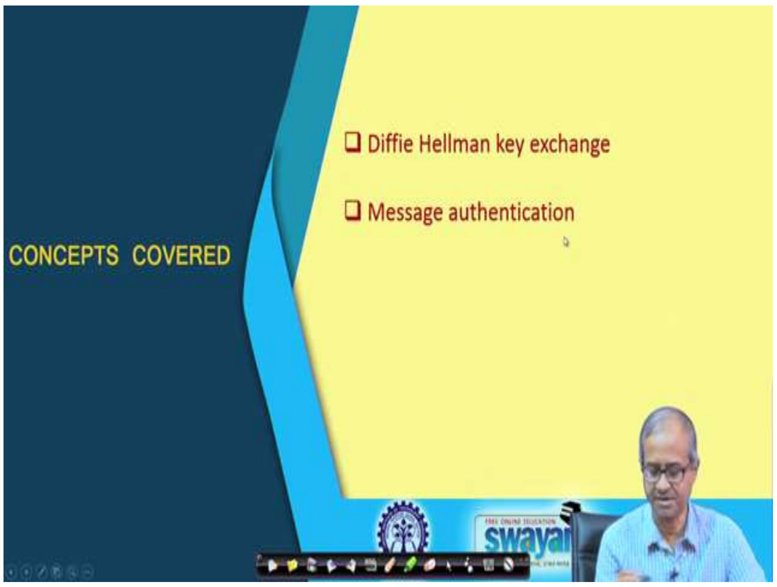
Ethical Hacking

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

Lecture – 30
Public – Key Cryptography (Part II)

In the last lecture if you recall, we were discussing about generally the public key cryptography, the concept and in particular the RSA algorithm. So, we continue with our discussion in this lecture so, where we shall be looking at some other cryptographic public key cryptography schemes and some of the issues.

(Refer Slide Time: 00:37)

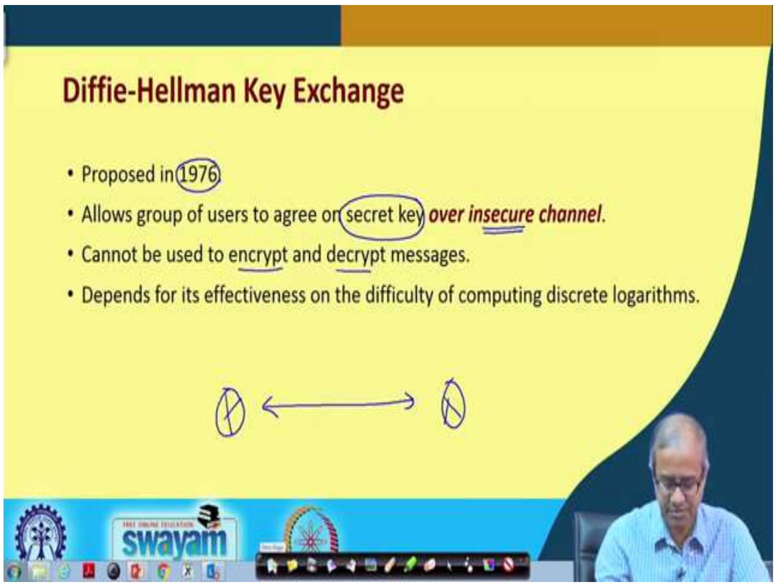


CONCEPTS COVERED

- ❑ Diffie Hellman key exchange
- ❑ Message authentication

So, what we will be looking at first, is something called Diffie Hellman key exchange which is also a public key kind of an algorithm and lastly, we shall be briefly talking about something on message authentication which of course, we shall again be coming back to later.

(Refer Slide Time: 00:58)



Diffie-Hellman Key Exchange

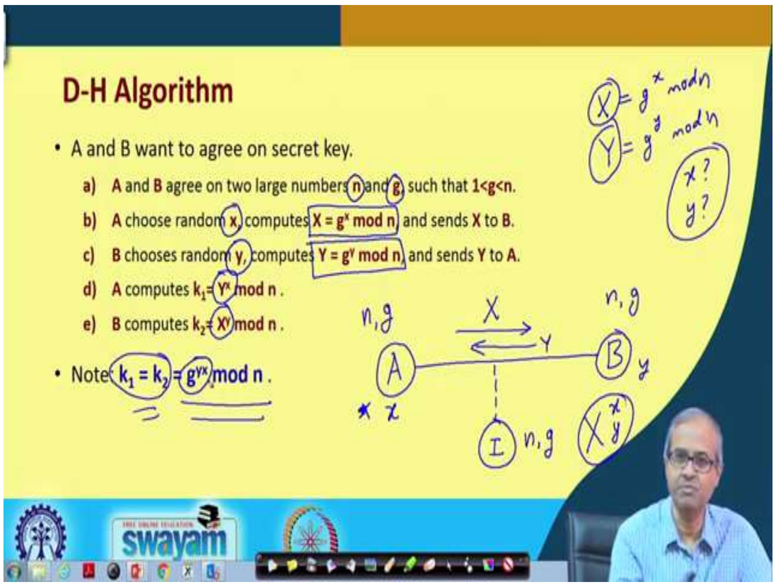
- Proposed in 1976
- Allows group of users to agree on secret key ***over insecure channel***.
- Cannot be used to encrypt and decrypt messages.
- Depends for its effectiveness on the difficulty of computing discrete logarithms.

So, Diffie Hellman key exchange is a very conceptually simple method for exchanging key between two parties; it does not use RSA, but a different kind of an algorithm which was also proposed long back in the 1970s, 76. So, here the idea is that two parties, two parties can exchange a set of messages and finally, they can agree on a secret key.

And this channel over which they are exchanging, this is assumed to be insecure. Insecure means the hackers may be able to hear whatever is going on. So, you should not be sending something using which the hacker will be able to know that what secret key we are finally, agreeing on. So, Diffie Hellman key exchange is one such algorithm which allows us to do that. But this is not an algorithm for encryption/decryption, just make it very clear. It is only used for agreeing on a common shared key. Just like we mentioned in the last lecture that you can use the RSA algorithm to generate an unknown number and send it to the other side and that random number can become the key, ok.

Not like that, it is something similar we do here. Now for RSA, you recall, we mentioned that the difficulty of breaking RSA is based on the computational difficulty of factoring the product, have two prime numbers. Diffie Hellman key exchange is also based on a similar mathematically complex problem. It is called discrete logarithm problem. This we shall talk about a little later. Let us look into the algorithm first.

(Refer Slide Time: 03:06)



D-H Algorithm

- A and B want to agree on secret key.
  - a) A and B agree on two large numbers n and g such that $1<g<n$.
  - b) A choose random x, computes $X = g^x \bmod n$ and sends X to B.
  - c) B chooses random y, computes $Y = g^y \bmod n$ and sends Y to A.
  - d) A computes $k_1 = Y^x \bmod n$.
  - e) B computes $k_2 = X^y \bmod n$.
- Note: $k_1 = k_2 = g^{yx} \bmod n$.

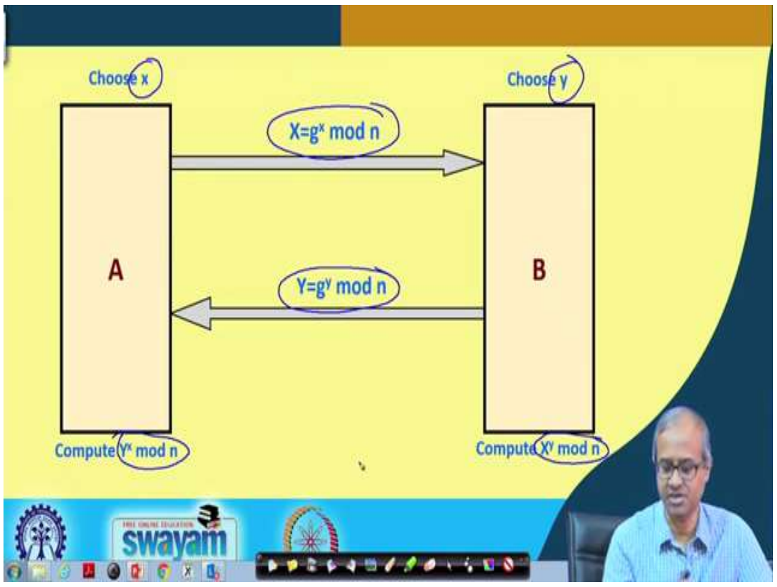
Diffie Hellman algorithm, the way it works, let us say there are two parties A and B which want to agree on some common shared key and there is a communication channel between them. I am assuming that this communication channel is not secure. There can be some eavesdropper. There can be some intruder which may be listening to whatever is going on in the channel. This is our model and assumption. First thing is that A and B agree on some two large numbers n and g. Of course, $g < n$. So, this n and g values are available to both A and B and this n and g has something which A has communicated to B over this channel. So, the intruder may also be have, they have these values of n and g. So, n and g on nothing which are very secret, I am assuming that the world knows because I have sent it over in securer channel.

Now the steps in the algorithm, it will, A chooses a random number x. These numbers are all larger number. So, I am not mentioning it separately, but these are all large numbers. A chooses a number x. Let us say small x, small x and it calculates $X = g^x\ mod\ n$ and this capital X is sent to B. B does a similar thing. B chooses some other random number y and it calculates $Y = g^y mod\ n$ and sends it back to A.

Now you see, you think of the intruder, say intruder knows X which is $g^x\ mod\ n$. It knows Y which is $g^y mod\ n$ because these are the numbers which have been communicated over the channel, which I is hearing. Now the discrete logarithm problem says that if I know g, if I know n, if I know X then what is the value of x? Similarly, for the other case if I know g, n and capital Y, what will be the value of y? This is the discrete logarithm problem which is known to be computationally very difficult. So, this Diffie Hellman key exchange algorithm is based on this assumption that even if the intruder knows X and Y, intruder will not be able to find what are the values of x and y, small x and small y, ok.

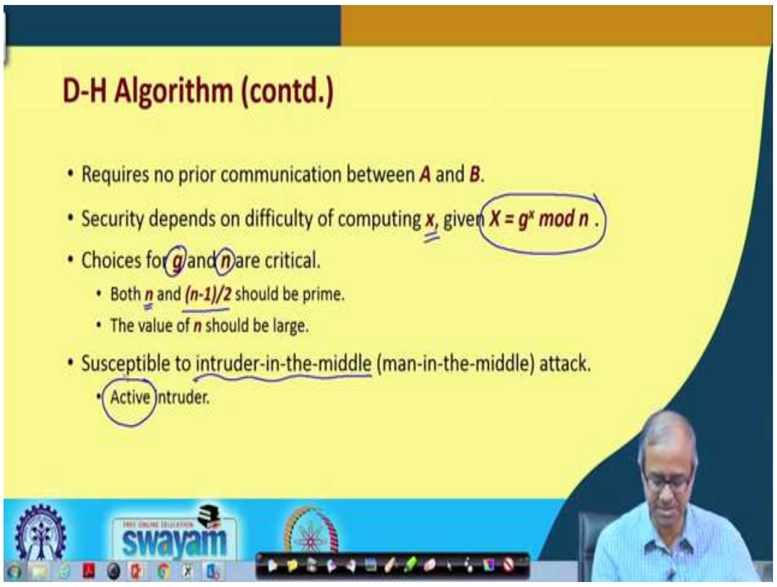
Now, after this stage what happens when A receives this Y, capital Y from B. So, it does again a Y^x, small x and B receives x and does a X^y. So, at the end what will happen? Both A and B are having the same value g^{xy}, but I, this poor fellow cannot do this. I will not be having g^{xy}, because I is only seeing this capital X and capital Y which are flown through the channel, but small x, small y values, these are only lying with A and B and finally, thought A and B will be having the value, same value of key which is given by $g^{xy}\ mod\ n$. This is how key exchange happens.

(Refer Slide Time: 07:31)



Pictorially I am showing it like this, same thing which I have just now mentioned, ok. So, A chooses small x. B chooses small y. A sends this capital X to B. B sends capital Y to y. Finally, A computes $Y^x mod\ n$ and B computes $X^y mod\ n$. Finally, they both get the same value of key. This is how Diffie Hellman key exchanged works.

(Refer Slide Time: 08:15)



D-H Algorithm (contd.)

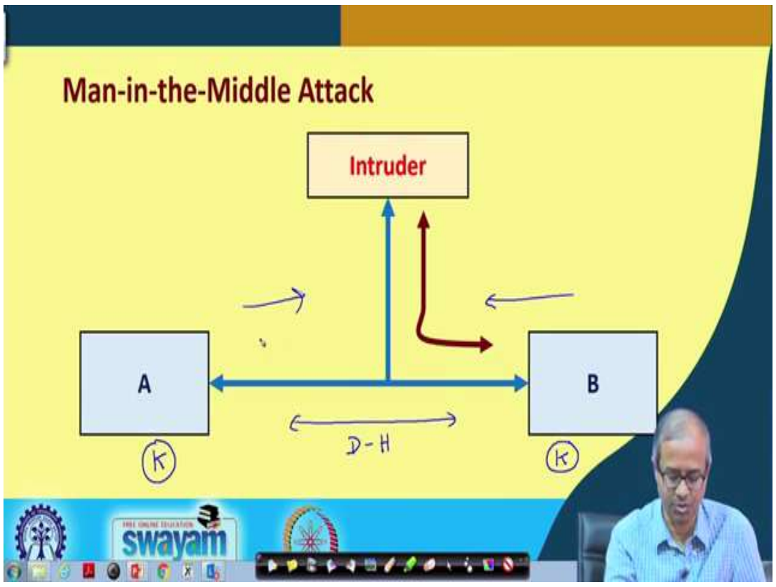
- Requires no prior communication between *A* and *B*.
- Security depends on difficulty of computing *x*, given $X = g^x \bmod n$.
- Choices for *g* and *n* are critical.
  - Both *n* and *(n-1)/2* should be prime.
  - The value of *n* should be large.
- Susceptible to intruder-in-the-middle (man-in-the-middle) attack.
  - Active intruder.

So, this method is interesting, good in the sense that there is no prior communication required between the A and B. They do not need any public key like in RSA to be known by the other party and so, on. So, from scratch you are starting on communication and you are agreeing only the value of g and n has to be sent at the beginning and after that they can agree on the common key, shared key. And as I had said the security depends on the discreet logarithm problem that given known value of X, g and n.

What is the difficulty of computing small x? That is and just one point is that I just mentioned g and n are chosen initially, but there are some issues or constraints you need to follow when you are choosing the values of g and n. Like the value of n should be such that n and also $\frac{n-1}{2}$ should be prime, because some attacks have been discussed by mathematicians where they said that if such constraints are not satisfied, then it may be feasible to attack Diffie Hellman system and since of course, the value of n should be large. But one problem here is that there is a man in the middle attack or intruder in the middle attack, where there is an active intruder. Here Diffie Hellman key exchange might mislead itself.

Like I am just telling you the basic idea.

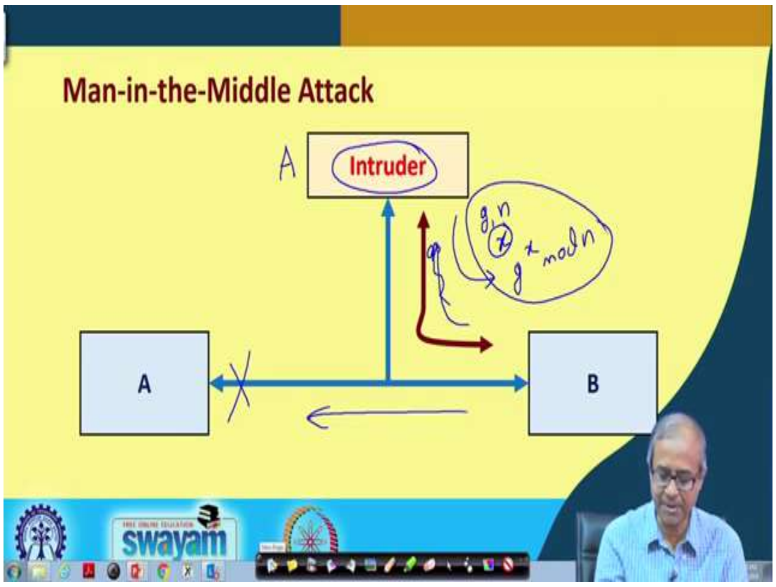
(Refer Slide Time: 10:14)



Man-in-the-Middle Attack

Just you think of a scenario like this, this A and B are sharing a communication channel and there is a intruder who has been looking at the message exchanges and have understood what kind of message exchange are going on. So, normally what would happen is that, this A and B would be using Diffie Hellman key exchange between themselves and would agree with the common shared key k between them. Then they can exchange messages, use an encrypted way. Using this message key B can send something to A, A can send something to B and so, on.

(Refer Slide Time: 11:07)

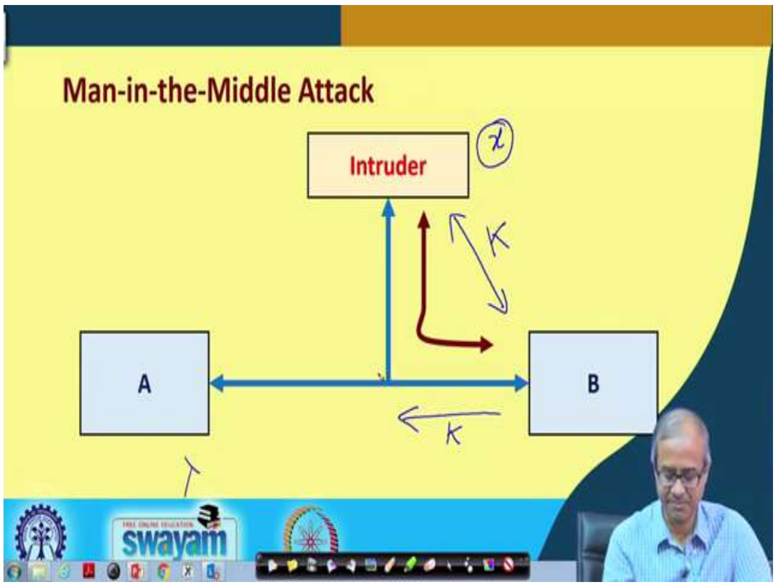


Man-in-the-Middle Attack

Now, suppose what happens is something like this. This A is not sending anything right now, but B is fabricating a message.

This intruder, this intruder is fabricating a message. What the intruder will do? Intruder will act as if intruder is A. So, intruder will be sending a packet to B with the initial values of g and n, but these are all fabrication attack, attacks and then again it will choose a number x and we will send the value of $g^x \ mod \ n$ to B. So, B will do the same thing. B will in good faith assume that this is A. It will send it back. It will not send it back here. It will send it back to A, but this intruder will also capture that same packet. But the first message was initiated not by A, but by the intruder. So, intruder has the value of x. B does not know that B is sending a similar packet to A which the intruders is also capturing.

(Refer Slide Time: 12:32)

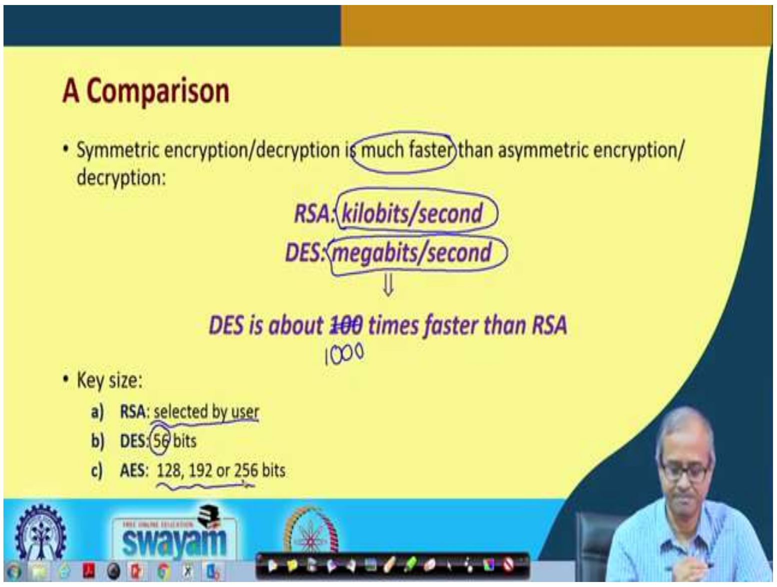


Man-in-the-Middle Attack

So, what will happen at the end is that.

The key k that has been shared will be shared between B and intruder, not between B and A, because the small x was generated by intruder. A does not know about small x. So, now, B whatever sends in an encrypted way using key k, intruder can intercept. It can decode it back. Some message privacy might get compromised. So, this is called, something referred to as man in the middle attack or the intruder in the middle attack. So, this Diffie Hellman method is susceptible to man in the middle attack. So, this you should remember where the intruder is an active intruder, is fabricating packets and is trying to mislead B into believing that intruder is actually a fine.

(Refer Slide Time: 13:36)



A Comparison

- Symmetric encryption/decryption is much faster than asymmetric encryption/ decryption:

RSA: kilobits/second

DES: megabits/second

$\Downarrow$

DES is about ~~100~~ 1000 times faster than RSA

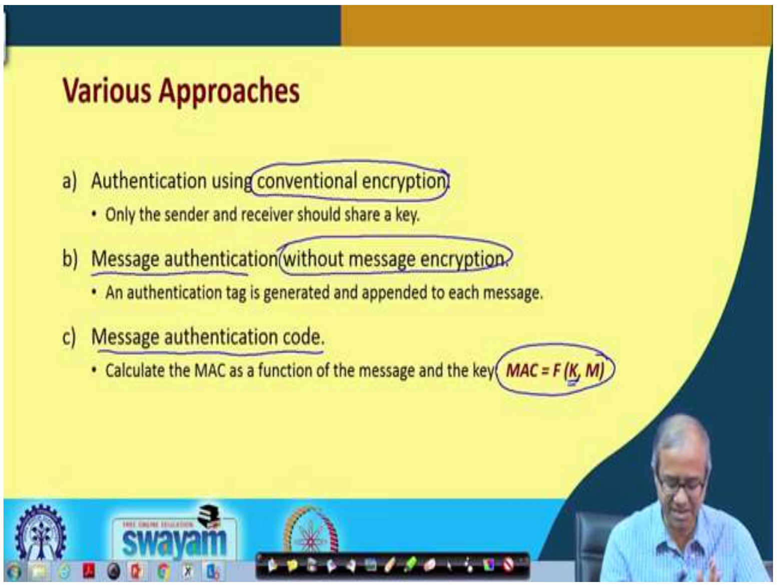
- Key size:
  - a) RSA: selected by user
  - b) DES: 56 bits
  - c) AES: 128, 192 or 256 bits

Now here I am showing a quick comparison between symmetric key and public key cryptographic algorithms. I have just, I have said very loosely that these symmetric algorithms are much faster as compared to public key. Just a simple comparison the RSA algorithm when it carries out encryption or decryption, this speed is thousands of bits per second or kilobits per second.

But DES, well AES is even faster than DES. DES is millions of bits per second. So, this is not a 100. This will be a 1000 actually. So, DES is about thousand times faster as compared to RSA. So, you can understand the differences. That is why we use a two-step process for encryption. First, we sent the secret key using RSA, then we do the actual encryption using DES or AES ok.

Talking about the key size for RSA. The key size can be selected by user. It is flexible, but for DES it is rather small. AES, it is 128, 192 or 256. These are some of the comparisons, ok.

(Refer Slide Time: 15:13)



Various Approaches

a) Authentication using conventional encryption
- Only the sender and receiver should share a key.

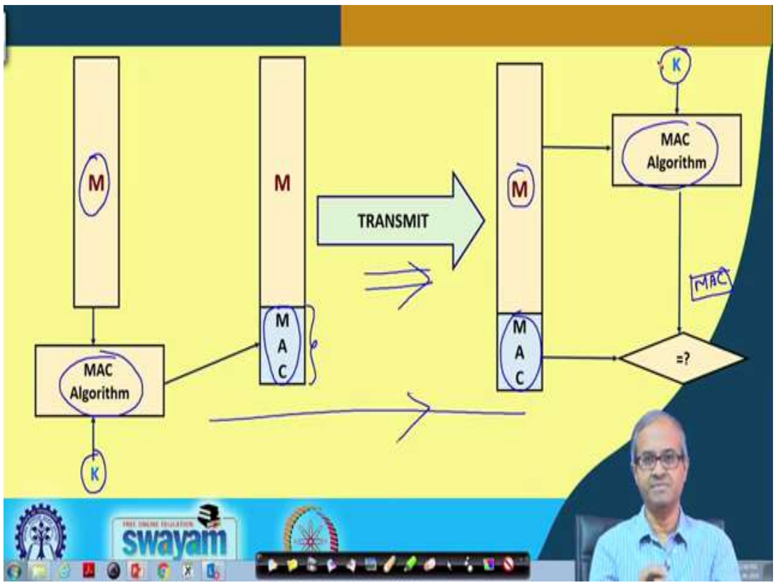
b) Message authentication without message encryption
- An authentication tag is generated and appended to each message.

c) Message authentication code.
- Calculate the MAC as a function of the message and the key $MAC = F(K, M)$

The last thing that we will be very briefly talking about here, is regarding message authentication. Well we shall be taking this up later again in some detail. Now message authentication means a sender has to authenticate itself with the receiver. The receiver should be sure about the identity of the sender. So, there can be multiple approaches here. So, you can use conventional encryption as I have already talked about using public key cryptography. You can carry out this kind of authentication or you can do message authentication without any encryption.

Here you use something called a message authentication code using something referred to as cryptographic hash function. This we shall be discussing later as I said. And the last thing you can use something called message authentication code where you use some kind of a function based on a key. There are multiple approaches there. So, I am giving you the general idea here, how it works.

(Refer Slide Time: 16:32)

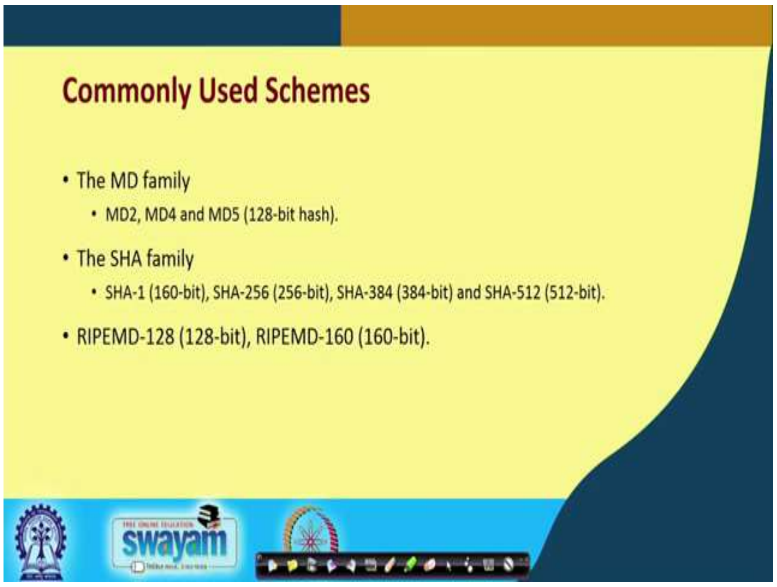


So, I am showing it with the help of a diagram. The idea is like this. Suppose the sender wants to transmit a message M. What it does is that before transmitting the message, it runs an algorithm, message authentication code which can also take a secret key as an optional argument parameter. It takes this M. It takes this k and it generates a small message authentication code MAC. Before transmitting this MAC is appended to the message and transmitted over the channel. The receiving end receives this whole thing. The receiver again runs this MAC algorithm on this message M using the same value of key. So, this will again generate a small message authentication code MAC.

So, if everything is fine, these MAC and this MAC should match. So, if they are equal then you conclude that well this message is actually coming from the sender, because this k, secret k is being shared only between these two parties. So, there are many variations to this. We shall be discussing this thing later, but the point to note is that authentication is a very important security primitive that is used to build security applications. In many cases, you need to be sure about the identity of the person who is sending you some message or some packet, data packet, ok.

So, with respect to the security of an organizational network, this authentication is something which is very important. This we shall be discussing in detail later.

(Refer Slide Time: 18:42)



Commonly Used Schemes

- The MD family
  - MD2, MD4 and MD5 (128-bit hash).
- The SHA family
  - SHA-1 (160-bit), SHA-256 (256-bit), SHA-384 (384-bit) and SHA-512 (512-bit).
- RIPEMD-128 (128-bit), RIPEMD-160 (160-bit).

And as I had said, we can use something called cryptographic hash functions and many of the cryptographic hash functions are very well known and are used in practice. There is an MD family, MD2, MD4, MD5, these are available then SHA family, sha.

Here the size of the hash function is a larger 160-bit, 256-bit, 384 and even 512-bit. The SHA family, these hash functions are considered to be better than the MD family and of course, there is RIPMED. There is another family. Many such hash functions are there. You can use them for authentication and other similar purposes. So, with this we come to the end of this lecture. Later on which we shall be looking at some more detail about this has functions and how all these primitives, cryptographic primitives can be combined together to develop some security applications which can fulfil some of the security services, which are required in certain cases. This we shall be discussing later.

Thank you.