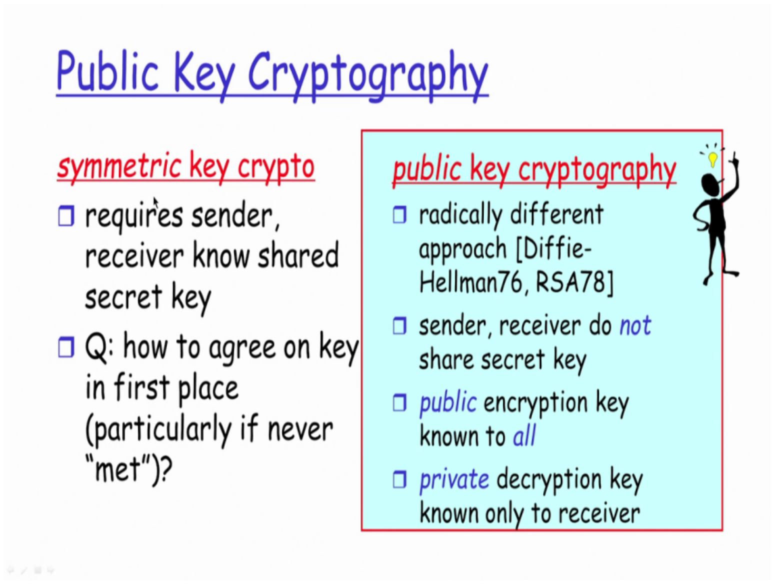
Information Security 3

Sri M J Shankar Raman,
Consultant Department of Computer Science and Engineering,
Indian Institute of Technology Madras

Module 53
Public Key Cryptography: An Introduction

So in this module we will basically talk about how a public key cryptography actually works as compare to a private key cryptography and also look at how is the authentication service typically implemented and what kind of different options that you have for implementing the authentication service in a typical application.

(Refer Slide Time: 00:32)



Public Key Cryptography

symmetric key crypto
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key cryptography
- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

So in symmetric key cryptography as we saw in the previous module, the sender and the receiver basically know the same shared secret key which will be used for both encryption and decryption.

So the sender will basically use the the shared symmetric key for encrypting which will generate the cIPher text and the receiver will use the same shared symmetric key for decrypting in the decryption algorithm and then generate back the original plain text but here the question that actually comes in is here is that how do I really have the sender and the receiver agree in the first place what key to use for doing this encryption and the decryption right.

So we cannot obviously say that I will first transmit the key in the plain text format so that the receiver knows what key I am going to make use of because if we do it in that manner it is equally possible that our intruder that we saw in the previous module has been listening on that particular exchange also through which the intruder will come to know what is the key that is being used and then it basically does not prevent him there is nothing that prevents him from using the same key to get back the original plain text when he is completely unauthorized to do so right.

So this becomes a challenge here where it is a question on how the sender and the receiver here will actually agree on what key to be made use of for doing the the symmetric key encryption and decryption on the sender and the receiver side respectively, so with this challenge we had the public key cryptography introduced where it was actually taken in a very innovative approach where the sender and the receiver do not share the same secret key and I have something called as a public key and I have something called as a private key right.

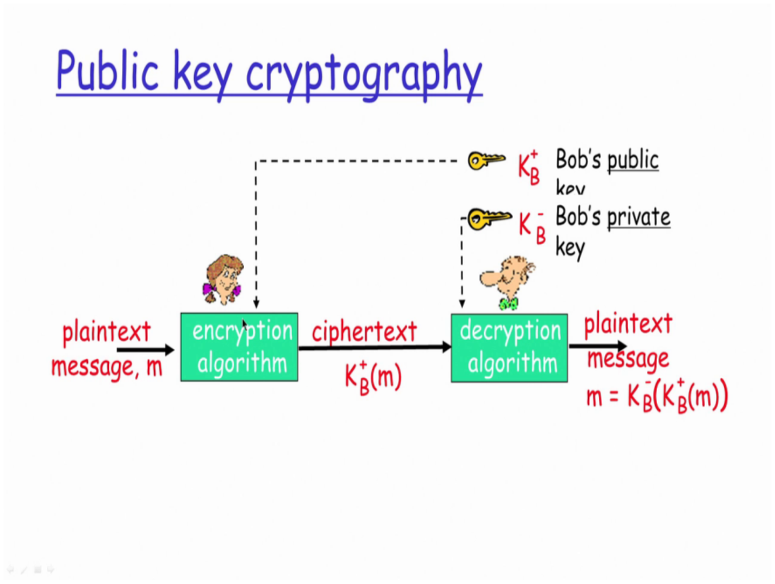
So the public key is the encryption key that is actually known for everybody right, so if I am actually valid user on the system I have a public key and a private key and I basically announce the public key to everybody in the system in the network so if anybody has to sent data to me they have to use my public key encrypt the data and then send it to me whereas the private key that I actually have I don't announce it to everybody and it is considered to be a secret right.

So in in in in a normal parlance when we say it is a secret we generally have a practice that if somebody is a close friend we basically go ahead and tell to him that ok this is a secret do not tell everybody and this is the message that I want you to considering it as a secret right but in the network security world the definition of a secret here is that it is not even known to the second person at all right, so with that definition of a secret the private key is called as a secret key.

Now what does this is been so this private key that I am suppose to be having for me should not be known to any other user or any other node in the network and it should be known only to me right. So we have now two keys one which is the public key which is to be used for encryption purposes so whenever somebody has data sent to me, they will use my public key and then encrypt it and send it me and then I have something called as a private key which I don't announce to anybody else.

And which I keep it to myself right. As a secret which will be typically used for decryption, decryption of the cipher text that is actually coming in from the sender to me right, so that is the key that I will use for decrypting the cipher text and regenerate back the original plain text, right?

(Refer Slide Time: 04:27)



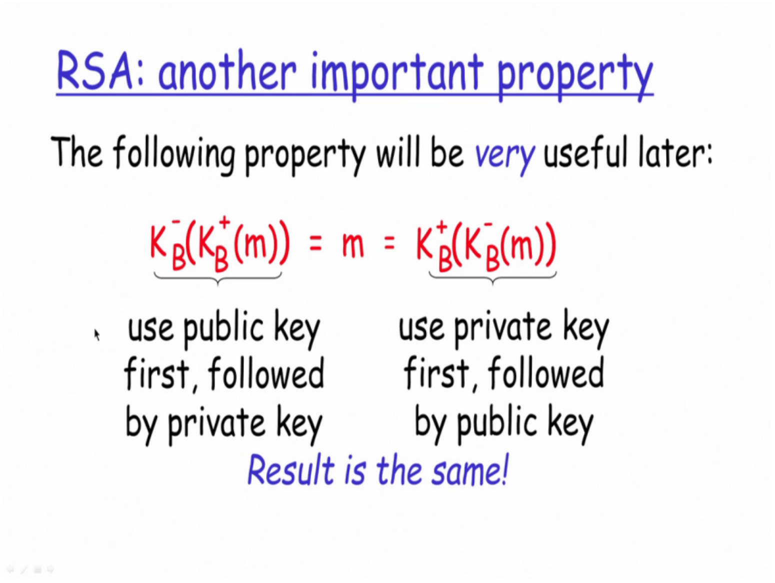
Now if you see this here so encryption algorithm is basically taking the plain text message and will use the public key of the receiver so Bob is a receiver here so I am going to use a public key of the receiver get the plain text message the encryption algorithm will going to do some magic generate the cipher text and the cipher text will be sent on the wire on the receiver side.

If Bob receives that cipher text message Bob will basically applies private key right, on the encryption algorithm and then he will generate the original plain text message back right. Now so the cipher text is basically application of Bob's public key on the plain text message and how is the plain text message generated back on this cipher text that has come on the wire, Bob basically applies this private key right, so magically the original plain text that had actually, introduced by the sender ok.

As an input to the encryption algorithm on the sending side gets regenerated back right, now how is this possible so Bob basically in this particular example has a private key and a public key pair right and this public key and private key pair is actually generated from a very large random number, random prime number and the public key and the private key has to be generated from the same random prime number based on certain mathematical principles.

So as long as these two keys the private key and the public key or generated with the same random large prime number and number two they satisfy this mathematical principles that are required to be satisfied it is guaranteed that even if the encryption algorithm as used this public key part of that pair right. The decryption algorithm can use the corresponding private key of that pair and then still even then I will be able to get back the original plain text message successfully right.

(Refer Slide Time: 06:44)



RSA: another important property

The following property will be *very* useful later:

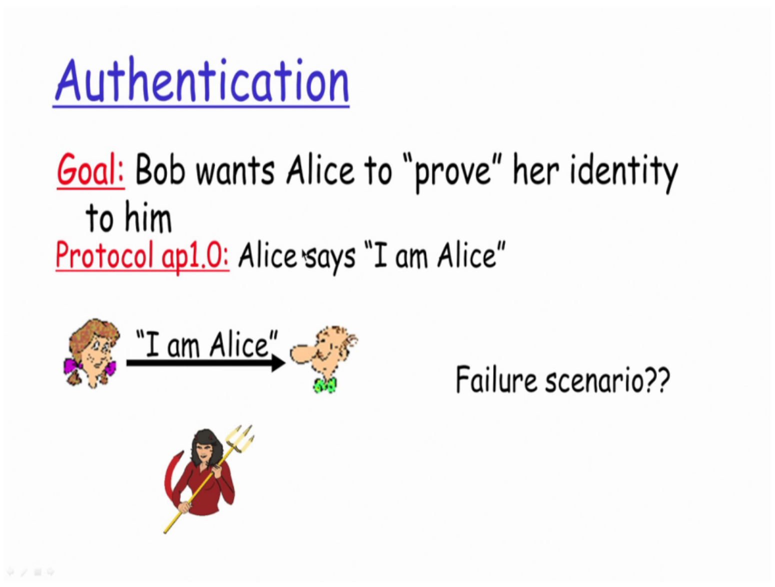
$$\underbrace{K_B^-(K_B^+(m))}_{\substack{\text{use public key} \\ \text{first, followed} \\ \text{by private key}}} = m = \underbrace{K_B^+(K_B^-(m))}_{\substack{\text{use private key} \\ \text{first, followed} \\ \text{by public key}}}$$

Result is the same!

So that is how the public key cryptography actually works right, on the other hand for some of the other purposes we will also be actually reversing this property of the RSA right, wherein I basically use the public key first followed by the private key to get the message and at other side I might also use a private key first followed by the public key so in in both cases we will always find that the original message is what is getting generated right.

So whether I apply the public key first and then apply the private key or I apply the private key first and then apply the public key in both cases I will be able to get back the original message, so this is a very very important property of the RSA algorithm which we will find very useful for implementing some of the other network security principles later on that we will be seeing.

(Refer Slide Time: 07:42)



Authentication

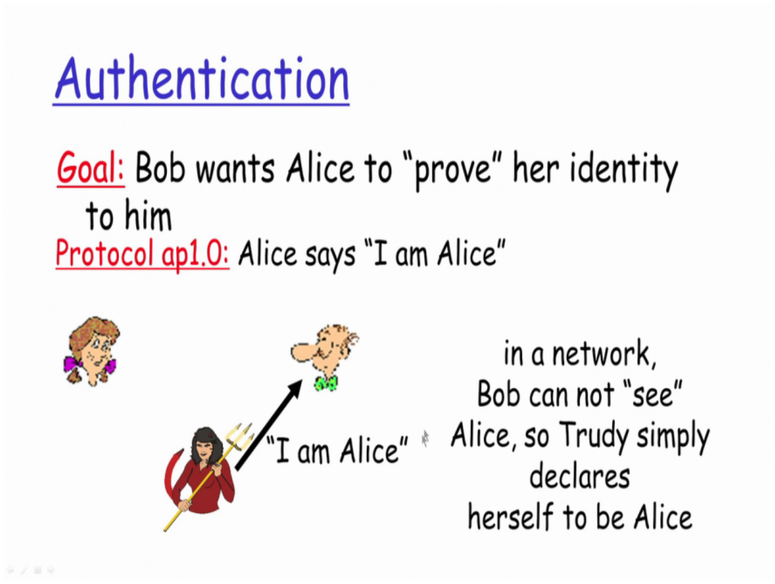
Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

"I am Alice"
Failure scenario??

So coming to authentications so as we discussed in the previous module. With the authentication what we essentially want to do here is that I want to make sure that the other person is really the person who he or she is claiming to be right, so in this particular example Bob wants to know that Alice should prove her identity to him right, so what could happen is very simply Alice can say I am Alice right, so that could be a message that is actually coming in saying I am Alice to Bob and what could be a failure scenario here to provide a very full proof authentication mechanism is that.

(Refer Slide Time: 08:23)



Authentication

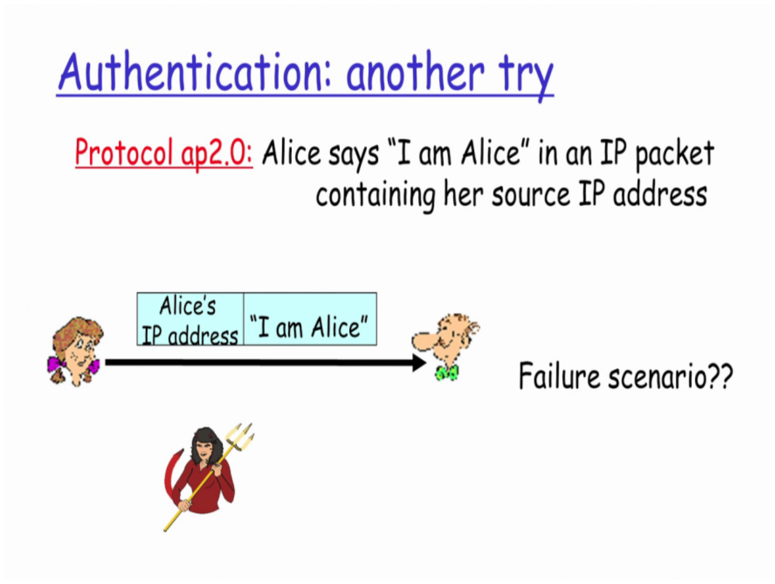
Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

"I am Alice"
in a network, Bob can not "see" Alice, so Trudy simply declares herself to be Alice

Now Trudy can also get into the network and then say I am Alice right, so there is no way by the simple mechanism of authentication for Bob to really know that actually Trudy is really sending a message trying to act as if he is somebody else right as if she is Alice right, so this basically has this basic problem of not even providing the most simple authentication mechanism.

(Refer Slide Time: 08:49)



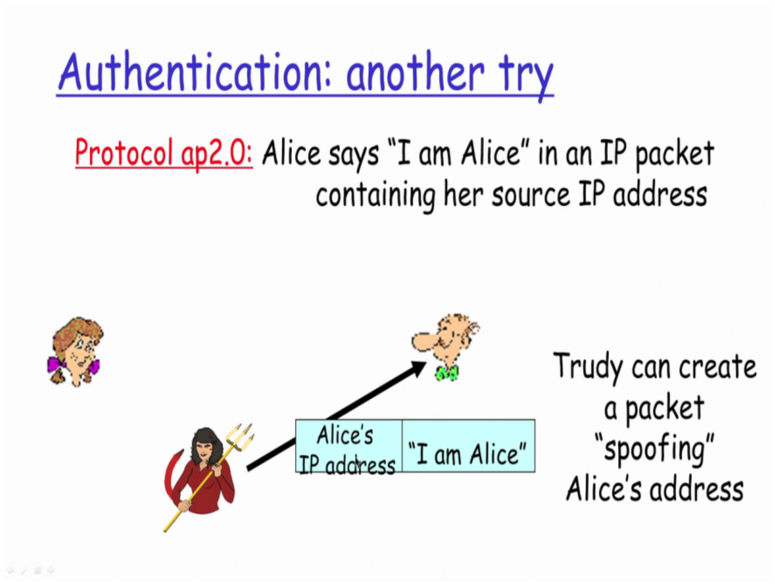
Authentication: another try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

Alice's IP address
"I am Alice"
Failure scenario??

So what can I do as an next evolution in this is that Alice can basically build a packet in which Alice IP address will be there followed by the message that I am Alice right and then send it to Bob. Now if Bob has basically by any any chance a list of mappings of IP addresses to users then Bob can really verify whether the IP address that is actually come is basically Alice's IP address in this message or somebody else's IP address and thereby sort of confirm right, that it was indeed Alice and not anybody else right.

(Refer Slide Time: 09:28)



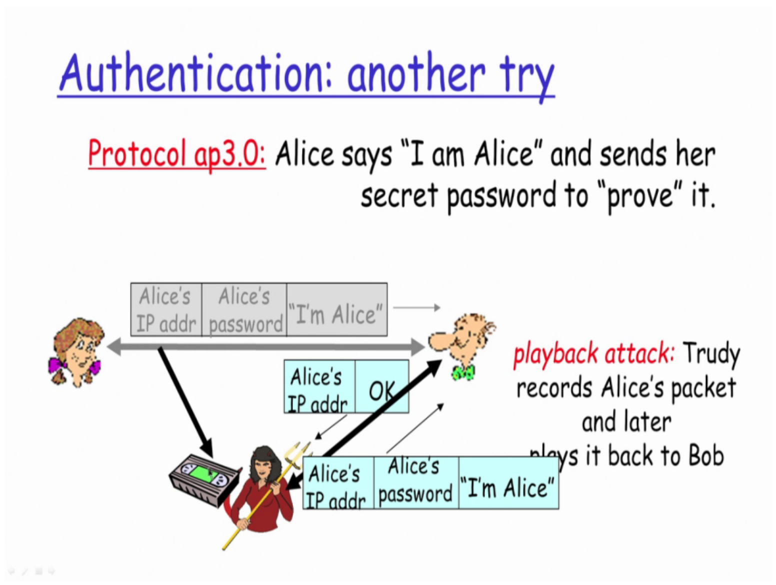
Authentication: another try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

So what could be a failure scenario here, Trudy can also create a packet by spoofing Alice's IP addresses and then putting the message as I am Alice and then send it to Bob right, so if for some time Alice is not in the network, Trudy will be able to do this because if both of them are in the network at the same time and Alice's Trudy's generating this kind of a packet there is a possibility that, it the network might detect it as a duplicate IP address from a different source and then act on it.

But if there is a situation where for example Alice is actually not logged in right now or not up in the network and in kind of a scenario, in that kind of a scenario Trudy is basically generating this packet and then sending it out Bob will not have any means to verify that it was really Trudy who was actually trying to spoof Alice IP address an then adding the message I am Alice and sending it to him. So this also gets a problem where our required expectation of authentication is actually not met, right?

(Refer Slide Time: 10:30)



Authentication: another try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

So what I can really do as a next step is that along with IP address and the message I can add a password, Alice password to sort of prove to Bob that ok so this is the password that I am using which again Bob verify with something which is internally which he has an internal database with him in some format to verify whether the IP address and the password and the message is actually mapping onto whatever he knows as Alice IP address and the password locally at his side.

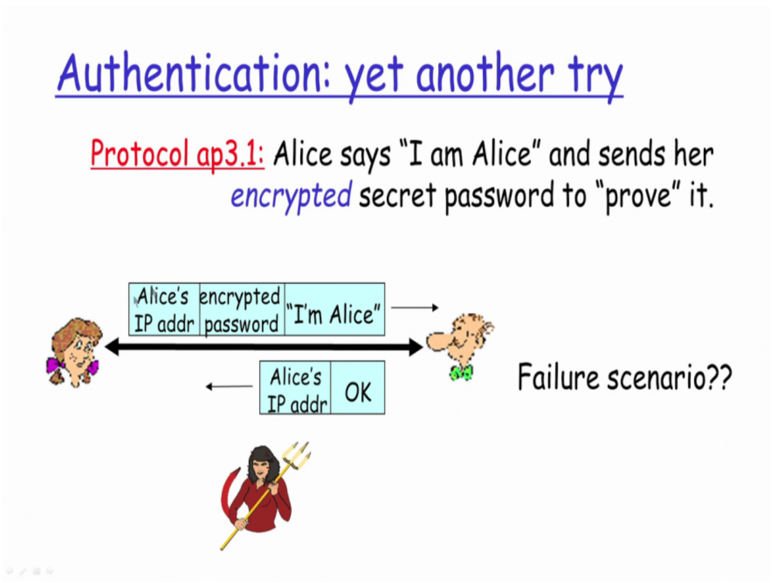
And then he confirms back to Alice saying that ok I accept you now we can actually start the exchange, data exchange right. So with the response message like like this and this format right, so now what's the failure scenario here, if Trudy has actually been doing a plain eves dropping of the message exchange between Alice and bob, so as we discussed eves dropping is a first type of activity that the person, the intruder will actually try to do just to get an idea of what kind of notes are there.

What kind of users are there, what is the pattern of the exchange that is actually happening and all that so if Trudy as actually been doing this role of eves dropping on the network trying to identify the packet flow from different people to different people, they can, the hacker can actually basically record this particular sequence and then do a playback right, so it’s like our normal audio video kind of a record and playback where if we put a cd we can just do a playback of whatever song.

Or whatever sequence that we want to hear it any point in time likewise Trudy can decide at a particular time in time to playback the recorded text that she had eves drop originally right, and at that time Bob can really think that ok it is right now Alice who has again come back on the network and start the communication but actually it would have been Trudy who is actually posing as an Alice by taking her IP address, by taking her password.

And also very easily putting the message as I am Alice right, so which all this information has actually been captured by Trudy as an attacker by just doing a plain eves dropping of the original messages that has been going between Alice and Bob previously right, some time back so this is basically what is refer to as a record and a playback attack, right?

(Refer Slide Time: 13:02)

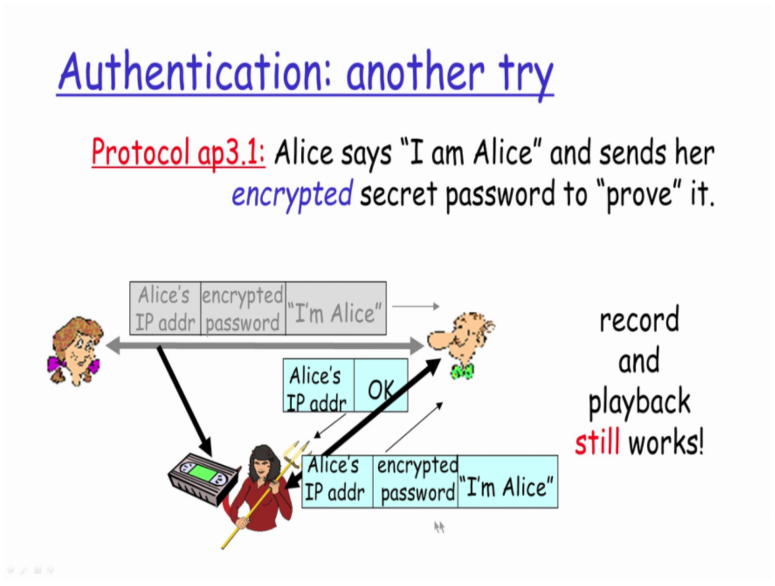


Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.

So I could do an encrypted password for Alice, and then Alice can send it out which is again not preventing the the scenario of a record and playback.

(Refer Slide Time: 13:10)



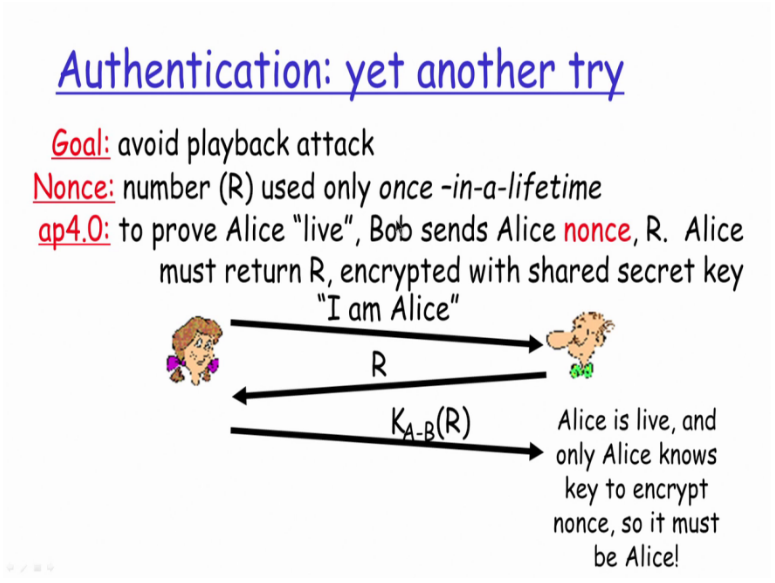
Authentication: another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.

record and playback still works!

I could actually record the entire sequence and then Trudy can as an attacker when she had eves drop got the original message and then played it back to Bob at a later point in time including the encrypted password which will again lead bob to believe that ok this is basically the original Alice and all the things or actually matching with what Alice should have sent me so and thereby Bob will really look at Trudy as an Alice and then start communicating with her as if she is Alice.

(Refer Slide Time: 13:45)



Authentication: yet another try

Goal: avoid playback attack

Nonce: number (R) used only *once -in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

So in the previous slide we actually saw at version of the authentication mechanism where we found out that there is a possibility of a playback attack happening, so the in the next version of the authentication mechanism we will try to see a way or a technique by which we can try to avoid this playback attack, so how do we try to avoid the playback attack, we basically use something called as a nonce value, now definition of a nonce value in network security is something.

It's actually a number which can used and valid only once in a lifetime right, so in order to prove that the other party Alice is live Bob will basically try to send a value, nonce value R to Alice right, now what should actually happens here is that with the help of a shared private key, a secret key that Bob and Alice is already having, Alice will be expected to return back this nonce value encrypted with that shared secret key right.

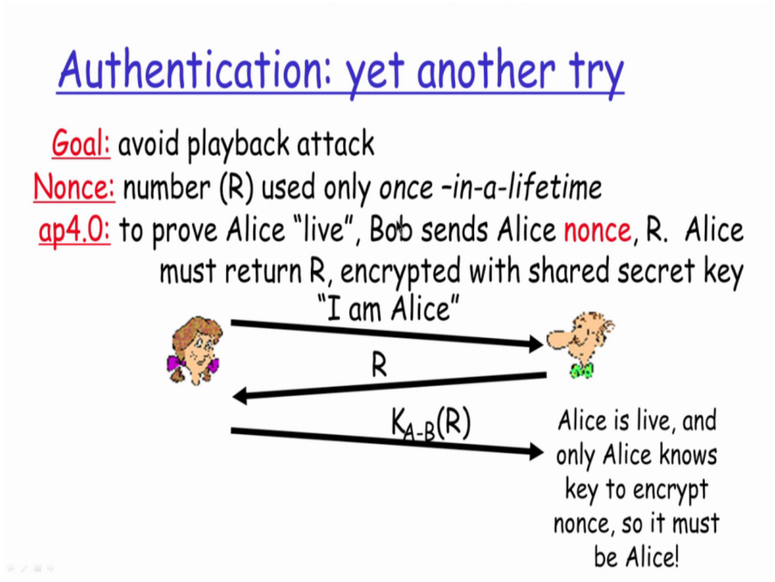
So for example let say that Alice in this particular case is figuring the conversation with Bob so by saying that I am Alice and now Bob is basically going to be actually sending out the nonce value that has been generated by him at his side to Alice right, now what is actually expected is that by the help of the shared private key which both Bob and Alice is actually sharing already, Alice will be expected to encrypt that nonce value that has been received from Bob.

And then send it back to Bob, now when Bob is basically receiving this encrypted values since the key is available with him also, he can basically make use of that key and decrypt it, decrypt this value regenerate the nonce value out and verify whether was it the same nonce value that was originally sent by him to Alice, now if if it had been subjected to a man and the middle attack, it would be very clear that this k a of b, that is the the shared secret key between Bob and Alice.

Would not be available with the person who is actually acting as an intruder now and because of which this particular encryption of the nonce value is going to be failing to result at the Bob side with the original nonce value getting generated out right, because there is a key would not be available with the intruder and if he sorts of goes into a brute force mechanism for example and tries to use any key, the original nonce value that Bob had generated is not going to be getting regenerated.

Then Bob tries to decrypt this encrypted nonce value on receipt of the encrypted nonce value back at this side right, so because of which here we actually try to avoid the the playback attack that was very easily possible in the mechanism and the authentication mechanism that we were talking in the the previous slide.

(Refer Slide Time: 17:04)



Authentication: yet another try

Goal: avoid playback attack

Nonce: number (R) used only *once -in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

"I am Alice"
R
$K_{A-B}(R)$
Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

So in order to ensure that the problem of the symmetric key being shared earlier, between the two parties is also eliminated.

Instead of using a symmetric key encryption for this encryption of the nonce value we could actually use a public key cryptographic technique where as we discussed I will basically have the authentication, the encryption done with the public key of the receiver right and then he decrypts it with the private key even get back the original nonce value hopefully and with that original nonce value Bob is really confirmed that it was actually the the other party Alice.

Who had actually sent, received the original nonce value from him and then has sent the encrypted value back right now right, so with this kind of an approach I will basically be able to have a very secure authentication mechanism wherein the two parties have actually authenticated themselves successfully, right? Here there is only one level of authentication but in certain situations both the parties will sort of challenge each other to authenticate themselves one after the other.

And there are situations like for example there is a protocol called chap protocol, the challenge access protocol in which there is a version of it called as a mutual chap where both sides try to challenge each other and sort of confirmed themselves internally that ok the other party is really the party that it is claiming to be right, so in which case I will have the mechanism of authentication done on both sides after which only, only if this mutual authentication is successful the data transfer will be allowed to actually happen.

Thank you.