Implementation of Needham-Schroeder-Protocol

Anshul Anand*, Jatin Goel*, Kritika Sharma* ,Rishabh Verma*,Varun Kumar*
Indian Institute of Information Technology, Allahabad
*icm2014501@iiita.ac.in
*icm2014503@iiita.ac.in
*icm2014502@iiita.ac.in
*icm2014004@iiita.ac.in
*icm2014008@iiita.ac.in

***Abstract*—In this work we have implemented the Needham-Schroeder-Symmetric Key Protocol ,a key transport/distribution protocol which employs a Key Distribution Centre to distribute session and long term keys between communicating parties.We have discussed the advantages/disadvantages of Needham-Schroeder-Protocol over other key distribution protocols and also the attacks possible on this protocol. We have also analyzed the protocol experimentally for the latency.**

***Index Terms*—Key-Distribution-Protocols , Network Security , Needham-Schroeder Protocol**

I. Introduction

Employing symmetric cryptography is more efficient and lightweight when compared to using asymmetric key cryptography for secure communication between two parties by using encryption and decryption.However we need a pair of shared keys between two parties in order to use symmetric cryptography.If there are N communicating parties then each party has to store N(N-1) keys. If we allow keys to be used bidirectionally it comes down to N(N-1)/2 . Apart from the storage overhead of the key , there is also the problem distributing keys between the communicating parties. Hence we look for a more efficient way to store and distribute the keys. We discuss Key Distribution Center (KDC) as a practical solution for the efficient key management problem.

A. Key-Distribution Center: KDC

KDC is a trusted third party , referred to as the Key Distribution Centre. Each person establishes a shared secret key with the KDC .

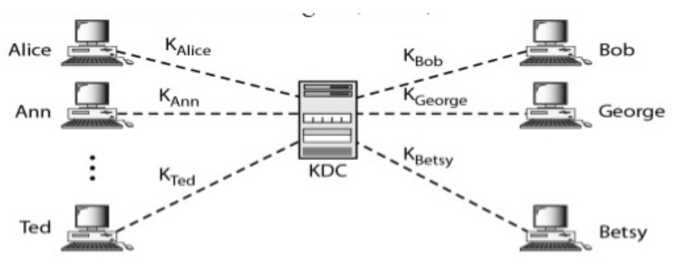

Fig. 1. Key-Distribution Center (KDC)

K_{Alice} represents shared secret between Alice and the KDC and similarly K_{Bob} represents shared secret between Bob and the KDC.

When Alice wants to communicate with Bob in order to send a confidential message , the process is as follows -

- The KDC is contacted by Alice inorder to get a temporary session key between itself and say Bob.
- KDC notifies Bob about the request of Alice.
- If Bob agrees , then a session-key is created between the two and distributed.

B. Session Keys

Each member has a secret key which is known to the member and the KDC , but not to any other member other than itself. If Alice wants to communicate with Bob , a shared session key will be created between Alice and Bob . When the communication is terminated the session key is no longer useful .

C. Needham-Schroeder-Protocol

Needham-Schroeder protocol establishes mutual authentication between two communicating parties say A and B in the presence of an adversary which is capable of -

- Intercepting messages
- Delaying messages
- Copying and eavesdropping on the message.
- Creating new illegitimate messages.

The actual transmission of messages in the protocol are as shown below in Fig 2. which also provides the meaning of notations used for each of the key. Further discussion of the protocol is done in detail in the Literature survey section.

II. Literature Survey

The protocol proposed by Needham and Schroeder in 1978 , is the basis of many new Key Management protocols including the widely used Kerberos authentication protocol suite. NS protocol is designed for mutual authentication via a shared-key. It uses a KDC to manage i.e generate and distribute the session key , which is a temporary symmetric shared key for symmtrically encrypted communication.

There are three parties involved in the protocol.

- Initiator

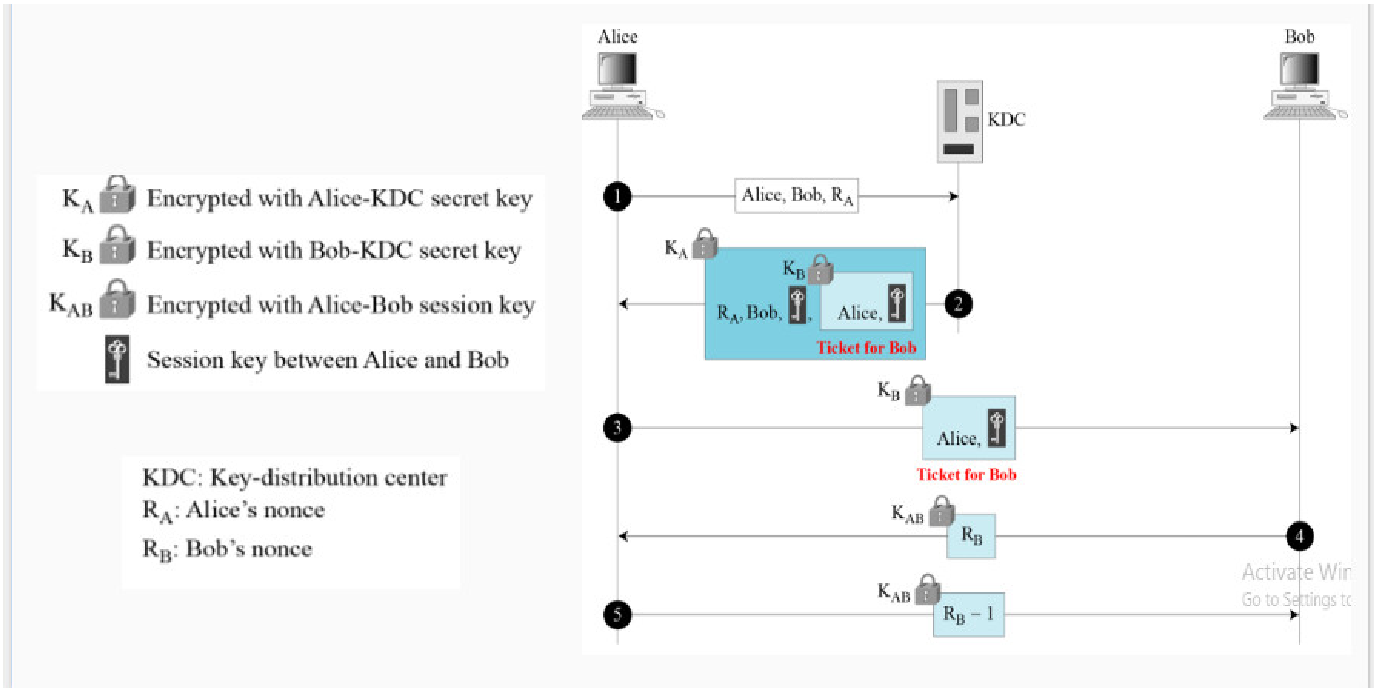

Fig. 2. Protocol Message Exchange Diagram

- Responder
- Key Distribution Centre.

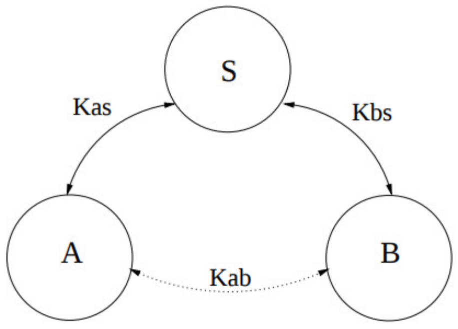

Fig. 3. Participating parties in NS Protocol

It is assumed that A and B already have secure symmetric communication with S using keys K_{as} and K_{bs} , respectively.

N-S uses nonces i.e randomly generated values included in messages. Nonces are used to check the freshness of the message , suppose A sends a nonce to B and B returns the same nonce back to A then it knows that the message is fresh.

Nonce is different from a Timestamp, the nonce is not repeated with high probability which is taken care by a pseudorandom number generator

The protocol is as shown in the Fig. 2

III. Flowchart Design

Needham schroeder protocol has been successfully implemented the basic architecture of which is given in Fig.4.

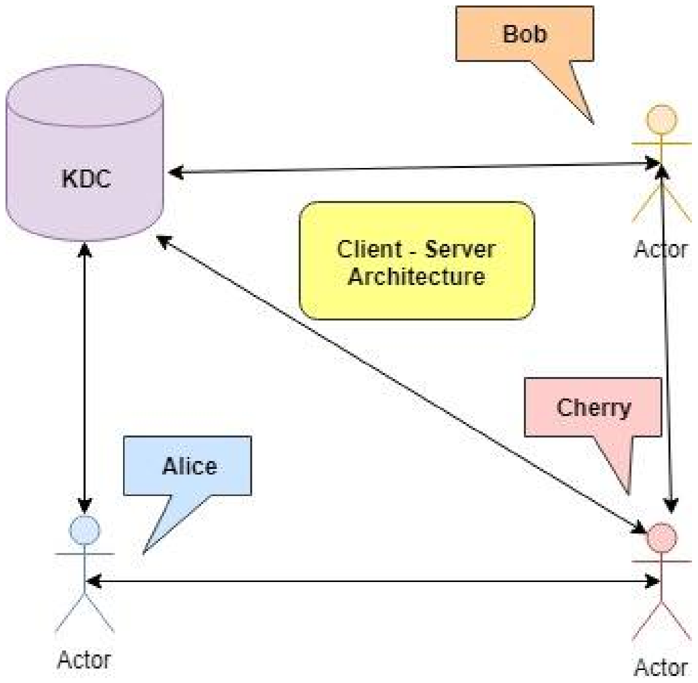

Fig. 4. Basic architecture of the implementation

In this implementation, there is KDC server running persistently. The KDC will be listening to all the clients continuously. Every client will be asked if he wants to connect to some other client in the network. If yes, then this client (Alice) will first decide whom he wants to connect to. Let say he wants to connect to another client, Bob. Assume K_A and K_B are the secret keys of Alice (initiator) and Bob (responder) respectively with KDC. Now, following steps are performed :

1) Alice will then send a message [Alice Bob R_A] (Message1) to KDC where R_A is a nonce generated by Alice.
2) Now, KDC will generate a session key K_{AB} for communication between Alice and Bob. It will then firstly form a ticket for Bob by encrypting [Alice K_{AB}] with K_B. Next, it will ecrypt [R_A Bob K_{AB} E_{K_B}[Alice K_{AB}]]] with K_A. Finally, it will send a message [E_{K_A}[R_A Bob K_{AB} E_{K_B}[Alice K_{AB}]] (Message2) to Alice.
3) Alice will now decrypt this message received from KDC using K_A. Hence, decrypted message is [R_A Bob K_{AB} E_{K_B}[Alice K_{AB}]]. So, Alice now knows the session key given by KDC. Now, Alice will forward the ticket E_{K_B}[Alice K_{AB}] (Message3) to Bob.
4) On the receiving the ticket from Alice, Bob will decrypt it using K_B. The decrypted message is [Alice K_{AB}]. Hence, Bob also knows the session key now.Bob will now generate its own nonce R_B and will encrypt this using K_{AB} obtained in the previous step. Then, it will send the encrypted message [$E_{K_{AB}}$[R_B]] (Message4) to Alice.
5) Alice will receive this message and will encrypt R_B-1 using K_{AB} and will again send this encrypted message $E_{K_{AB}}$[R_B-1] (Message5) to Bob for the purpose of authentication for futher communication.

The flow diagram for this implementation has been given in Fig.5.

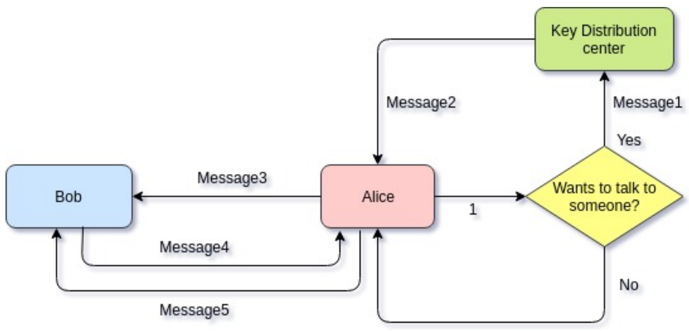

Fig. 5. Flow chart of the implementation

IV. Analysis

The total number of messages exchanged in NS-Protocol are - 5

The total number of keys stored by the KDC is at max N + N(N-1)/2 for N communicating parties as , it will store N secret keys and N(N-1)/2 for the session keys .

Attacks are discussed in the Discussion section.

V. Experimental Study

The protocol has been implemented using the concepts of socket programming in Python3. Following libraries have been used :

1) PyCrypto
2) threading
3) socket
4) IP
5) netifaces

We are running a KDC server persistently. Also, there are three clients in the network. Any of them can initiate a communication with any other client. Each session will be provided with a session key by the KDC for authentication purpose. Once the session key is established, further communication is continued with the encryption of messages using this session key.

We have used ARC4 (Rivest Cipher 4 also known as RC4) symmetric key encryption scheme which is provided by the Pycrypto Library. To use this encryption/decryption scheme we have to provide byte string as plaintext/ciphertext. Here are steps to use this inbuilt cipher scheme:

1) Convert Normal string to byte string using Encoding.
2) Use that byte string for encryption or decryption.
3) Convert byte string to Normal string using Decoding after encryption/decryption.

The protocol was run for a number of times and the latency in message delivery was recorded (in ms) and tabulated as shown below.

TABLE I
Needham-Schroeder Protocol

S.No.	Latency b/w Client and KDC	Latency b/w Clients
1	4.06	9.62
2	6.02	10.42
3	4.89	8.72
4	6.99	11.42
5	4.06	9.62
6	4.79	9.38
7	6.34	10.74
8	5.50	10.98

The average latency between client and KDC is - 5.49 ms
The average latency between clients is - 10.18 ms
A plot for ease of analysis is provided by Fig. 6

VI. Discussions

The NS Protocol overcomes the failures of the previous protocol like replay attack and MITM (Man-in-the-middle) attack for few exchanged messages and has time and space complexity as mentioned in the Analysis section of the report.

Here we discuss some of the attacks which are still possible on the NS Symmetric protocol:

- Replay Attack :
  Replay attack is possible in the NS protocol (Denning and Sacco [4]). Attacker can use an older value for

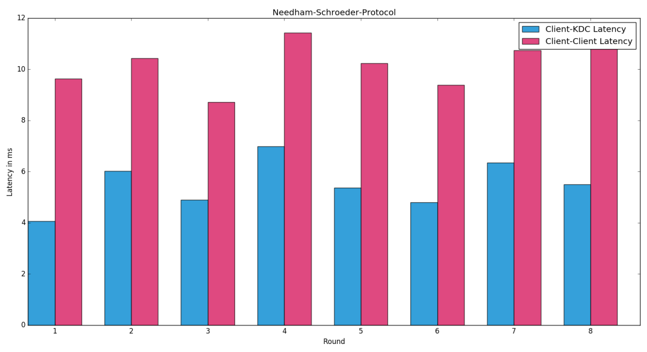

Fig. 6. Bar Graph showing Latency

K_{AB} which has already been compromised say via brute-force.He can then replay the message to Bob K_{AB} , A K_B, who will accept it,as it cannot judge the freshness of the message.
Problem : Message 3 is not protected by nonces. There is no way for B to know if the K_{AB} it receives is current. An intruder can retrieve or compromise the older key in a longer amount of time using brute-force approach on many of the messages it can eavesdrop on.

VII. Conclusion

In this report we presented the Needham-Schroeder Symmetric Key Management protocol and understood the underlying principles of network security involved in its design. NS-Protocol improves a lot over its previous ancestors but still has possible attacks over it. Understanding the NS-Protocol is especially beneficial in understanding the more recent key-management protocols like Kerberos which have the NS-Protocol as its basis.

VIII. References

[1] https://cgi.csc.liv.ac.uk/ alexei/COMP522$_10/COMP522-Needham-Schroeder-05.pdf$

[2] https://www.cs.utexas.edu/ byoung/cs361/lecture60.pdf

[3] http://www.edmath.org/MATtours/discrete/concepts/ciso.html [Last accessed on October 28, 2017]

[4] Dorothy E. Denning and Giovanni Maria Sacco. 1981. Timestamps in key distribution protocols. Commun. ACM 24, 8 (August 1981), 533-536. DOI=http://dx.doi.org/10.1145/358722.358740

[5] https://en.wikipedia.org/wiki/RC4