**RSA Encrypted Key Exchange Protocol Analysis**

***Protocol assumptions***

When modeling the RSA encrypted key exchange protocol there was only one strong assumption that needed to be made. That was that ‘e’ is uniquely generated every time the protocol is run i.e. **(uniq-gen e)**. This is because on page 4 of the encrypted key exchange paper, it mentions that the public key, EA, is generated from large numbers. These numbers are uniquely generated each time for the RSA protocol, so it holds that in turn the key ‘e’ itself is uniquely generated. That’s why such a strong assumption can be made in-protocol.

***RSA Security Goals***

The security goals of RSA are mutual authentication and to prevent a dictionary attack of the secret password know only to the initiator and responder. Mutual authentication is confirmed once the initiator sends the responder its challenge back encrypted under the generated symmetric key, ‘r’. So, after the 5th message the mutual authentication should be proven. A dictionary attack is when an attacker uses messages encrypted under a specific password to crack the password. The attacker would need the messages encrypted under ‘p’ to see if their guessed password is correct.

***Initiator’s Point of View***

Chart

Description automatically generatedText, letter

Description automatically generated

**Mutual Authentication**

This is a realized shape showing the partial execution of the RSA encrypted key exchange protocol. To check for mutual authentication there were 2 skeletons coded. It’s important that the known secret password ‘p’ stays secret between init and resp. So, we add **(non-orig p)** in each skeleton. And each of the challenge from either party should be fresh so we include **(uniq-orig ca)** and **(uniq-orig cb)** for the init skeleton and resp skeleton respectively. Looking at this shape, the dashed first arrow lets us know that the message the responder is receiving is not the same as the message the initiator sent. We also know this because the skeleton has a-0 in the variables and the strand. However, the second message that is sent by the responder and received by the initiator is the correct message. But the mutual authentication is not proven by this shape because init or resp could not complete the protocol past the second message to get to mutual authentication at the 5th message.

***Responder’s Point of View***

Chart

Description automatically generatedText, letter

Description automatically generated

**Mutual Authentication**

This is a realized shape showing the partial execution of the RSA encrypted key exchange protocol from resp’s point of view. To check for mutual authentication there were 2 skeletons coded. It’s important that the known secret password ‘p’ stays secret between init and resp. So, we add **(non-orig p)** in each skeleton. And each of the challenge from either party should be fresh so we include **(uniq-orig ca)** and **(uniq-orig cb)** for the init skeleton and resp skeleton respectively. Similar to the last shape, we know the attacker is listening to the first message because the dashed line means the message received isn’t what it was supposed to be. We can also see that a-0 is in the skeleton. The messages between init and resp stop after the initial message, so mutual authentication was not completed between init and resp. Not even authentication of one party was completed.

**Leaked Password**

Chart

Description automatically generated

To watch for a dictionary attack, 2 skeletons were coded with the same assumptions, with the same reasoning, as the skeletons original init and resp skeletons above**, (non-orig p), (uniq-orig ca), and (uniq-orig cb)** and for all the same reasons. In each of these 2 skeletons, a deflistener strand was included to listen for the message contents encrypted under the secret password ‘p’. Although this is not a realized shape, the red first node for resp lets us know resp didn’t receive the contents of that first message, and the blue first node of the listener means the listener was able to get contents of the message as if a dictionary attack was possible. Although the correct assumptions were made according to the paper, that first listener strand node is blue as if the dictionary attack was successful so the protocol is flawed, because we know the dictionary attack didn’t occur because there are no arrows between init/resp and the listener strand.

**Diffie-Hellman Encrypted Key Exchange Protocol Analysis**

***Protocol assumptions***

When modeling the Diffie-Hellman encrypted key exchange protocol there were a few strong assumptions that needed to be made. That was that the exponents x and y, x for the initializer and y for the responder, are uniquely generated every time the protocol is run i.e. **(uniq-gen x)** and **(uniq-gen y)**. This is because an important aspect of the plain Diffie-Hellman protocol is that while the “generator” is public, the initiator and responder both generate their own secret components, x and y respectively. And these are fresh each time you do the protocol. That’s why such a strong assumption can be made in-protocol.

***DH Security Goals***

The security goals of Diffie Hellman are mutual authentication and to prevent a dictionary attack of the secret password, ‘p’ known only to the initiator and responder. Mutual authentication is confirmed once the initializer sends the responder its challenge back encrypted under the generated symmetric DH key. So, after the 4th message the mutual authentication is proven. A dictionary attack is when an attacker uses messages encrypted under a specific password to crack the password. The attacker would need the messages encrypted under ‘p’ to see if their guessed password is correct. So, those messages are where the attacker is looking.

***Initiator’s Point of View***

Chart

Description automatically generatedText, letter

Description automatically generated

**Authentication Property**

This is a realized shape showing the full execution of the Diffie Hellman encrypted key exchange protocol. To check for mutual authentication there were 2 skeletons coded. It’s important that the known secret password ‘p’ stays secret between init and resp. So, we add **(non-orig p)** in each skeleton. And each of the challenge from either party should be fresh so we include **(uniq-orig ca)** and **(uniq-orig cb)** for the init skeleton and resp skeleton respectively. This shape tells us that mutual authentication has completed since the 4th message was sent and received between init and resp.

***Responder’s Point of View***

Chart

Description automatically generatedText, letter

Description automatically generated

**Authentication Property**

This is a realized shape of the partial execution protocol from resp’s perspective. Again. it’s important that the known secret password ‘p’ stays secret between init and resp. So, we add **(non-orig p)** in each skeleton. And each of the challenge from either party should be fresh so we include **(uniq-orig ca)** and **(uniq-orig cb)** for the init skeleton and resp skeleton respectively. The dashed first arrow lets us know that the message the responder is receiving is not an equal message. That is, the attacker likely sent a message to resp in place of init. We know this because the output skeleton also has a-0 in the variables and the initiator strand. However, the next two messages that were sent/received were equal messages so mutual authentication is still a possibility. But mutual authentication cannot be proved with this shape because the 4th message was not sent and received. However, the authentication of init has been proven, but because this shape does not have an arrow to a 4th node under init, resp has not authenticated itself.

**Leaked Password**

To watch for a dictionary attack, 2 skeletons were coded with the same assumptions, with the same reasoning, as the skeletons above, **(non-orig p), (uniq-orig ca), and (uniq-orig cb)** and for the same reasons. Except in these 2 skeletons, you include a listener strand to listen for the contents encoded under the secret password ‘p’. No realized shapes were produced from the listener strand skeletons. And all the unrealized ones had red nodes for the first node in the listener strand. This indicates that any dictionary attack would be unsuccessful on this protocol.