**Needham Schroeder Symmetric Key Protocol**

The Needham–Schroeder Symmetric Key Protocol, based on a symmetric encryption algorithm. It forms the basis for the Kerberos protocol. This protocol aims to establish a session key between two parties on a network, typically to protect further communication.

**Security Goals of Needham Schroeder Symmetric Key Protocol**:

When we use a conventional encryption algorithm with Needham Schroeder protocol, then we assume that each participant in the protocol has a secret key that is known only to itself and to its authentication server, the contents of which are accordingly secret. The essential step in setting up secure communication between A and B is for the initiator, say A, to generate a message with two properties:

(a) It must be comprehensible only to B, i.e. allow only B to use its contents to identify himself to A.

(b) It must be evident to B that it originated with A.

**Protocol Implementation:**

A diagram of a computer

Description automatically generated

Figure 1: Needham Schroeder Symmetric Key Protocol

The Figure 1 above summarizes how Needham-Schroeder Symmetric Key Protocol works, during which Alice (A) and Bob (B) establish a session key.

Here, Alice (A) initiates the communication to Bob (B). S is a server trusted by both parties. In the communication:

* **A** and **B** are identities of Alice and Bob respectively.
* **Kas** is a symmetric key known only to **A** and **S**.
* Kbs is a symmetric key known only to B and S.
* Na and Nb are nonces generated by A and B respectively.
* Kab is a symmetric, generated ket, which will be the session key of the session between A and B.

The protocol can be specified as follow:

1. A -> S: A, B, Na

Alice sends a message to the server identifying herself and Bob, telling the server wants to communicate with Bob.

1. S -> A: {Na, Kab, B, {Kab, A}Kbs}Kas

The server generates Kab and sends back to Alice a copy encrypted under Kbs for Alice to forward to Bob and also a copy for Alice. Since Alice may be requesting keys for several different people, the nonce assures Alice that the message is fresh and that the server is replying to that particular message and the inclusion of Bob's name tells Alice who she is to share this key with.

1. A -> B: {Kab, A}Kbs

Alice forwards the key to Bob who can decrypt it with the key he shares with the server, thus authenticating the data.

1. B -> A: {Nb}Kab

Bob sends Alice a nonce encrypted under Kab to show that he has the key.

1. A -> B: {Nb - 1}Kab

Alice performs a simple operation on the nonce, re-encrypts it and sends it back verifying that she is still alive and that she holds the key.

**Modelling Needham Schroeder Symmetric Key in CPSA**:

Using CPSA, we analyze Needham Schroeder Symmetric Key protocol in the Dole-Yao network intruder model. We will discuss how we model the protocol in CPSA and interpret the shapes produced by CPSA.

1. In this protocol, we have three roles: Alice, Bob and Authentication Server. Therefore we have to define three roles in the protocol definition.
2. We have to declare Alice’s identity, Bob’s identity and Authentication Server’s identity as names for alice, bob and authentication server.
3. Alice will create a nonce Na and also receive a nonce Nb from Bob. So we declare both as text in Alice’s role. Authetication server receives the nonce Na so we declare it in Authentication server. Bob never receives the nonce from N as Na so we declare only Bob’s nonce Nb in Bob’s role.
4. We also declare a symmetric session key sk in all the roles.
5. Additionally, we have a message m in alice’s role because Alice can never know what message was encrypted under Bob’s secret key. So we declare it as a message m from Authentication server to Alice.
6. We define two skeletons, one in Alice’s perspective and one in Bob’s perspective.
7. We declare the long-term keys of alice and bob with authentication server are never leaked. So, we declare them as non-originating.
8. We also declare nonces Na and Nb as uniquely generated for every session and are never repeated.

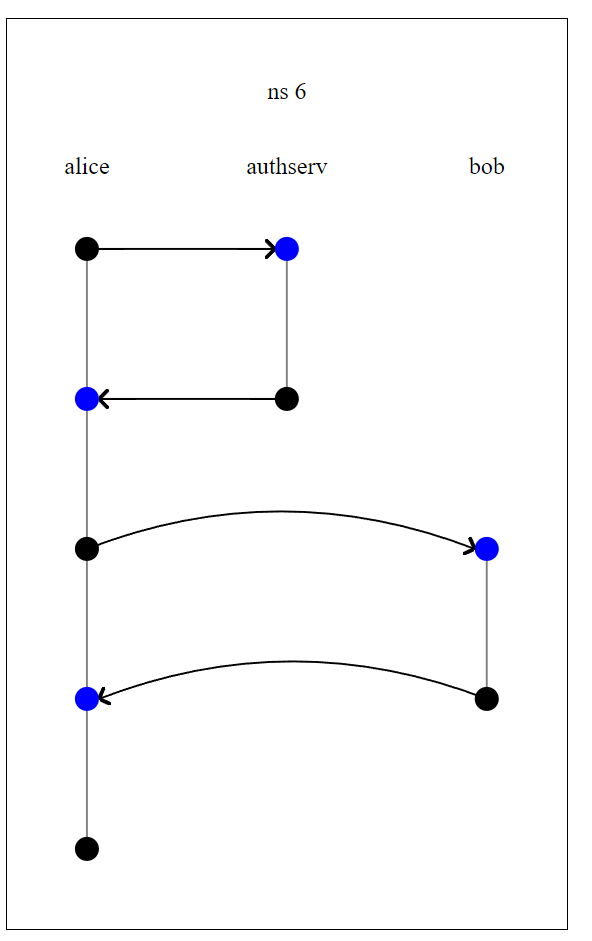
**Assumptions Made for Needham Schroeder Symmetric Key Protocol:**

We assume that the long-term keys for Alice and Bob are kept secret and never leaked or shared with someone else. We also assume that the session key generated between Alice and Bob is kept secret and never leaked or shared to an outside party.

**Analyzing the shapes generated by CPSA:**

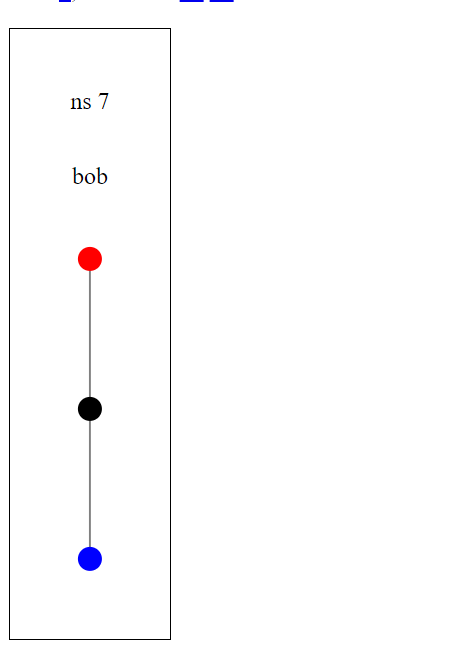
**Alice’s point of view:**

1. A line of dots with black and red dots

   Description automatically generatedThe first shape is a single strand of Alice where CPSA is trying to complete the communication without needing to communicate with Bob or Authentication server and it fails to do so.
2. The second shape shows us a complete communication between Alice, Bob and Authentication server where we have solid lines from all the nodes except the last one. The last message sent from alice to bob which is a simple operation on nonce Nb and sent back to Bob. We don’t know if the message was received by Bob because we never get confirmation back from Bob.

**Bob’s point of view:**

1. In Bob’s point of view, we again see CPSA trying to complete the communication to Bob without needing any other strand which it fails to do so as seen in the diagram with a red node.



1. In Bob’s perspective, we see a different shape. We have a complete communication, but we don’t have a solid line from Alice to authentication server. This is because the first message has Alice’s identity, bob’s identity and a random nonce sent in clear text. This message can be completed by anyone apart from Alice who knows the identities of Alice and bob. That is why we don’t have a solid line from Alice to authentication server.

A screen shot of a diagram

Description automatically generated

1. There is one more shape, where we have two strands of Alice with only one strand of Bob and Authentication server respectively. This is possible because Alice can initiate the protocol twice intentionally or accidentally with Bob. As seen previously, we don’t have a solid line from Alice to authentication server. Authentication server sends the message to both the strands of Alice. Alice sends the message to Bob, but bob can communicate with only one of the strands of Alice. Therefore, we have a single completion between Bob and Alice.

A diagram of a line

Description automatically generated with medium confidence

**Attacks on Needham Schroeder Symmetric Key Protocol:**

While the CPSA model does specifically reveal any attacks on the protocol itself, there can be a compromise on the protocol if the session key between initiator and responder gets leaked to the adversary. This attack was first documented in the paper “**Timestamps in Key Distribution Protocols**” by Dohorthy E. Denning and Giovanni Maria Sacco.

If the encryption algorithms are strong and keys random, it is unlikely that communication keys will be compromised by cryptanalysis. We are more concerned with the communication key's direct exposure due to negligence or a design flaw in the system, i.e., an intruder may be able to break into the AS or into A's or B's computer and steal a key. Let us suppose that a third party C has intercepted and recorded all the messages between A and B in steps (3)-(5), and that C has obtained a copy of the communication key Kab. C can then later trick B into using the Kab as follows:

1. First C replays the message Y to B:

C -> B: {Kab, A}Kb

1. Thinking that A has initiated a new conversation, B requests a handshake from A:

B -> A: {Nb}Kab

1. C intercepts the message, deciphers it, and impersonates A’s response:

A -> B: {Nb-1}Kab

Thereafter, C can send bogus messages to B that appear to be from A, intercepting and deciphering B's replies. Note, however, that A can still communicate safely with B by initiating a new conversation (assuming A's attempts are not blocked by C), and that B can communicate safely with other user.

**Modelling the attack in CPSA:**

In order to model a leaked key in CPSA, we deliberately leak the key when sending the message from authentication server to Alice. The original shapes are still valid because they are valid executions of the protocol but we get two new shapes:

1. In Alice’s point of view, we can see that the bob strand is no longer needed to complete the protocol. This is because once the secret key was leaked, we are not sure that all the communication that alice makes to bob is actually made to bob himself. The intruder has access to the secret key, so he can intercept alice’s message and reply back to alice and complete the protocol without ever needing bob in the exchange.

A screen shot of a drawing

Description automatically generated

1. In Bob’s point of view, as explained in the attack described by the paper, an intruder can replay a legitimate message of alice to bob to initiate a conversation. Bob, not knowing that the secret key was leaked will reply back to the intruder thinking that he is initiating the conversation with Alice. Therefore after the first message from Alice, bob can complete the exchange without ever needing the alice strand.

A screen shot of a drawing

Description automatically generated