

# **Cryptography and Network Security**

**Prof. D. Mukhopadhyay**

**Department of Computer Science and Engineering**

**Indian Institute of Technology, Kharagpur**

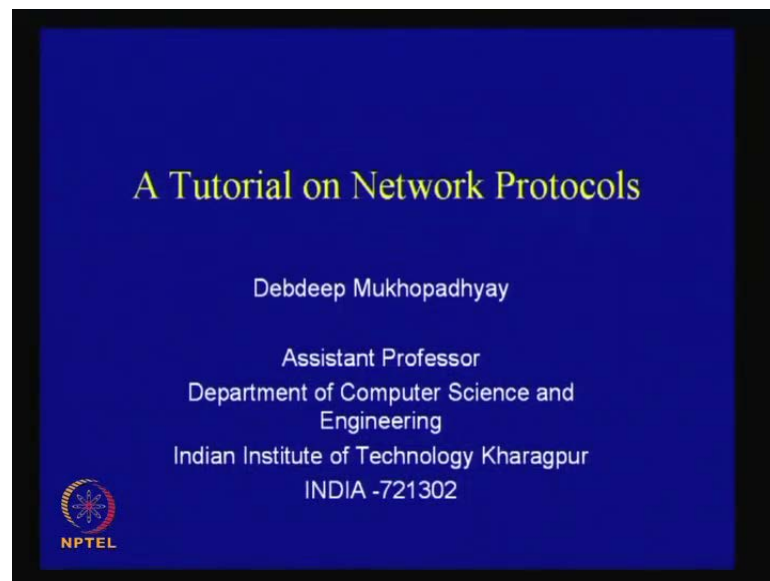
**Module No. # 01**

**Lecture No. # 38**

## **A Tutorial on Network Protocols**

Welcome to today's class. So, today we shall be discussing about network protocols. So, we essentially have been studying about various primitives in cryptography, but this is the kind of a first objective of first place, where we will try to combine or use the cryptographic primitives and try to build up a protocol for security.

(Refer Slide Time: 00:46)



So, **the protocol**, essentially there are certain objectives in a protocol. So, the first thing is to understand, what are the objectives in a protocol. So, and there are various kinds of protocols, also existing in the literature. So, presently, we shall try to understand a category or a type of protocols. So, let us try to understand what are the objectives that, we shall be considering in this class.

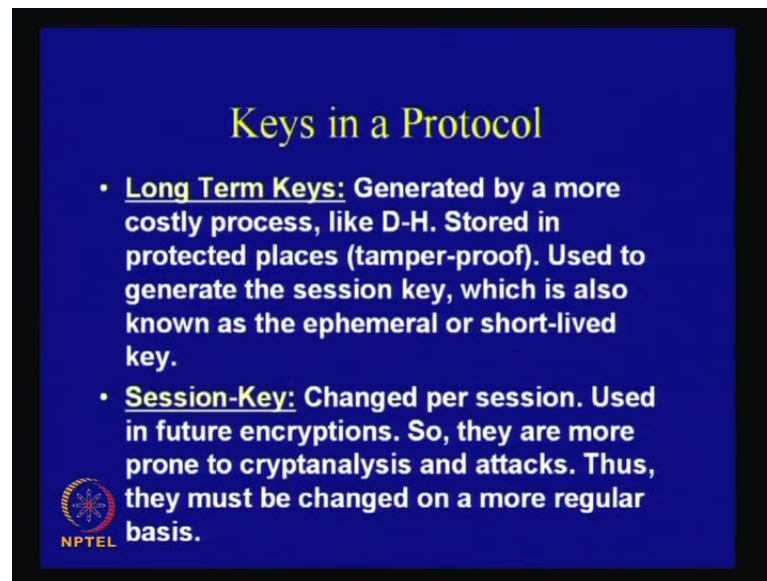
(Refer Slide Time: 01:04)



So, first of all, we will try to, essentially understand, what are the important and implicit assumptions, which are there in a cryptographic protocol or what are the requirements from a cryptographic protocol. So, we shall actually consider a 5 stage design of a three-party protocol for key establishment, where we shall see that, some of the important concerns are confidentiality, authentication and the fact that, an adversary can be an insider too and also, about the replay of old messages. So, these are some of the things which, **some of the things which** we have actually seen previously, but we shall see them again in the context of a protocol.


So, note, also another thing that, we essentially will be considering a three party protocol in this case, but there are various protocols, which can also be two parties; but the objective of this particular discussion is, to understand the basic properties, which are required there in a protocol. So, we shall be also understanding or discussing about Needham Schroeder protocol and the Kerberos protocol which are very famous protocols in this domain and we shall conclude with some discussions.

(Refer Slide Time: 02:07)



**Keys in a Protocol**

- **Long Term Keys:** Generated by a more costly process, like D-H. Stored in protected places (tamper-proof). Used to generate the session key, which is also known as the ephemeral or short-lived key.
- **Session-Key:** Changed per session. Used in future encryptions. So, they are more prone to cryptanalysis and attacks. Thus, they must be changed on a more regular basis.

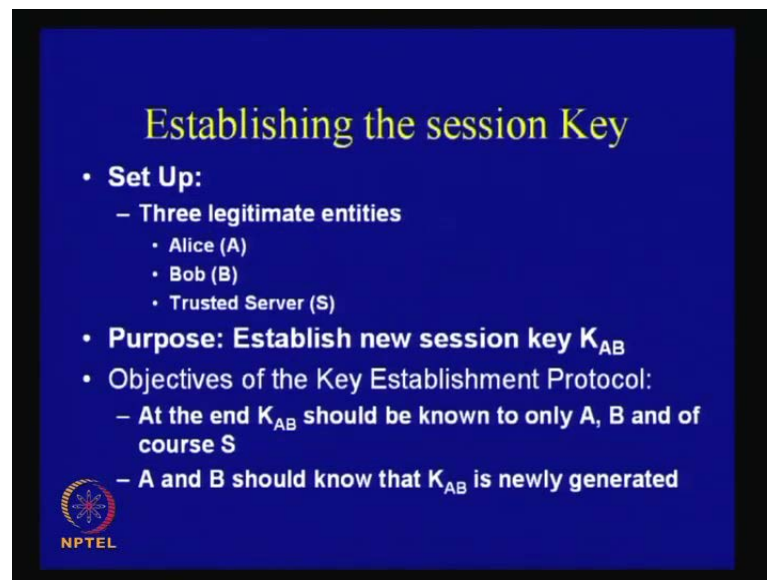
 NPTEL

So, the first thing is that, what are the keys which exist in a protocol. So, there are essentially two types of keys which exist; one which is a long term key and the other which is a session key. Now, the long term key is generally generated by a more costly process which is, could be like Diffie Hellman and it is stored in protected places and it is generally, well preserved. The idea is that, the cost of a long term key is quite high. So, therefore, it is probably determined or established by a public key exchange, which is actually, which comes with an accompanying cost, I mean, there is a cost on the network; there is a cost in terms computation. So, therefore, the protocol or the systems we will try to protect this long term keys and change them less frequently. As opposed to that, we have something which is called as a session key, which is actually used to change per session and change much more regularly.

So, the idea is that, this session keys are actually more vulnerable or more prone or more subjected to cryptanalysis; because, since, what we do essentially, we use, we generally setup or use a public key protocol to set up a long term key and then, use this long term key to generate session keys. Now, this session key is used to for bulk encryption, like for bulk data encryption like when we are actually using may be a private key or a symmetric key algorithm. And the idea is that, therefore, the observer or the attacker actually sees lot of cipher text which are generated or used by the session key. So, therefore, the session key is actually the prime target of a standard cryptanalytic approach. So, therefore, that implies that, the session key needs to be changed more


regularly. So, in our case also, we will see that, we will assume that, there are some long term keys which are being already prepared or stored in a tamper proof place and it is used to generate the session key, which is actually, sometimes also called the ephemeral or the short lived key.

(Refer Slide Time: 04:05)



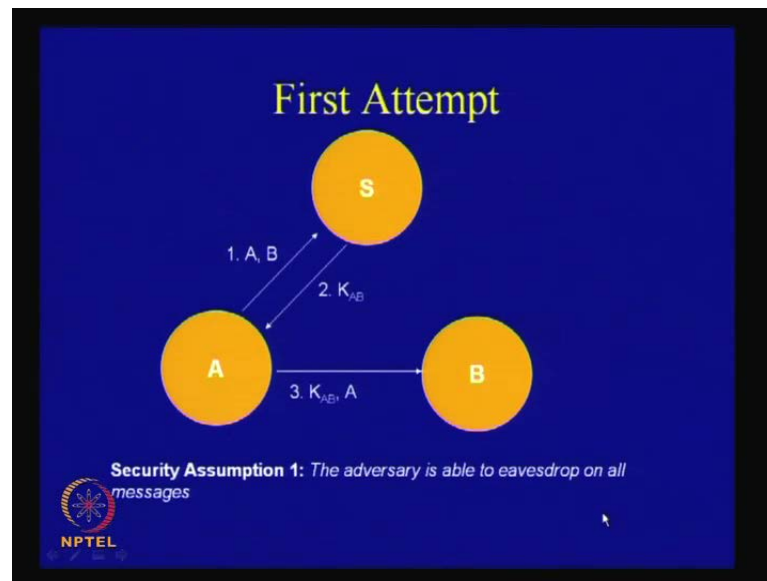
**Establishing the session Key**

- **Set Up:**
  - Three legitimate entities
    - Alice (A)
    - Bob (B)
    - Trusted Server (S)
- **Purpose: Establish new session key  $K_{AB}$**
- Objectives of the Key Establishment Protocol:
  - At the end  $K_{AB}$  should be known to only A, B and of course S
  - A and B should know that  $K_{AB}$  is newly generated

 NPTEL

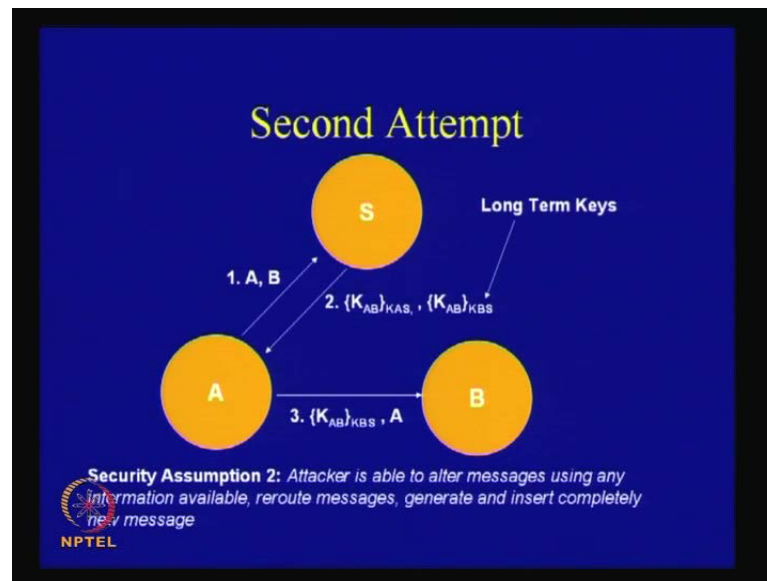
So, therefore, the context that we are considering right now, is actually, there are two parties or two legitimate entities that is Alice and Bob and there is a third entity which is the trusted server. So, the trusted server actually has got knowledge about Alice and Bob and it tries to perform a key establishment between A and B. Therefore, the objective of A and B, is to essentially establish a new session key called  $K_{AB}$ . And this  $K_{AB}$ , will be used for further communication between Alice and Bob. So, that is the basic context and the objectives of the key establishment protocol is that, at the end of this particular protocol,  $K_{AB}$  should be known to only A B and of course, S; because, S is the trusted server. And also, A and B should know that  $K_{AB}$  is newly generated; that is  $K_{AB}$  should not be an old key; it should be kind of a fresh key. So, these are the two primary objectives behind this protocol. Now, let us try to build up this protocol and try to build up this protocol in stages.

(Refer Slide Time: 05:15)



So, the first attempt could be a very simple attempt. So, it could be like, that is, this A and B which are actually the legitimate entities and S is the trusted server. So, what A could do is that, A could request S and say that, I want to generate a session key  $K_{AB}$ , which I will use for my further communications with the party B. So, in a very naive situation, S can directly send back  $K_{AB}$  and you know that, A can again send it back to B, saying that, this is the key which S has sent me and we will use it for further communication. Now, obviously, you understand that, this is a very naive protocol and therefore, it can be subjected to various kinds of attack. Now, the attack, we can, if we try to kind of write it in the form of a security assumption, we know that this protocol is weak, because, we know that, the adversary is able to eavesdrop all the messages; that is these channels are not at all trusted channels.

(Refer Slide Time: 06:29)



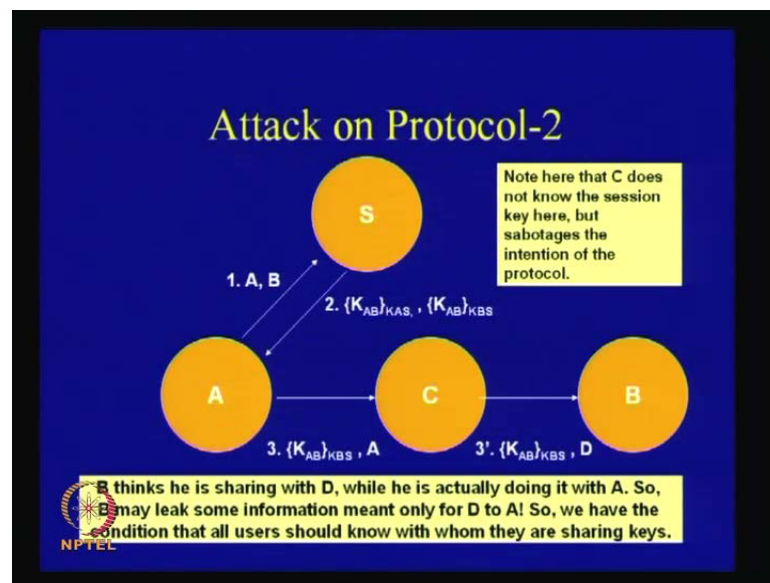
So, since these channels are not trusted channels and eavesdropper can simply observe this traffic  $K_{AB}$  and get the session key. So, of course, this is not what is intended. So, the obvious idea is to use encryption and that is the second attempt. So, what we do is that, we sent, again A sends to S that, it wants to communicate with B and, so, it sends this identifier A and B. So, note that, these identifiers could be in the form of, may be your IP address or any anything, which indicates the particular, the particular user. So, we are considering these two entities and A and B are unique identifiers in the network, for the users A and B. And, which S also is aware of. Therefore, S is a trusted server. Now, what S does is that, S takes  $K_{AB}$  or generates somehow a  $K_{AB}$  and encrypts it by the long term key.

So, as we told that, the long term keys are there. So, the long term keys have already been established by some Diffie Hellman kind of technique, which we have seen in our previous classes and  $K_{AB}$  is communicated to A and of course, this is encrypted by  $K_{AS}$  and  $K_{BS}$ , which are long term keys. So, therefore, you know that, S is aware of the fact that A and B are legitimate users. And therefore, from some data base, it retrieves the value of  $K_{AS}$  and  $K_{BS}$  and encrypts it and gives it back to A and B. So, now, A also has a long term key. So, A has a long term key called  $K_{AS}$  and therefore, it can take this traffic and it can decrypt this and obtain the value of  $K_{AB}$ . And what it does is that, the other part of the traffic, that is  $K_{AB}$ , which is encrypted by  $K_{BS}$ , it just communicates as a cipher text with, along with the identifier that is A. So, from the face

of this, we know that, the previous problem is alleviated; that is, the key is not being transferred in the, in clear and also, B is aware of the fact that, it is establishing the key with A.

So, this A is a better protocol, definitely, like the, than the previous protocol, but still it is vulnerable to attacks. The attack is possible because, of this assumption, which is quite practical in a normal network, is that, the attacker is able to alter messages using any information available; it is also able to read out messages, generate and insert, completely new message. So, which means that, the attacker is not necessarily a passive attacker, but it can be an active attacker as well. So, it is capable of altering messages; it is capable of inserting new messages and it is kind of, it is just not passively observing the network, but it can also inject some other extra information into the network.

(Refer Slide Time: 09:05)



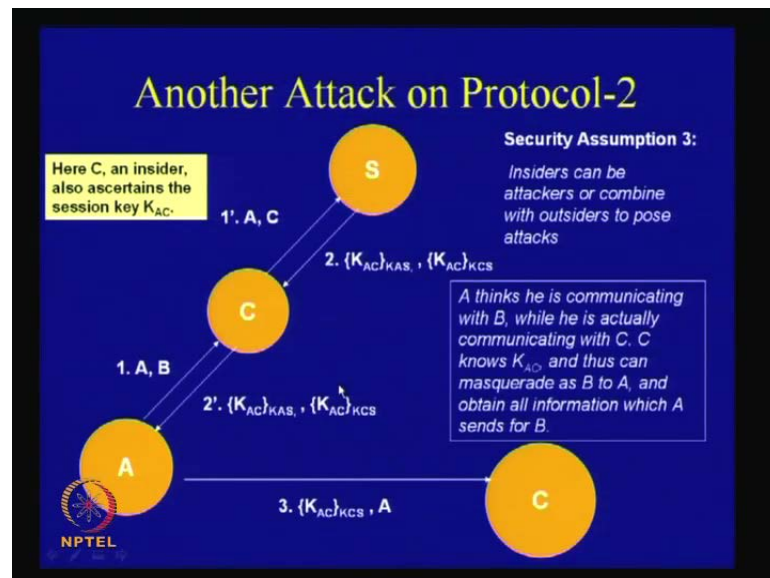
So, therefore, what happens is a possible attack like this. So, we, let us consider that, there is an adversary called C and we essentially see what C does is this; that is, when A is being, I mean, when S communicates to A, this encryption, that is  $K_{AB}$  encrypted with  $K_{AS}$  and  $K_{AB}$  encrypted with  $K_{BS}$ ,...What and therefore, we know that A communicates or rather tries to send to B,  $K_{AB}$  which is encrypted by  $K_{BS}$  and A. So, what C does is that, instead, it takes the first part, but modifies the second part and makes it something like D. So, what is the problem? The problem is, B thinks that, B takes the



first part and of course, decrypts it and thinks that, the  $K_{AB}$  key is actually meant for communicating with D.

So, therefore, you see that, A and B... So, therefore, the second... Therefore, you remember, the objective of this protocol, the second objective was that, at the end of the protocol, both end A and B should be convinced that, they are establishing a new key and should know, with whom they are communicating. But in this case, there is a problem. The problem is that, B thinks that, it is communicating with D. And therefore, what may happen is that, B may leak some information which is meant only for D to A. So, because, B, may actually end up in communicating things, which are actually meant for being only sent to D; because, it thinks that, it is using  $K_{AB}$  for communicating with D, can actually end up in leaking those information to the user A. So, therefore, you see that, these are example, where C or the adversary does not necessarily know the session key at the end of the thing, but it has definitely sabotaged the intention of the protocol. The protocol was meant to establish a key between A and B; and it was also meant to make sure that, A knows that it is communicating with B, and B knows that it is communicating with A.

(Refer Slide Time: 11:17)



So, therefore, this is a vulnerability of this protocol. The other possible vulnerability could be, this protocol, where here, C is not necessarily an external user, but it could be an insider also. So, what may happen is that, C knows or rather S knows that, C is a valid



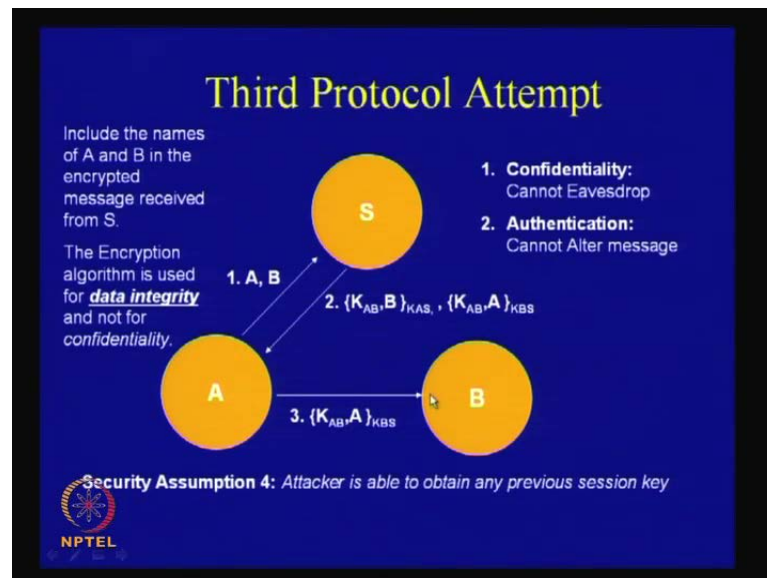
user and therefore, there is a long term key, which is meant for C also, in the data base of S. So, there could be like a long term key which is  $K_{CS}$ , which is maintained in the data base of S. So, when A sends to S, A comma B, C, it may come in between the traffic and may modify the traffic A comma B to A comma C. And what may happen is that, in this process, S will understand or think that, it is actually communicating with the, it is actually supposed to generate a key, which is meant for communication between A and C. And therefore, what it will do is that, it will generate a key or a session key, which is  $K_{AC}$  and encrypt it with  $K_{AS}$ , which is for A and also, and the other part will be, with  $K_{CS}$  which is meant for C.

And now, when this particular message comes to C, what C will do is that, it will take the second part and decrypt it with  $K_{CS}$ , because, it has got  $K_{CS}$  and obtain the value of  $K_{AC}$ . And therefore, next, what it may do is that, it may benignly or harmlessly just communicate this traffic to A. Now, A has got no way of understanding that, this is not what it intends, to deceive; because, you see that, this is also nothing, but a bit string; this is also nothing, but a bit string. So, therefore, there is no way of, that A can actually understand that, this is not what it is intended to be said. Therefore, it will take the first part and it will decrypt it by  $K_{AS}$  and will communicate back the next part, that is  $K_{AC}$  encrypted with  $K_{CS}$ , and send it to B. But note one thing that, it may happen that, if you send this part to B, then, B will decrypt it by its own key; because, B has got  $K_{BS}$ . And what may happen is that, it may get something, which is completely arbitrary.

And therefore, there may be a problem. So, what C may do is that, C may instead come and block that traffic and block it from being sent to B. So, therefore, you see that, B has got no way of understanding that, there has been a problem and at the end of the protocol, what is happening is that, A who thinks that, it is actually communicating with B, is actually communicating with C. So, C is, in this case, masquerading as B to A. So, C is actually masquerading as B to A and therefore, it is obtaining all the information which A actually sends for B. So, therefore, you see that, A thinks he is communicating with B, while A is actually communicating with C. So, C knows  $K_{AC}$  and thus, can masquerade as B to A and obtain all information, which A sends for B. So, therefore, you see that, the assumption is that, insiders can be attackers or they can combine with outsiders to pose attacks. So, therefore, the attacker need not be an external agent, but

could be an insider, who has got a long term session key established with a trusted server.

(Refer Slide Time: 14:44)



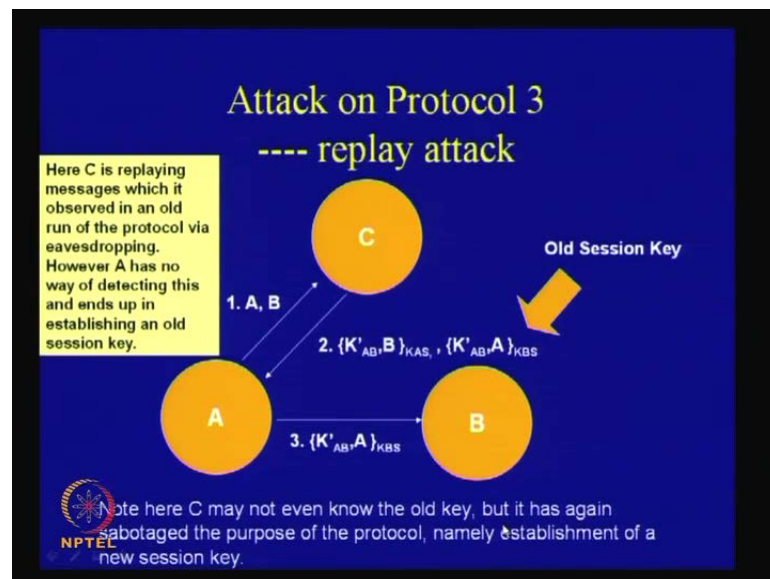
So, what is the implication of these two protocols, or rather, these two protocols attacks? The implication is that, we have to use the identifiers A and B inside the encrypted packets. So, therefore, the idea is that, the encryption algorithm, in this case, has to be used for data integrity and not for confidentiality. So, the previously, we have seen that, we have used the encryption algorithm for confidentiality; that was the first step. But in this case, we are actually using the encryption algorithm for data integrity; because, we want to convince B that, it is actually communicating with A; and I want to convince A that, it will be actually communicating with B. So, therefore, in this case, the idea is that, A sends to S again A comma B, but when S sends back a message, it takes this B and A and puts it in the plain text and encrypts it along with K AB.

So, therefore, you say that, the identifier is encrypted along with the session key, which has been generated. Now, you see that, the previous attack will not work. Because, if C sits in between and modifies it, then, it cannot generate the... **Because, when...** The moment A will decrypt it, it will immediately understand that, it is B, with whom it is communicating. So, therefore, you see that, if C actually poisons or injects a wrong message, that will be immediately be detected. And therefore, the previous attack will not hold. So, therefore, what A does is that, it just takes the second part, that is K AB and

A, and encrypts it, I mean the, which has been encrypted by  $K_{BS}$  and just relays it back to B.

So, therefore, the previous problem is quite nicely solved. So, here we see that, the confidentiality is achieved, because, since, encryption has been used and adversary cannot eavesdrop; authentication has been enforced; because, an adversary cannot actually alter the messages. But then, we again have a new security assumption. The assumption is based upon the fact that, there could be a scenario, where the adversary is aware of previous session keys. So, it could be like, an adversary was actually a member of the group at some point of time, but has left that group. So, now, when again this attack, the moment when we are considering this particular protocol, at that point, this adversary is actually outside the group, but it knows a previously generated session key. So, these are typically called as the replay attacks. So, let us again consider, this particular protocol in the light of this kind of scenario.

(Refer Slide Time: 17:22)



So, let us consider that, there is an adversary C which does this. So, what C does is that, here C is replying messages, which it has actually observed in an old run of the protocol via eavesdropping. So, it could be either eavesdropping or it could be, like it has been a part of the previous protocol. So, however, A has got no way of detecting this, and let us see, what happens. Therefore, A sends this message A comma B to C and you remember that, C was rather, A thinks that, it is actually communicating with S, the trusted server,

but instead, C comes and receives the message, and instead of allowing the message to go to S and come back to S, it actually replays up an old observed message.

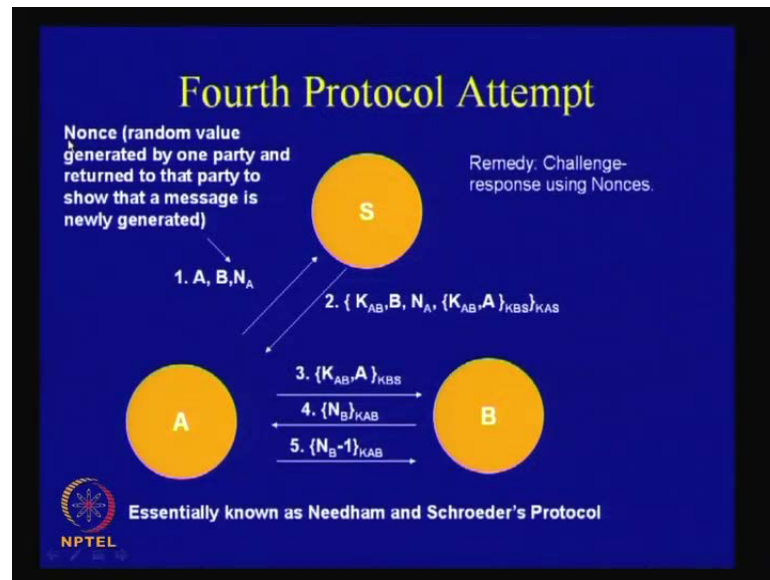
The old observed message is what? It is  $K_{AB}$ , which is an old session key, and which has been encrypted by  $K_{AS}$ , along with the identifier B, and  $K_{AB}$ , along with the identifier A, which has been encrypted by  $K_{BS}$ . So, now, you note that, this is an old traffic which has been generated. So, this message has been communicated back to A. So, what A does is that, it just takes the second part and communicates and sends it to B. Now, what is the problem? The problem here is that, C has actually knowledge of this  $K_{AB}$ , which is an old generated session key; and therefore, the session key which has been actually established between A and B, is actually already known to the adversary. So, therefore, in this case, we are not able to enforce the criteria that, we want A and B to establish a new key and not an old key. So, remember that, one of the objectives of our protocol was that, A and B should actually generate a new key and not an old key; a fresh key.

The reason is this; because, it may happen that, an old key has leaked and if the old key has leaked, that should not actually affect my communication or the security of the communication at this point. So, therefore, here the problem is that, you know that, A has got no way of detecting this and ends up in establishing an old session key. So, you also can note that, here C may not even know the old key. So, we are considering the fact that, C may not even know the old key, but then again, it has sabotaged the basic purpose of the protocol. The purpose, one of the important purpose of the protocol was that, was establishment of a new session key; the reason being very simple; that is, if you ensure that, always the old session key is generated and a new session key is not generated, then, the eavesdropper can obtain lot of cipher texts, which are being generated by the same old key; and then, it can actually try out various other cryptanalytic methods and try to actually, I mean, get the information about the key by some other means.

So, that is the basic purpose of changing the key; because, we want to ensure that, if I change the key, then, the cryptanalyst is not actually getting access to a large number of cipher texts which have being generated by the same session key. So, if in this case, an adversary ensures that an old key is being generated and new key is not allowed to generate, then, that is also, we will say as a vulnerability of the protocol or weakness in the protocol. So, in order to solve this kind of protocol attacks, the thing which is used in

some concepts like nonce or nonce, which are actually nothing, but some randomly generated fresh entities, fresh bit strings. People also use time stamps, like that, may be the time of the day. So, there are various other methods.

(Refer Slide Time: 21:10)



**But...** So, the idea is that, an nonce is being generated, which is nothing, but a random value which is generated by one party and returned to that party, to show that, a message is newly generated. So, what may happen is that, A sends to S, AB comma N A. So, the previous thing you see that, it was A comma B; now, we have added a fresh n once, which is N A. Now, what we do is that, when S sends back the message to A, it takes the key K AB, which is the session key, along with B, along with this N A, which is again sent back, so that, when A decrypts it, it gets back its N A and knows that, it is the new thing which has been generated.

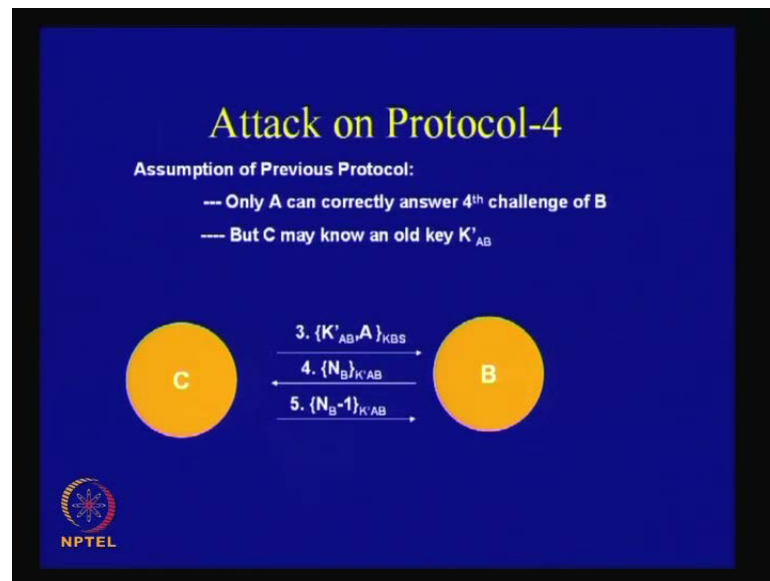
So, therefore, the message which has been received back, is actually corresponding to this message and hence it is fresh. And along with it, is this thing, that is K AB comma A, which has been encrypted by K BS and the whole thing has been encrypted by the K AS. So, often this small entity, this is, this K AB comma A, which has been encrypted by K BS is called the ticket. So, it is called the ticket, like A wants to communicate with B, but in order to do so, it actually obtains a ticket from S. So, it obtains two things, it obtains the session key along with the ticket

These are the two essential things, which it obtains from the server S. So, what it obtains is, the session key  $K_{AB}$  along with the ticket  $K_{AB} \text{ comma } A$ , which has been encrypted by  $K_{BS}$ . So, therefore, you see that, when this traffic comes to A, it can use this long term key called  $K_{AS}$  and decrypt and obtain back the value of  $K_{AB}$ . Along with it, it will also obtain the value of  $N_A$ ; be sure that, it is not a replay, and a new thing, and also this message, this entity B, which means that, it is actually, it is communicating with B itself, and there is no active adversary sitting between A and S; this is guaranteeing that fact.

So, now, what it does is that, it also obtains this ticket and it is unable to decrypt this; because,  $K_{BS}$  is not known to A and it sends back the ticket to B. Now, you note that, B knows the value of the long term key, which is  $K_{BS}$ ; so, it decrypts it, obtains  $K_{AB}$  which is the session key, along with the identifier A. Now, what it does further is that, it actually generates a fresh  $N_B$ , which is again an n once, encrypts it with  $K_{AB}$ , which is the new session key, and sends it to A. Now, what A does is that, A decrements this  $N_B$  minus and obtains  $N_B$  minus 1, again encrypts it with  $K_{AB}$  and sends it back to B.

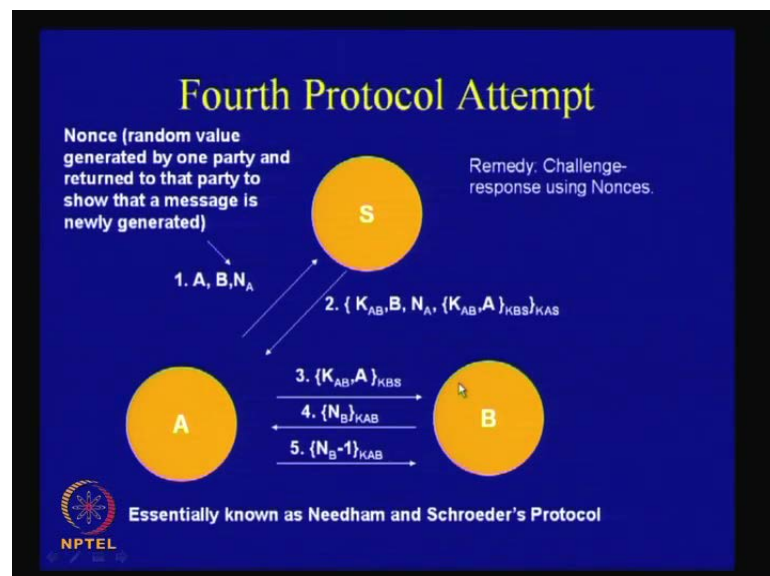
Now, you note that, this exchange ensures to B or rather gives a guarantee to B that, this is the freshly generated key. And, it also ensures to B that, A is actually knowledgeable about  $K_{AB}$ ; that is, A actually knows  $K_{AB}$ . Because, whenever it decrypts by  $K_{AB}$ , it knows that, actually the thing which A has used to encrypt, is the session key  $K_{AB}$ . So, this is the kind of key confirmation, in the context of B, that B is confirmed, that A is actually having the knowledge of the session key  $K_{AB}$ . Now, this is a very famous protocol and called as a Needham and Schroeder's protocol. So, we see that, stage by stage, we can actually derive this Needham Schroeder's protocol. Now, this protocol is actually a central, I mean rather, it is a central to what we know as the Kerberos protocol. So, that is the next thing which we will consider.

(Refer Slide Time: 24:53)



But before we go into that, let us consider yet another attack which can take place because, of the communication between A and B. The attack would be like this; that is, the previous protocol assumed that, only A can correctly answer the fourth challenge of B, but it may happen again that, C may know an old key  $K_{AB}$ .

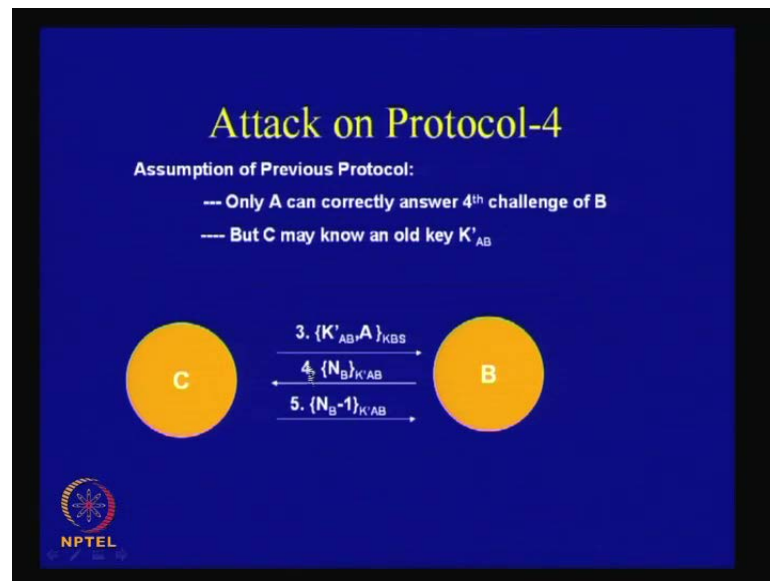
(Refer Slide Time: 25:20)



So, therefore, when B sends, I mean... So, if you consider the previous protocol, the previous protocol this message was what,  $K_{AB}$  comma  $A$  encrypted with  $K_{BS}$ .

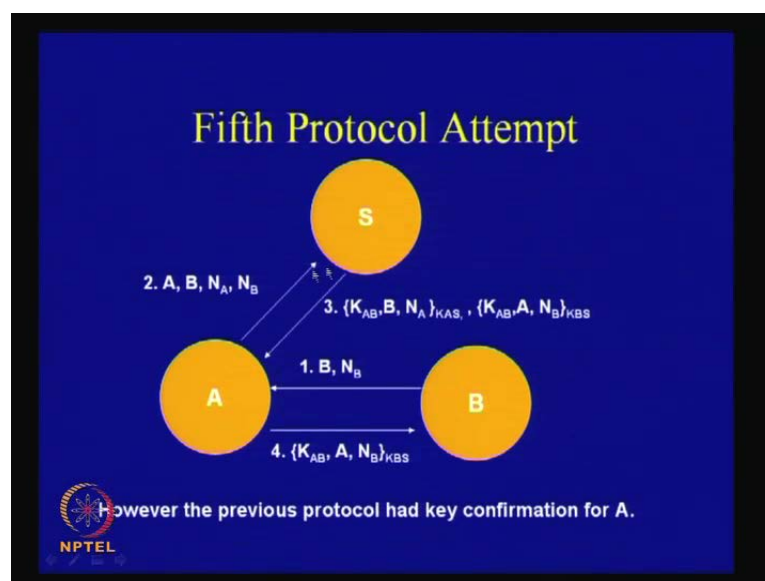


(Refer Slide Time: 25:34)



So, what may happen is that, C may come in between this communication, and may instead, relay this message, that is,  $K_{AB}$  comma A, which has been encrypted by  $K_{BS}$ . Now, you see that, when B decrypts, it actually obtains  $K_{AB}$ , which is again an old key. Therefore, when it encrypts or does the freshness, it really does not solve the problem. Because, again we see that, A and B, or rather at least B, is actually obtaining the old key, which is again violating the objective of the protocol.

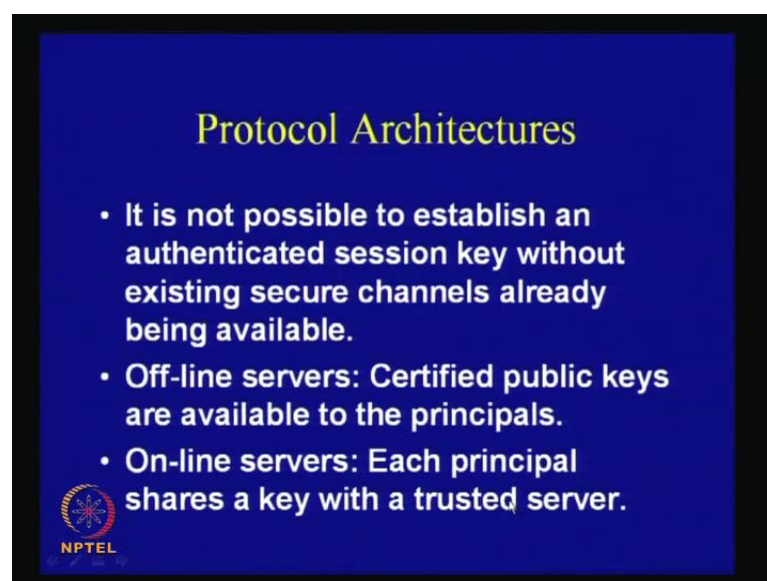
(Refer Slide Time: 26:03)



Therefore, we need to again change this protocol slightly, and the change is as follows: A communicates to S, A comma B comma N A comma N B, and S communicates to A, this message, that is K AB comma B comma N A, encrypted with K AS, and the other part is, K AB comma A comma N B comma, which has been encrypted by K BS. So, now, when A obtains this first part, it can actually decrypt it by using its long term key, obtain the session key K AB, and that is the idea. Therefore, you see that, this is obtained and when it sends back this thing, so, this message that is a second part. Then, B again uses, this is the key K BS, and decrypts and again obtains back the session key K AB.


However, the previous protocol actually tried to do something more than this protocol. Therefore, the previous protocol also tried to ensure that, there is key confirmation for A. **So, but the...** However, this protocol is actually a lesser protocol or actually obtains less, but obtains it in a secured fashion. The objective of a protocol is to actually state some objectives and achieve them. And, if you state some objectives and you are unable to achieve them, and actually open up some more vulnerabilities, then, that is not a very safe protocol to use. So, here you see that, there is a difference between the previous protocols and this one. The difference is that, the first message is actually initiated by B. So, in the previous cases, the messages were initiated by A; but in this case, this message is initiated by B and then, it has been communicated. So, that is one change or rather difference, from between the, compared to the previous protocols.

(Refer Slide Time: 27:44)



**Protocol Architectures**

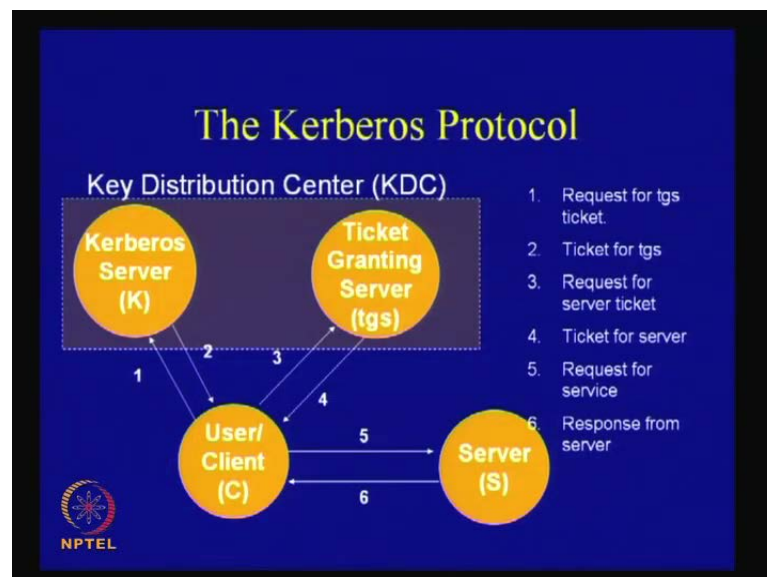
- It is not possible to establish an authenticated session key without existing secure channels already being available.
- Off-line servers: Certified public keys are available to the principals.
- On-line servers: Each principal shares a key with a trusted server.

 NPTEL

So, now, let us see, some of the discussions about protocol architectures like, it is... Therefore, this is a very, I mean, a well-known fact that, it is not possible to establish a new session key, if you do not have a priorly developed knowledge; like, what I mean to say is that, if you want to agree upon on a new information, then, there has to be some information, which you have exchanged previously.

It is a well-known fact. That is, if I want to make a new session key, you have to assume that, there is a long term key. So, that is basically, very central to the protocols that we are discussing. So, therefore, it is not possible to establish an authenticated session key without existing secured channels already being available. And in order to do this establishment, you use servers; like in the previous case, we have seen some cases of online servers, whereas, servers are actually part of the protocols. But in some other cases, there may be offline servers. The offline servers may actually give something, which is known as public keys and can distribute it along, among the network. So, it could be like a public key infrastructure or a PKI as commonly called. So, therefore, there can be offline servers, there can be online servers and depending upon that, there are various kinds of protocols.

(Refer Slide Time: 29:04)

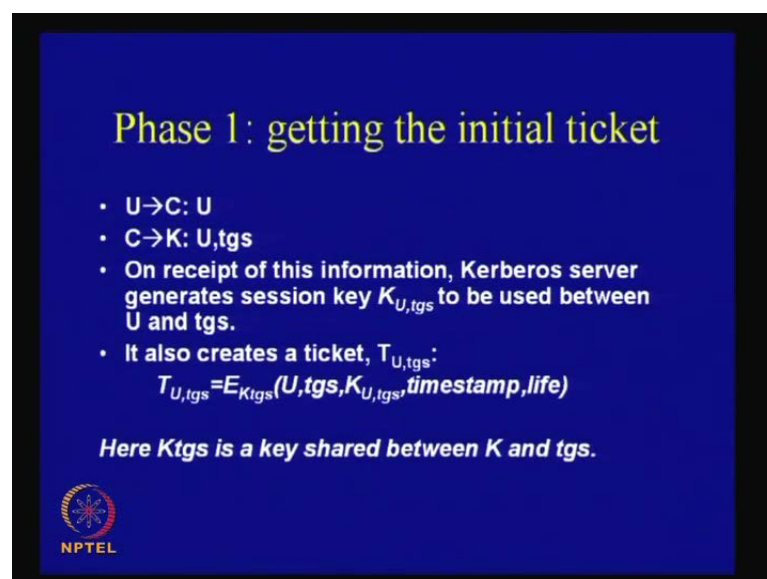


So, now, let us see, one very famous protocol, which is used in distributed networking. It is called as the Kerberos protocol. Now, you, if you have followed the Needham Schroeder's protocol, you will see that, there is a very strong application of the idea in

this protocol. So, here, there is a key distribution center and consider that, there is a user client C, who wants to actually use some application, which is there in the server S. So, in order to do so, it has to communicate with this key distribution center and obtain tickets. So, tickets are kind of, like, will enable you to access a particular service. So, what it does is that, it first communicates with the Kerberos server, so, the key distribution center has made of two servers, one is called the Kerberos server, the other is called the ticket granting server or the TGS.

So, what the user client does is that, it sends a request to this Kerberos server and asks for a ticket to communicate to the ticket granting server. So, these two actually is the ticket for TGS; for communication with the ticket granting server. Then, the client actually communicates with the ticket granting server and does a request for the actual server ticket, like the ticket for the service, which it wants from the server S. So, then, the ticket granting server gives it back, the, gives the ticket to the user client. The user client, then, actually requests for a service and obtains the corresponding response or the service. So, this is the broad idea. So, you see that, this is actually straight away an application of the Needham Schroeder's protocol; because, in the Needham Schroeder's protocols also, this was the central server and this was A and this was B, you can imagine like this. Similarly, when you are wanting to communicate with this, then, this is A, this is B and this is the central server S. So, you see the analogy, between that Needham Schroeder's protocol and the Kerberos protocol.


(Refer Slide Time: 31:08)



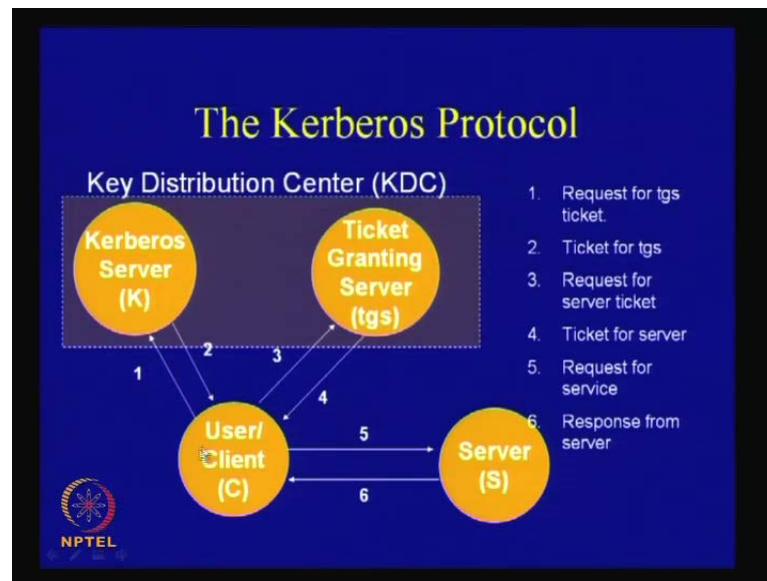
**Phase 1: getting the initial ticket**

- U → C: U
- C → K: U, tgs
- On receipt of this information, Kerberos server generates session key  $K_{U,tgs}$  to be used between U and tgs.
- It also creates a ticket,  $T_{U,tgs}$ :  
$$T_{U,tgs} = E_{K_{tgs}}(U, tgs, K_{U,tgs}, \text{timestamp}, \text{life})$$

*Here  $K_{tgs}$  is a key shared between K and tgs.*

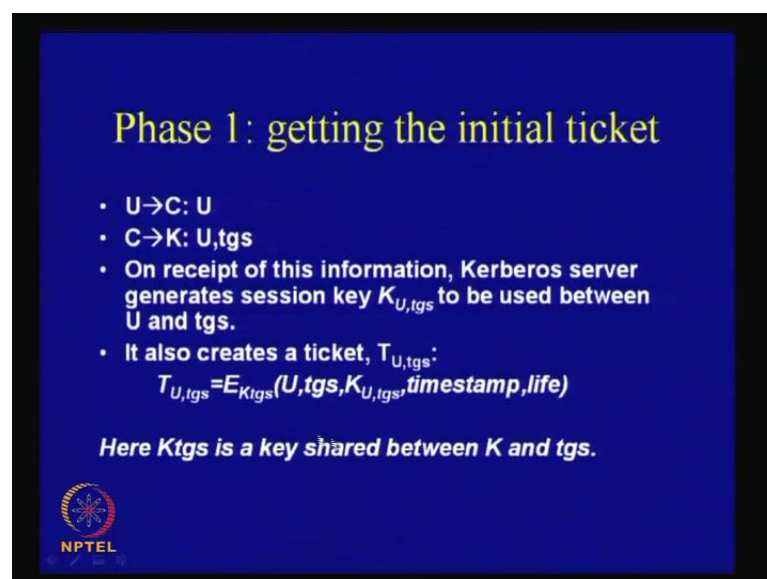


(Refer Slide Time: 31:28)



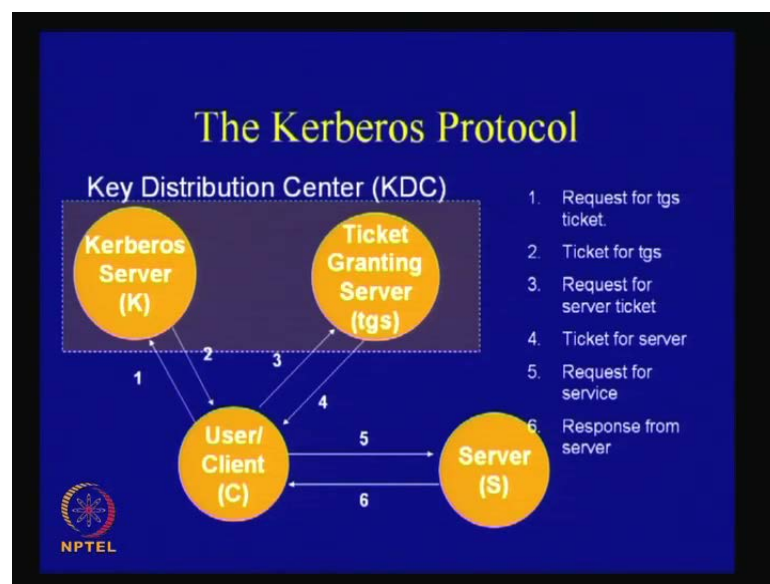
So, there are three phases in this, as you see. The first phase is like this: the user, actually communicates with the client and sends its own identifier which is you; the client then sends back to k, that is, so, you see that, what we are saying is that, we are actually dividing this user and the client into two entities. So, there is something like, you can imagine like, I am sitting over a machine, which is my, I mean, which is the, actually the client and I am the user. So, the user logs into this client and wants to use this service.

(Refer Slide Time: 31:52)



So, there is an amount of authentication between the user and the client as well. Therefore, what the user sends, that what I send to the client, is my own identifier. So, it could be in the form of a login. And what the client sends back, sends to K, that is the Kerberos server, is these two things; that is, his identifier of the user along with the identifier of the ticket granting service, where it wants to communicate with. So, on receipt of this information, the Kerberos server generates a session key, which is called as K U tgs. So, K U tgs is the session key, which is to be used by the user or the client, to communicate with the ticket granting service.

(Refer Slide Time: 32:36)



So, remember the previous Needham Schroeder's protocol. So, the previous Needham Schroeder's protocol imply that, this Kerberos server has a long term key for the ticket granting service as well as a long term key for the client.




(Refer Slide Time: 32:50)

**Phase 1: getting the initial ticket**

- **U → C: U**
- **C → K: U, tgs**
- On receipt of this information, Kerberos server generates session key  $K_{U,tgs}$  to be used between U and tgs.
- It also creates a ticket,  $T_{U,tgs}$ :  
$$T_{U,tgs} = E_{K_{tgs}}(U, tgs, K_{U,tgs}, \text{timestamp}, \text{life})$$

*Here  $K_{tgs}$  is a key shared between K and tgs.*




So, therefore, these are the long term keys. So, the long term keys are, I mean... So, there are two long term keys. So, you understand that, that, there is one key which is the  $K_{tgs}$ ;  $K_{tgs}$  means, it is the long term key of the Kerberos server, for the ticket granting service; and there is another long term key, which is the  $K_U$ , which is which is maintained by the Kerberos server for the user U.

(Refer Slide Time: 33:10)

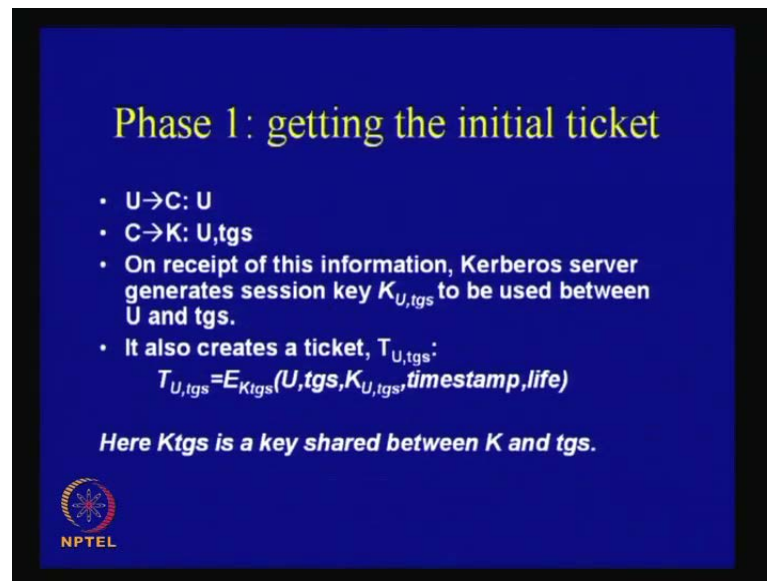
**Phase 1: getting the initial ticket**

- **K → C:**  
$$E_{K_U}(T_{U,tgs}, K_{U,tgs}, tgs, \text{timestamp}, \text{life})$$
- The client uses its key (generated from a password by a one-way function) and decrypts the message.
  - *If the authentication is successful, the client stores the session key, the ticket and other information for future usage.*






(Refer Slide Time: 33:20)



**Phase 1: getting the initial ticket**

- $U \rightarrow C: U$
- $C \rightarrow K: U, tgs$
- On receipt of this information, Kerberos server generates session key  $K_{U,tgs}$  to be used between  $U$  and  $tgs$ .
- It also creates a ticket,  $T_{U,tgs}$ :  
$$T_{U,tgs} = E_{K_{tgs}}(U, tgs, K_{U,tgs}, \text{timestamp}, \text{life})$$

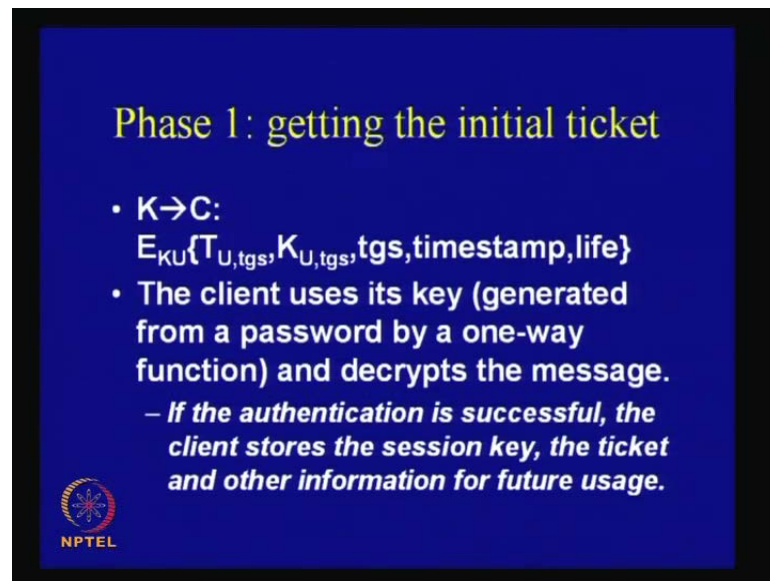
*Here  $K_{tgs}$  is a key shared between  $K$  and  $tgs$ .*



So, what it does is this. That is, first, on receipt of this information, the Kerberos server generates session key called  $K_{U,tgs}$  which is to be used between the user and the  $tgs$ . So,  $K_{U,tgs}$  is what, the session key. And it also creates a ticket  $T_{U,tgs}$ ; so, that means, that is for user  $U$  to communicate with the ticket granting service and the ticket actually comprises of the identifier of the user  $U$ , the  $tgs$ , which is the server of the, identifier of the ticket granting service or the server and also, the session key which is  $K_{U,tgs}$  along with the time stamp and the life of this ticket.


So, there is a kind of life or expiry date of this ticket. And this entire ticket has been encrypted by  $K_{tgs}$ , which is a long term key, which is maintained by the server  $K$ , that is, Kerberos server, for the ticket granting service. So,  $K_{tgs}$  is the key which is shared between  $K$  and  $tgs$ . So, it is again a long term key;  $tgs$  has previously knowledge of  $K_{tgs}$ . The ticket granting service is already aware of this  $K_{tgs}$ . Therefore, immediately, when the ticket granting service receives this packet, it can actually decrypt this by  $K_{tgs}$  and obtain the value of this session key  $K_{U,tgs}$ .

(Refer Slide Time: 34:50)



**Phase 1: getting the initial ticket**

- $K \rightarrow C$ :  
 $E_{K_U}\{T_{U,tgs}, K_{U,tgs}, tgs, \text{timestamp}, \text{life}\}$
- The client uses its key (generated from a password by a one-way function) and decrypts the message.
  - *If the authentication is successful, the client stores the session key, the ticket and other information for future usage.*



And, since this, and when this Kerberos server sends to the client, it sends this  $K_U$  tgs along with this ticket  $T_{U,tgs}$ , along with this identifier of the ticket granting service and the time stamp and the life; it encrypts this packet by this long term key  $K_U$ . Therefore, again, the client or the user is aware of this key. And therefore, it will actually decrypt this and obtain the value of the session key. And the idea is that, the client uses its key, which is generated from a password, which could be by a one way function and decrypts its corresponding message. And, if this authentication is successful, then, the client will store the session key, the ticket and other information, for future use. So, it will store this ticket as well as this  $K_U$  comma tgs for further use.


(Refer Slide Time: 35:42)

### Phase 2: Getting Server Tickets

- $C \rightarrow tgs: S, T_{U,tgs}, A_U$
- $A_U$  (Authenticator)  
 $= E_{K_{U,tgs}}(C, \text{timestamp})$

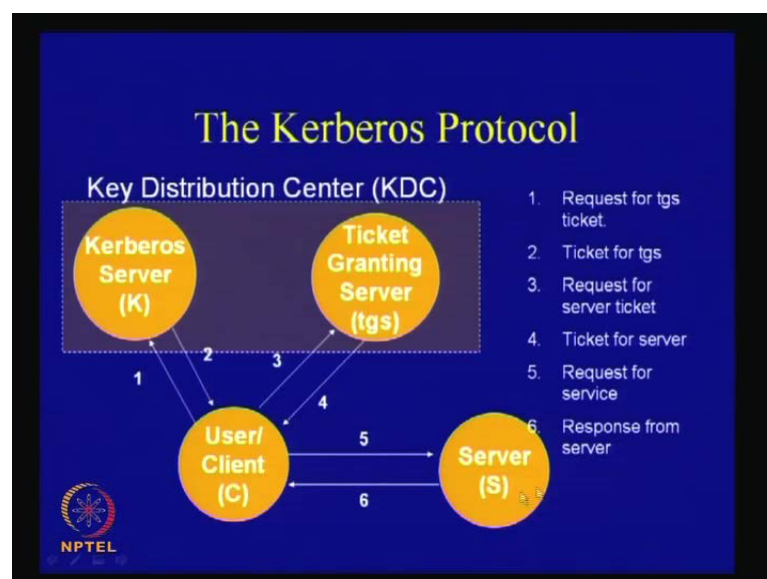
On receipt of this message, tgs decrypts  $A_U$  with  $K_{U,tgs}$  (from  $T_{U,tgs}$ ) and checks if the message is fresh (protection from replay attacks).

It creates a new session key  $K_{C,S}$  and a corresponding ticket,  $T_{C,S}$

$$T_{C,S} = E_{K_S}(C, S, K_{C,S}, \text{timestamp}, \text{life})$$


So, therefore, what the client does now, is that, it has to communicate with what, with the ticket granting service; because, it has to convey the, convey what, to convey the session key to the ticket granting service. So, what it does is that, it sends the  $T_{U,tgs}$  like this; because, you note that, this  $T_{U,tgs}$  is already encrypted. So, you do not need to encrypt it and also, the identifier of the server. Now, you see that, it is sending the identifier of the, of  $S$ , because,  $S$  is now the service which it wants.

(Refer Slide Time: 36:14)



S is this one, S is this service identifier. Now, when it sends this information to the tgs, it also sends that, I want to actually communicate with the server and I want the corresponding ticket for communicating with the server.

(Refer Slide Time: 36:30)


**Phase 2: Getting Server Tickets**

- C→tgs: S,  $T_{U,tgs}, A_U$
- $A_U$  (Authenticator)  
 $= E_{K_{U,tgs}}(C, \text{timestamp})$

On receipt of this message, tgs decrypts  $A_U$  with  $K_{U,tgs}$  (from  $T_{U,tgs}$ ) and checks if the message is fresh (protection from replay attacks).

It creates a new session key  $K_{C,S}$  and a corresponding ticket,  $T_{C,S}$

$$T_{C,S} = E_{K_S}(C, S, K_{C,S}, \text{timestamp}, \text{life})$$

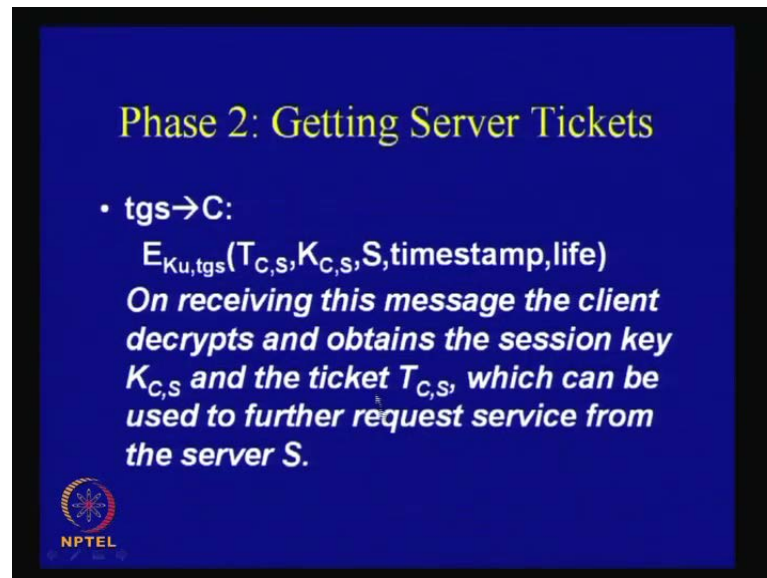
 NPTEL

So, now, what it does is that, along with it, it also gives or generates an authenticator. The authenticator is very simple; it actually uses the session key called  $K_{U,tgs}$ , the session key means, the session key between the user and the ticket granting service and generates a time stamp along with the, its own identifier, that is the clients' identifier, and sends it along with this message, to the ticket granting service. Now, you see that, from this ticket granting service, actually obtains a value of  $K_{U,tgs}$ , which is the session key. And, it can immediately decrypt this authenticator; because, this authenticator has been encrypted using  $K_{U,tgs}$  and obtains back this value of time stamp, obtains and gets the time stamp, which has been used in this message.

And therefore, when, in the next message, when the ticket granting service communicates back with the client, it actually uses this time stamp to indicate again, that it is fresh message and not a case of replay which has been sent. And therefore, you see that, the corresponding ticket... Now, the ticket granting service is actually sending a message back to the user and sending what, sending the ticket for communicating with the server along with the session key, which the user will actually use to communicate with the server. So, there are two things we are needed to be communicated; one is the


session key and the other thing is the ticket. And therefore, when it sends it, it will send this corresponding  $K_C$  comma  $S$  and also the time and along with the time stamp, to indicate that, it is not a case of the replay attack and also the life of the ticket.

(Refer Slide Time: 38:217)



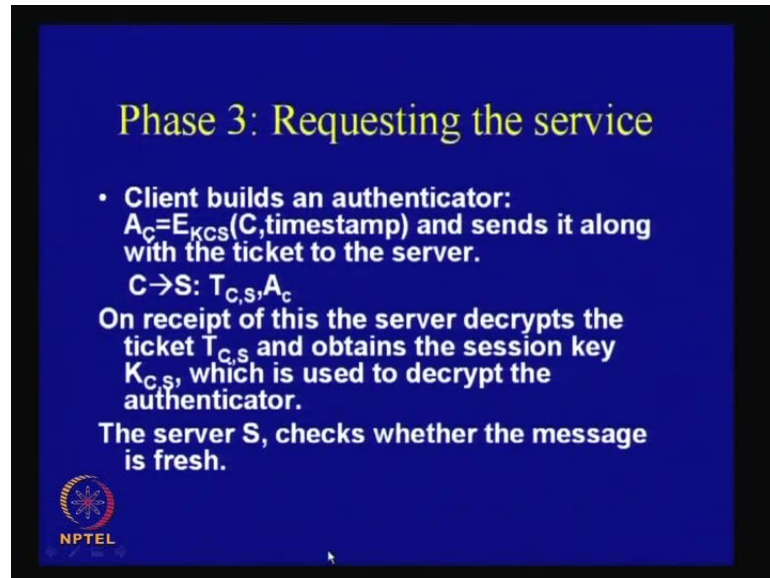
**Phase 2: Getting Server Tickets**

- $tgs \rightarrow C$ :  
 $E_{K_u, tgs}(T_{C,S}, K_{C,S}, S, \text{timestamp}, \text{life})$   
*On receiving this message the client decrypts and obtains the session key  $K_{C,S}$  and the ticket  $T_{C,S}$ , which can be used to further request service from the server  $S$ .*

 NPTEL

So, this is actually sent along with  $K_C$  comma  $S$ . What is  $K_C$  comma  $S$ ? It is the session key between the client and the server. So, there are two important things, along with the  $S$ , which is the identifier of the server along with the time stamp and the life. So, you see that, everything has got its importance; the ticket has got its importance; every place, where you are using the encryption also, has got a meaning. So, on receiving this message, the client will decrypt and obtain the session key  $K_C$  comma  $S$  and the corresponding ticket, which is  $T_C$  comma  $S$  which can be used for further request service from the server. So, immediately, when  $C$  receives this, it has actually obtained the corresponding session key and also the ticket.

(Refer Slide Time: 39:02)




**Phase 3: Requesting the service**

- **Client builds an authenticator:**  
 $A_C = E_{K_{CS}}(C, \text{timestamp})$  and sends it along with the ticket to the server.  
 $C \rightarrow S: T_{C,S}, A_C$

On receipt of this the server decrypts the ticket  $T_{C,S}$  and obtains the session key  $K_{C,S}$ , which is used to decrypt the authenticator.

The server  $S$ , checks whether the message is fresh.



So, now what the client will do is that, it will communicate with the server  $S$  and it will send simply the ticket  $T_{C,S}$ , along with an authenticator, which is again, similarly generated like the previous authenticator; only this time, you are encrypting by the corresponding session key, which is  $K_{C,S}$  and encrypting the corresponding time stamp. So, the idea is that, when the server receives it, it obtains the corresponding session key and if it wants, you can again send back the authenticator, I mean, corresponding time stamp, so that, it is again clear that, there is no case of replay between the client and the server. So, therefore, on receipt of this, the server decrypts the ticket  $T_{C,S}$  and obtains the session key  $K_{C,S}$ , which is used to decrypt the authenticator; the server  $S$  checks whether the message is fresh or not.


(Refer Slide Time: 40:01)

**Phase 2: Getting Server Tickets**

- $C \rightarrow tgs: S, T_{U,tgs}, A_U$
- $A_U$  (Authenticator)  
 $= E_{K_{U,tgs}}(C, \text{timestamp})$

On receipt of this message, tgs decrypts  $A_U$  with  $K_{U,tgs}$  (from  $T_{U,tgs}$ ) and checks if the message is fresh (protection from replay attacks).

It creates a new session key  $K_{C,S}$  and a corresponding ticket,  $T_{C,S}$


$$T_{C,S} = E_{K_S}(C, S, K_{C,S}, \text{timestamp}, \text{life})$$


(Refer Slide Time: 40:06)

**Phase 2: Getting Server Tickets**

- $tgs \rightarrow C:$   
 $E_{K_{U,tgs}}(T_{C,S}, K_{C,S}, S, \text{timestamp}, \text{life})$

On receiving this message the client decrypts and obtains the session key  $K_{C,S}$  and the ticket  $T_{C,S}$ , which can be used to further request service from the server  $S$ .




Because, the server  $S$  can check whether the message is fresh; because, you see that, how the ticket has been generated; because, the ticket also has a time stamp and the message that you are saying as a authenticator also, has a time stamp.



(Refer Slide Time: 40:12)

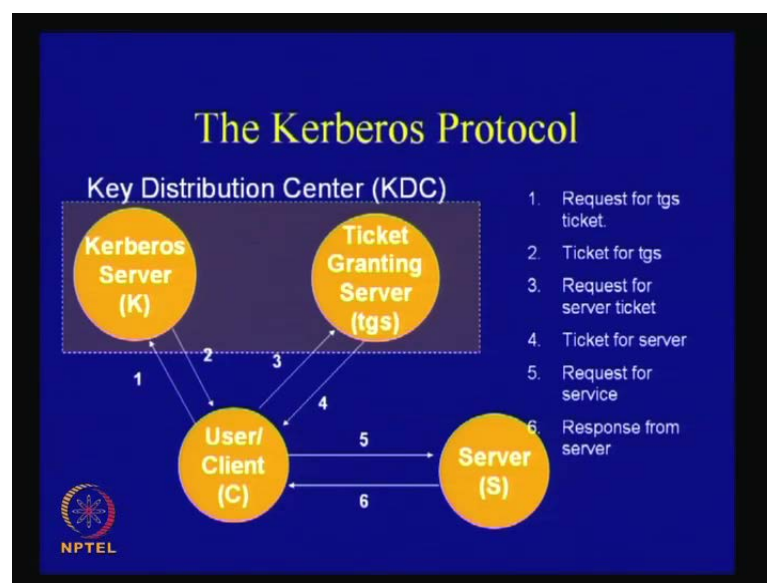
### Phase 3: Requesting the service

- Client builds an authenticator:  
 $A_C = E_{K_{CS}}(C, \text{timestamp})$  and sends it along with the ticket to the server.  
 $C \rightarrow S: T_{C,S}, A_C$
- On receipt of this the server decrypts the ticket  $T_{C,S}$  and obtains the session key  $K_{C,S}$ , which is used to decrypt the authenticator.
- The server S, checks whether the message is fresh.



So, therefore, you can immediately understand that, whether, the authenticator also has a time stamp. So, you understand that, the server S will check whether the message is fresh or not; because, it can compare these two things and see the life of the ticket. So, the timestamp and the life are used to check that, it is indeed a fresh message and actually, prevent replay kind of ((annoyances)).

(Refer Slide Time: 40:47)

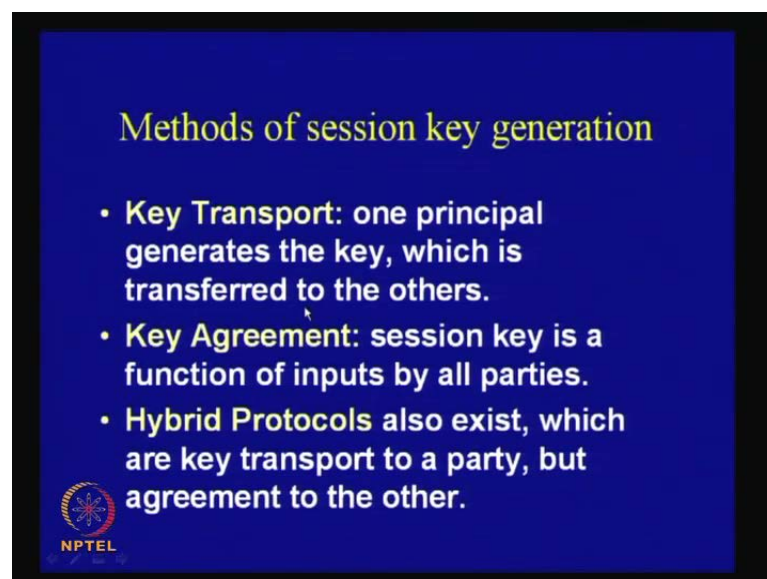


So, this is the broad idea behind this kind of protocol attacks. So, therefore, you see that, the previous three things, that is the three phases, like the phase one, where you are

actually obtaining a ticket for communicating between the client and the server; and you are obtaining a ticket for communicating between the client and the server, I mean, the first one was client and the ticket granting server, second one is a client and the actual server, is actually based on the Needham and Schroeder's protocol.


And essentially, the objective of this protocol is to actually obtain the three important criteria. The three important criteria are essentially that, an eavesdropper will not be able to observe; the second one is that, there should not be any insider, who can actually do an attack; even if the old key is being retrieved and I mean, even if the old key is leaked, it should not be again used; I mean, it should not be like, the old key is again and again forced to be generated by the parties involved. And the other thing is that, you are also ensuring that, there is a kind of freshness in the keys that you are generating. That is... So, these ideas are kind of related, but these are the important objectives, behind the protocol of this nature. So, now, you can have various varieties of protocols; you can have protocols which are two parties, three parties, multi parties, but the idea is that, the objectives or the basic objectives, are essentially the same.

(Refer Slide Time: 42:16)



**Methods of session key generation**

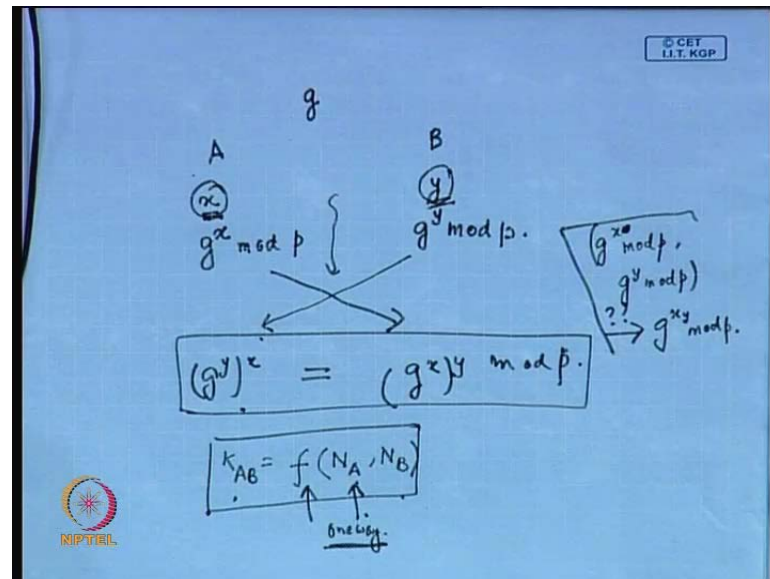
- **Key Transport:** one principal generates the key, which is transferred to the others.
- **Key Agreement:** session key is a function of inputs by all parties.
- **Hybrid Protocols** also exist, which are key transport to a party, but agreement to the other.

 NPTEL

So, this Kerberos protocol was actually a part of a project which was generated by MIT. So, now, there can be various modes of session key generation. There could be like, things like, key transport where one principle generates the key, which is transferred to the other. So, therefore, as we have seen, that the previous cases where examples of key

transport; because, the session key has been generated by the server and it has been actually passed to both A and B in some form. There could be some others protocols, where the session key is actually a function of inputs generated by all parties; like, if there are two parties A and B, and both actually gives some kind of a share, then, that is used to generate a key.

(Refer Slide Time: 43:04)



So, a classic example of that is something, as we have seen, is the example of the famous Diffie Hellman protocol, where you know that, there is generator which has been previously established and it could be like  $x$ . So, generator means generator of a finite field and or a finite group and  $x$  is a member of this, which is generated randomly and B also generates  $y$ , which is the random share. So, what A does is that, A computes this value of  $g$  power of  $x$ . This could be modulo given prime number. So, and the other thing which can be generated by B is,  $g$  power of  $y$  mod  $p$  and what may happen is that, this is communicated to B and again, this is communicated to A. And now, when  $g$  power of  $y$  mod  $p$  comes to A, it actually takes  $g$  power of  $y$  and raises it to this power of  $x$  and what A does is that, it takes  $g$  power of  $x$  and raises to power of  $y$ .

And in both cases, we know that, these are the same. Therefore, if you take modulo  $p$ , then, the both of these are exactly the same. So, you see that, here also, we have actually generated a random number  $x$  and a random input  $y$  and you are actually establishing a key, which is  $g$  power of  $x$   $y$  between these two parties. So, this is the classic example of

a key agreement, where key agreement between the parties A and B, where the session key is actually a function of the shares, which has been generated by A and B. So, typically, we can say that, there is a function called  $f$  and there A and B has actually contributed and generated like  $N_B$  and  $N_A$  and the session key  $K_{AB}$  is actually a function of both  $N_A$  and  $N_B$ . So, therefore, this function has to ensure that, it is neither skewed towards  $N_A$  or neither skewed towards  $N_B$ ; both of them should have an equal contribution to a value of  $K_{AB}$ ; because, what we will consider in the case of cryptanalysis is that, I have a knowledge of  $N_A$  along with some public information, can I obtain this value of  $K_{AB}$ ?

So, in this case for example, you imagine, like, if you actually know the value of  $x$  and you know the value of  $g$  power of  $y$ , then, you know that, immediately, you can calculate  $g$  power of  $xy$ . But as a cryptanalytic point of view, you would be interested, like, because, if there is any eavesdropper who observes this, then, the eavesdropper obtains what,  $g$  power of  $x \bmod p$  and  $g$  power of  $y \bmod p$  and it is interested to find out, some way of computing the value of  $g$  power of  $xy \bmod p$ . So, this, we have seen is believed to be a difficult problem, which is known as a famous Diffie Hellman problem. So, therefore, the idea is that, the function  $f$  should not be skewed towards either  $N_A$  or  $N_B$  and is typically a one way function; because, it is also important, that a knowledge of  $K_{AB}$  does not actually reveal the value of  $N_A$  and  $N_B$ . So, therefore, we would like, ideally, at the end of the protocol also, A and B should not be aware, like A should not be aware or get A knowledge of  $y$  and B should not get a value of  $x$ .

So, therefore, the typical requirement of a protocol or rather, a function would be that, it has to be essentially one way by nature. So, there are some other protocols which are also hybrid. So, hybrid means, they are partially, it is actually a key transport protocol and partially, it is an actual agreement protocol. So, therefore, it could be like partially, the protocol is a hybrid, I mean, it is a actually a key transport, that is, if there are two members A and B, then, to A, it is a key transport, but may be to B, it is actually an agreement.

(Refer Slide Time: 47:06)



So, there could be various kinds of protocols as well. So, let us see one example. We will see one example of a hybrid protocol. You can also categorize your protocols based upon the number of users. So, you can have, as I told you that, two party protocols or multi party or which is also called as conference key protocols. So, therefore, if you are going for conference key protocols, it will only complicate the matter a great deal and there will be more other criteria and requirements, we will come up; but the basic criteria that which we have seen already, will still remain. Therefore, it will only make it, like the protocols are to be more complex and has to consider more practical scenarios and problems.

(Refer Slide Time: 47:39)

**Hybrid Protocol**

- $A \rightarrow B: A, N_A$
- $B \rightarrow S: \{N_B, A, B\}_{K_{BS}}, N_A$
- $S \rightarrow A: \{K_{AB}, A, B, N_A\}_{K_{AS}}, N_S$
- $A \rightarrow B: N_S, \{A, B\}_{K_{AB}}$
- $B \rightarrow A: \{B, A\}_{K_{AB}}$

Observe that B is not being given  $K_{AB}$  explicitly. He can compute using a function  $f, K_{AB} = f(N_B, N_S)$ .

To B this is an example of agreement, while for A it is a key transport.

So, now let us see, one example of a hybrid protocol. So, here there are again three parties, A B and S which is involved and you see that, A actually communicates with B and sends A comma N A. So, therefore, A will communicate to B and send the value of A and also a fresh input, which is generated and called as N A. Now, what B sends back to S is N B comma A comma B comma K BS. So, you see that, this is quite, as I mean, as we have seen in the previous case, the protocol, as we have seen in the previous context, it is just one packet like that.

And, so, that is the fifth protocol, similar to that fifth protocol. So, you see that, this is being generated and B sends this to the server S. So, B actually sends it back to S and what... The other thing which it also sends is the corresponding value of N A. So, N A is also the other thing which has been obtained from A, I mean, B has received the corresponding entity from A and it has just relayed it back to S. So, what S does is that, S sends to A, a key which is called as K AB, which is a newly generated session key, K AB, along with A, along with B, along with this fresh N A and it has encrypted the entire thing by a long term key called as K AS.

And you note that, the other thing which has been sent is this value of NS. So, the NS is a fresh number which is being generated by the server. Now, what A does or A sends to B, is this NS; along with A and B, which has now been encrypted by K AB. So, you note one thing that, in this case, the A has actually obtained back, obtained the key; it has just

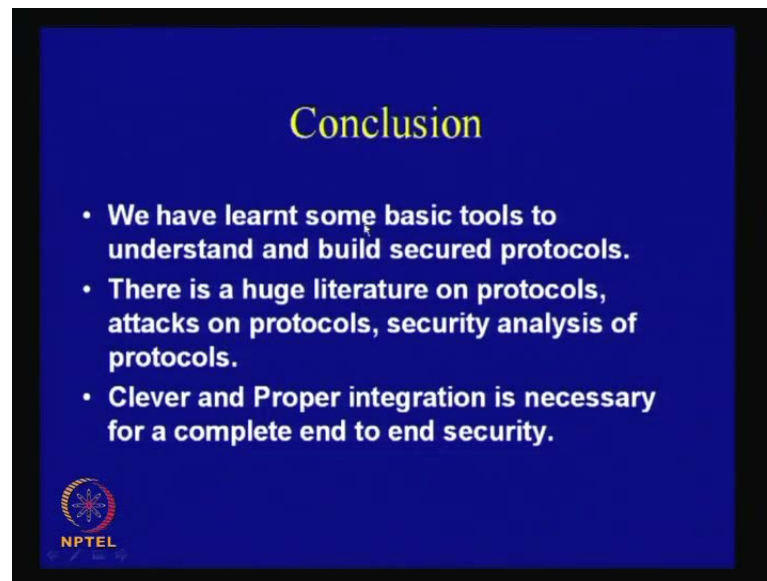
got back the key from the server; because, it has been just transported to A. The key has been just transported to A and it has just got it; it can simply obtain it by decrypting this packet, by using its long term key called  $K_{AS}$ .

Now, what B does is that, B sends again back to A  $K_{AB}$ , I mean,  $B \text{ comma } A$ , which has been encrypted by  $K_{AB}$ . But if you observe, one interesting point here is that, so far as B is concerned, B has actually not been provided  $K_{AB}$  explicitly; because, B has never actually obtained  $K_{AB}$  explicitly. You see that, B... What are the messages B has actually got? B has a knowledge of  $N_A$ , and it has also got the knowledge of  $N_B$ , of course, and it has also got the knowledge of  $N_S$ , because, that is the thing which the server has communicated to B through A. Now, how has the server generated this key  $K_{AB}$ ? That is important. Because, that  $K_{AB}$  has been generated by the server and communicated to A.

Now, what the server does is that, using its key or rather, part  $N_S$  and the part  $N_B$ , it actually applies a one way kind of function and obtain session key called as  $K_{AB}$ . And this  $K_{AB}$  is actually again, can be generated by B; because, B also has got a knowledge of  $N_S$  and it has got a knowledge of  $N_B$ . So, B also has got a knowledge of  $N_B$ , of course, because, it is his own key. Therefore, now, it can again apply this function  $f$  and it can obtain the value of  $K_{AB}$ . So, therefore, so far as B is concerned, B is actually not been given  $K_{AB}$  explicitly, but it has been used to compute a function  $f$ , by using, it has been used by or rather, generated by computing a function  $f$ . So, therefore, to B, it is an example of agreement, while for A, it is actually an example of key transport. So, this is a typical example of a hybrid kind of protocol, where, to one party, it is a transport and to the other party, it is an example of a agreement.




(Refer Slide Time: 51:52)



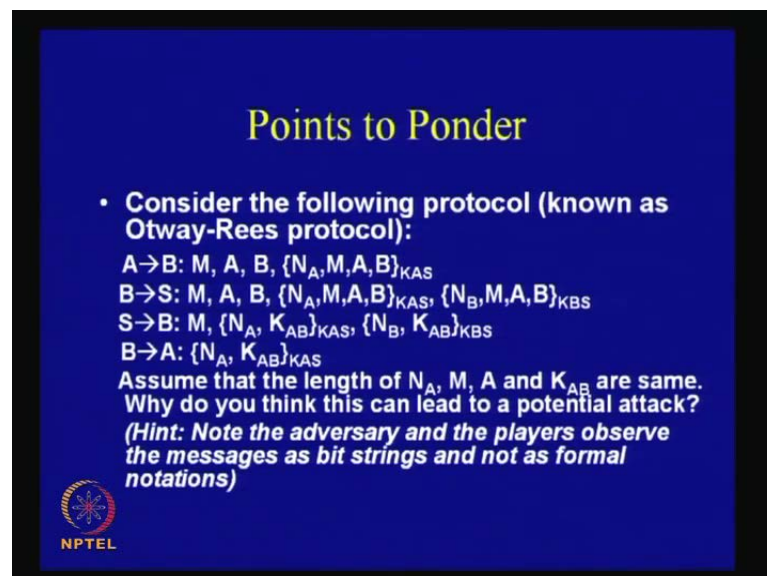
### Conclusion

- We have learnt some basic tools to understand and build secured protocols.
- There is a huge literature on protocols, attacks on protocols, security analysis of protocols.
- Clever and Proper integration is necessary for a complete end to end security.




So, the conclusion is that, we have learnt some basic tools to understand and build secured protocols; that is a huge literature on protocols, attacks on protocol, security analysis of protocol and the idea is that, clever and proper integration is very much necessary for a complete end to end security.

(Refer Slide Time: 52:16)



### Points to Ponder

- Consider the following protocol (known as Otway-Rees protocol):  
A→B: M, A, B, {N<sub>A</sub>, M, A, B}<sub>K<sub>AS</sub></sub>  
B→S: M, A, B, {N<sub>A</sub>, M, A, B}<sub>K<sub>AS</sub></sub>, {N<sub>B</sub>, M, A, B}<sub>K<sub>BS</sub></sub>  
S→B: M, {N<sub>A</sub>, K<sub>AB</sub>}<sub>K<sub>AS</sub></sub>, {N<sub>B</sub>, K<sub>AB</sub>}<sub>K<sub>BS</sub></sub>  
B→A: {N<sub>A</sub>, K<sub>AB</sub>}<sub>K<sub>AS</sub></sub>  
Assume that the length of N<sub>A</sub>, M, A and K<sub>AB</sub> are same.  
Why do you think this can lead to a potential attack?  
(Hint: Note the adversary and the players observe the messages as bit strings and not as formal notations)



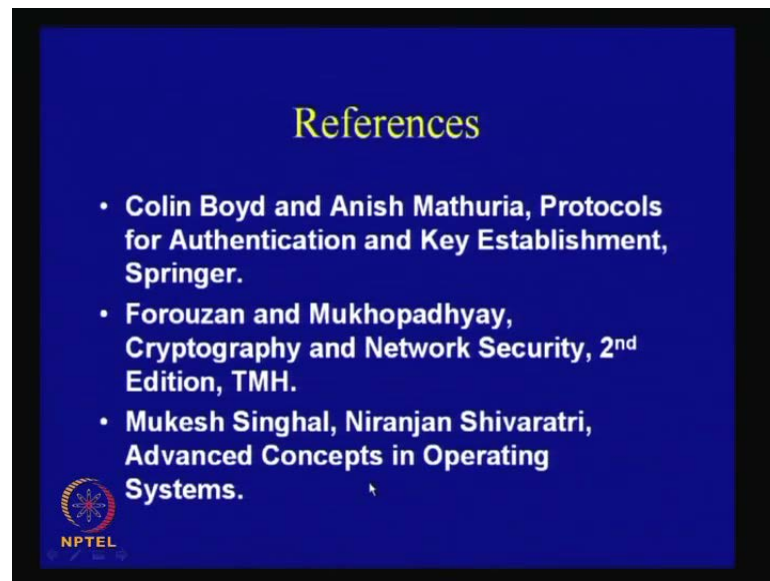
So, we have seen that, you have to apply your basic encryption tools and basic techniques in a clever way and ensure that, attacks are really, I mean, at least the known attacks do not work; that is... And we can actually consider this problem. This is the

problem, which is known as the famous Otway Rees protocol. So, here, what A does is that, A sends to B,  $M$ , this message  $M$  comma A comma B, and sends this packet  $N A$  comma  $M$  comma A comma B, which has been encrypted by  $K_{AS}$ , which is again a long term key. B sends to S,  $M$  comma A comma B and the second, this packet is  $N A$  comma  $M$  comma A comma B, which has been encrypted by  $K_{AS}$ ; the other part is  $N B$  comma  $M$  comma A comma B, which has been encrypted by  $K_{BS}$ .

The server S sends to B, this  $M$ , along with  $N A$  comma  $K_{AB}$ , that is, it has generated a session key  $K_{AB}$ , encrypted it by  $K_{AS}$ ; the other part is  $N B$  comma  $K_{AB}$ , which has been encrypted by  $K_{BS}$ ; and B sends to A,  $N A$  comma  $K_{AB}$  comma, which has been encrypted by  $K_{AS}$ . So, this is a classic protocol, but you can imagine that, if the lengths of  $N A$  comma  $M A$  and  $K_{AB}$  are same, then, this, there can be a potential attack against this protocol. So, you can just think about this, rather, how the attack can work. As a hint, you can note that, the first message and the third message has got a similarity; that is, this part and this part, you can observe that, this packet and this packet and this packet, have all been generated by  $K_{AS}$ . It has been encrypted by  $K_{AS}$ . So, you can just think or ponder about, whether this fact can be actually exploited for developing an attack and also note that, the adversary and the players observe the messages as bit strings and not as formal notations.

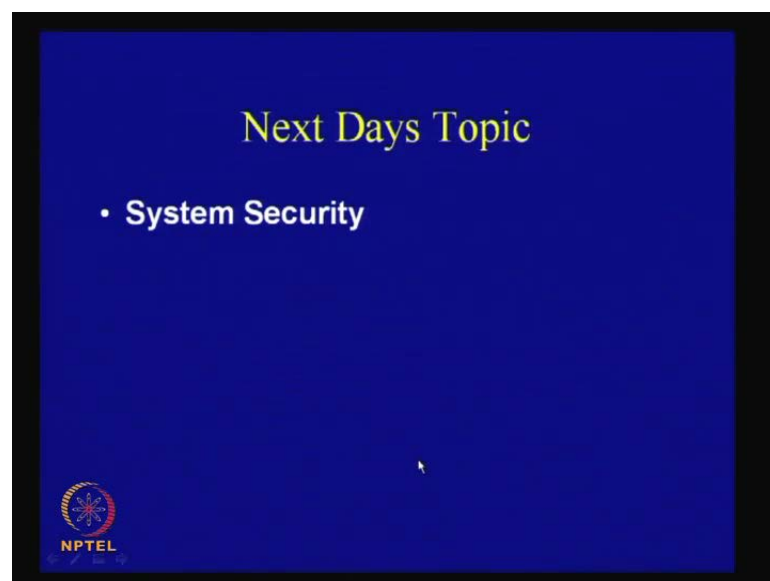
So, it will not observe this. Therefore, if there is an attacker, which can sit between the clients, which can actually send some bit strings which are not detectable by the receiver and can somehow fool this protocol, then, there can be an attack. So, therefore, you can just consider, whether this protocol is safe against a possible adversary.

(Refer Slide Time: 54:32)



So, I have actually used these references called Protocols for authentication and key establishment by Colin Boyd and Anish Mathuria very extensively for this particular lecture. The 5 state protocol, you can actually find detailed in this book and you can actually read it for more references; there is a huge representative of protocols present. And he has an excellent treatment on this subject. You can also refer to this other books, which also have been used. So, you can find more references on the Kerberos protocols in this particular text.

(Refer Slide Time: 55:03)



So, next day, we shall actually take up the topic on system security.