but the other ones this number indicates the size of the hash. So, SHA-512 indicates that the final hash values 512 bits and basic unit of processing the message is broken up into smaller chunks.

The first three uses 512 bits, the last two uses 1024 bits and number of rounds in that function F; these are very similar in their construction 80, 64, 64, 80, 80 and, as I said earlier that the message size cannot be unlimited, there is an upper limit; now the upper limit is very large in fact. For the first three it is $2^{64} - 1$. It is a huge number, for the last two it is $2^{128} - 1$ which means you recall in that message that you are padding, at the end you put the message length, ok.

Now, this $2^{128} - 1$ means here this message length will be 128 bits. So, the number of bits reserved for this. L is fixed. That determines the maximum size of the message, fine.

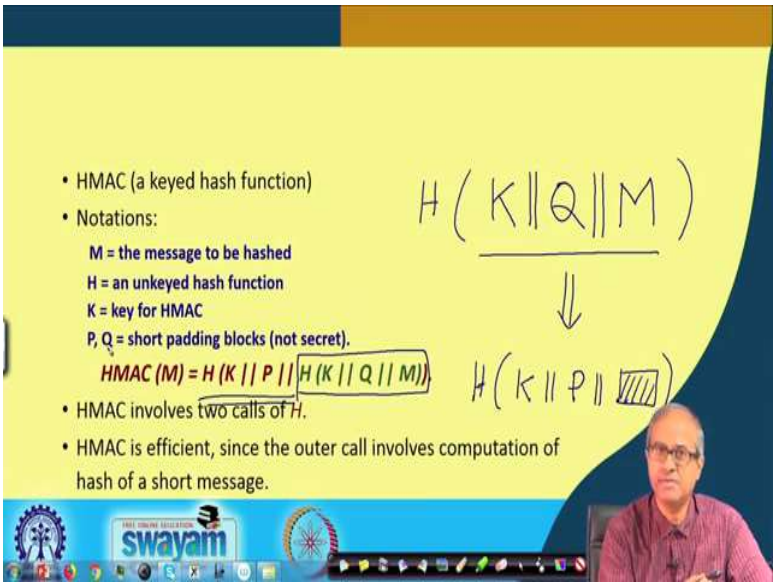
(Refer Slide Time: 18:56)



Now, let us look at another kind of a hash function which is called HMAC. Well, HMAC is a method to generate a message authentication code, but it does not use any encryption. It is derived from a cryptographic hash function like SHA, some shuffling. Let us, let us say SHA-1 simplest. Motivation behind this is because, you see they MAC algorithms we talked about earlier, they uses a secret key and they uses encryption and decryption, but here I am repeating.

We are trying to avoid the encryption process because encryption is slower and if you use only hash functions in the computation; since they run faster, they will be quicker and such hash functions are also widely available.

(Refer Slide Time: 19:50)



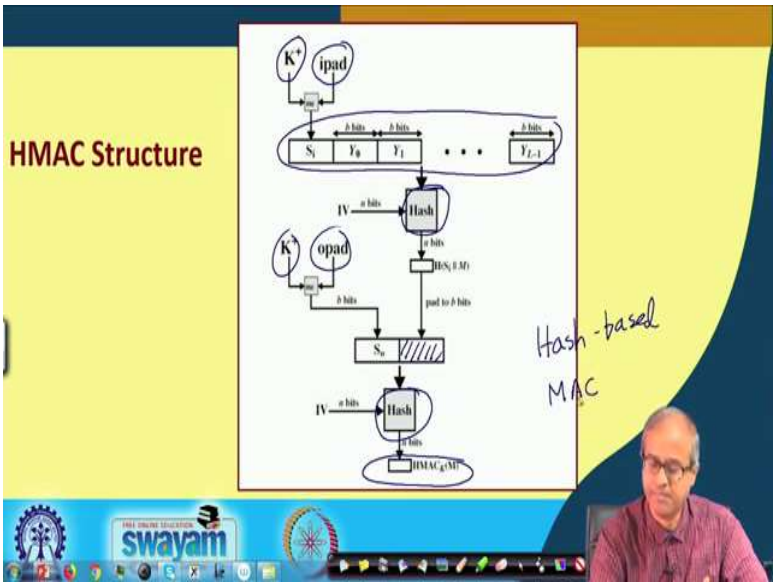
Now, the way HMAC, this HMAC works is as follows. Let us say M is the message. You want to generate the MAC or message authentication for file. You want to apply the hash function on the same. So, you apply the hash function in two steps. This is the first step where you have the message M, you have a short padding block Q, you add Q to this M.

There is a secret key K. Just like an encryption/decryption you have a secret key also, but you are not doing an encryption. You also put this K. You append these three things together, you put them side by side together and then you apply the hash function on this whole thing.

This is the first step of it and if we apply the hash function, you will be generating a hash digest. If it is SHA-1, it will be 160 bits, ok. This is not the end of the story. You apply another cycle of hash. In the next cycle you take this hash value as your last part. This is the computed hash value. Then you have the another short padding block P. Then that same K key, this you concatenate again, put them together and apply hash on this whole thing. This will be your second round of hash calculation.

This is how this HMAC works. For added protection, it does or computes hash function two times, it uses a secret key. In addition, it uses some random padding blocks P and Q right. Because it does not use the encryption, it is efficient to compute.

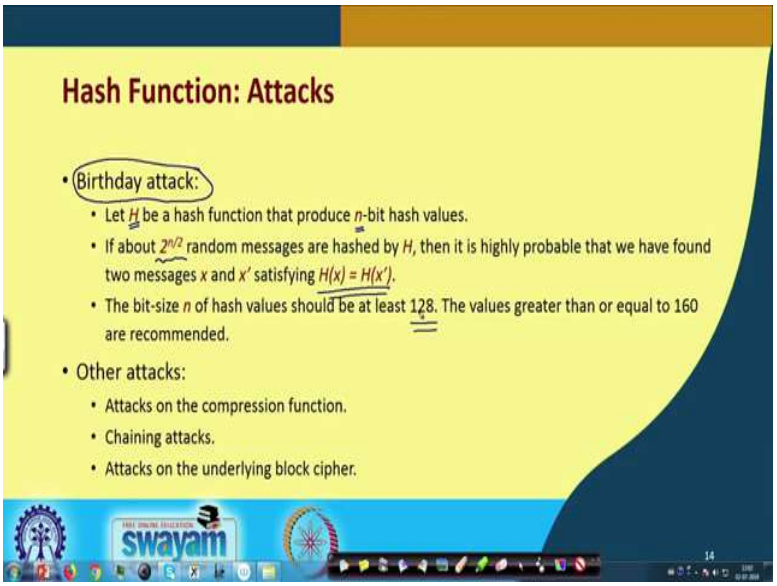
(Refer Slide Time: 22:03)



So, I am showing this pictorially. So, here there are two hash function computation. One is here; other is here, right. Now, you can see this padding is done. These are the initial bits. You are dividing them into different blocks and one block at a time is being fed to this, to this particular hash function. So, just using the same principle there is a key. There is a pad that P and Q you are using. There is a key. There is a pad. In both these steps you are using that.

In the first step you are talking about the whole message. So, the pad is appended only once, in the second step you are talking only about the hash function generated in the first step and you add a pad to it, then you apply the second hash, ok. So finally, whatever you get that is your HMAC value, hash based MAC; HMAC is the full form for Hash Based Message Authentication Code, fine.

(Refer Slide Time: 23:20)



Talking about the attacks on hash functions, well whenever there are cryptographic algorithm, there are people who also try to attack and break them. Now, in hash function the main issue is collision, how difficult or how easy is to find another message so that the same hash value is obtained.

Now, there is a well known kind of an attack on hash function, which is referred to as birthday attack. Well, again I am not going into detail, because it is a little beyond the scope of this discussion. I am just telling you the basic idea. Let us say H is a hash function that generates n bit hash digest values. Birthday attack says, if you can generate $\frac{2^n}{2}$ random messages.

Say, hash values are much smaller, $\frac{2^n}{2}$ is even smaller, not 2^n. I am talking $\frac{2^n}{2}$. Random message or hash phi H, then there is a theory which says that there is a high probability that you have two messages x and x by x'. You have generated with this same hash value. There is a theory behind it, ok. Number theoretic proof is there.

Now, because of this birthday attack, this is possible. So, the number of bits in n must be at least 128, because if you use a smaller value then the birthday attack can be mounted and you can easily break it. To make it more difficult, you must use more number of bits. That is why I said SHA-512 uses 512 bits of the key, it is much larger.

So, it is much more difficult to mount this attack, ok. There can be other kind of attacks also. So, here again I am not going to detail, you can try to attack that function, F compression function. There is something called chaining attack and whatever block cipher you are using, if you in an encryption method in which you can try to attack that; there are many kinds of attack which have been reported but with limited success.

If your underlying algorithm is good, if you are using a large enough secret value then you can expect that it will not be easy for anyone to break your code, ok.

So, with this we come to the end of this lecture. So, we had actually talked about various different kinds of hash function and then use in both authentication and ensuring message integrity. So, the next lecture we shall be looking at some other applications of these kind of hash functions.

Thank you.