Tutorial 4

1.Needham-Schroeder Authentication Protocol

Consider the Needham-Schroeder Authentication Protocol:

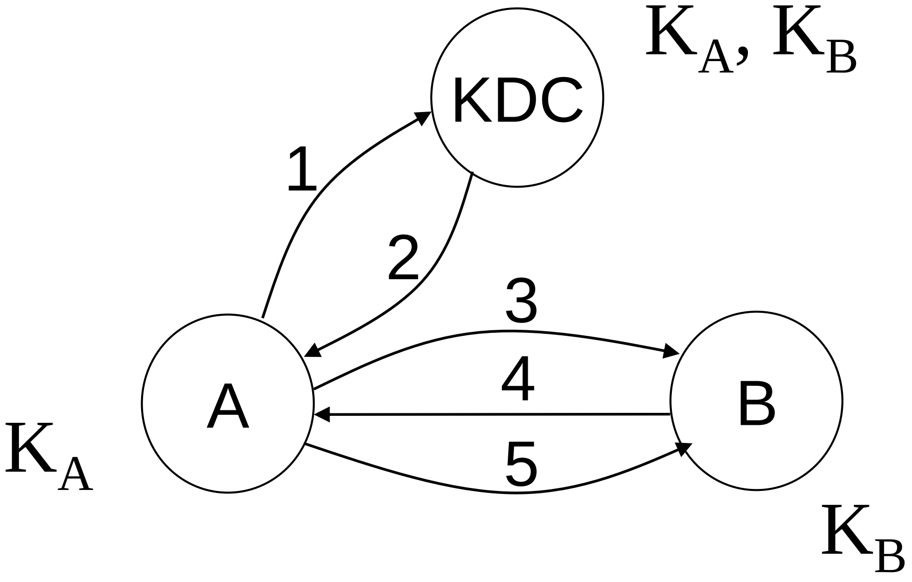
$A \rightarrow KDC$: A, B, N_A

$KDC \rightarrow A$: $E_{KA}(N_A, B, K_{AB}, E_{KB}(K_{AB}, A))$

$A \rightarrow B$: $E_{KB}(K_{AB}, A)$

$B \rightarrow A$: $E_{KAB}(N_B)$

$A \rightarrow B$: $E_{KAB}(N_B - 1)$



Needham-Schroeder Authentication Protocol

The first step is not protected, in the sense of not being confidential. This isn’t a problem in terms of the correct running of the protocol.

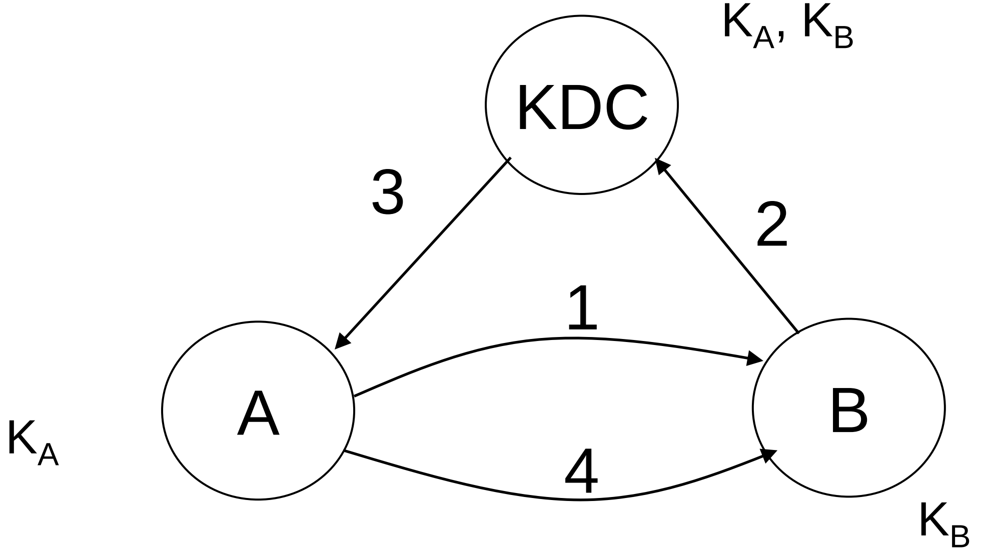
Step 3 is vulnerable to being replayed by an opponent who has determined an old session key K_{AB}. This assumes that Bob doesn’t necessarily record all session keys previously used.

A time-stamp can be used to avoid this kind of situation. Note that a time-stamp inside the step 3 encryption cannot be changed by an opponent who only knows K_{AB} for an old session.

There are other versions of the protocol.

Modified Needham-Schroeder (Neuman et al. 93)

1. A → B: A, N_A
2. B → KDC: $B, N_B, E_{K_B}(A, N_A, T_B)$
3. KDC → A: $E_{K_A}(B, N_A, K_{AB}, T_B)$, $E_{K_B}(A, K_{AB}, T_B), N_B$
4. A → B: $E_{K_B}(A, K_{AB}, T_B), E_{K_{AB}}(N_B)$



Could you suggest another approach to prevent the replay attack without using timestamp?

Modified NS Protocol

1. A → B: A
2. B → A: E_{Kbs} (A, N’)
3. A → S: A, B, N_a, E_{Kbs} (A, N’)
4. S → A: E_{Kas}(N_a, B, K, E_{Kbs}(K, A, N’-1))
5. A → B: E_{Kbs} (K, A, N’-1)
6. B → A: E_K(N_b)
7. A → B: E_K (N_b-1)

2.Kerberos Version 5

Kerberos Version 5

1: $C \rightarrow AS$: Options, ID_C, R_C, ID_{tgs}, Times, N_1

2: $AS \rightarrow C$: R_C, ID_C, $Ticket_{tgs}$,$E_{Kc}[K_{c,tgs}$,Times, N_1, R_{tgs}, $ID_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[$Flags, $K_{c,tgs}$, R_C, ID_C, AD_C, Times$]$

3: $C \rightarrow TGS$: Options, ID_V,Times,N_2,$Ticket_{tgs}$, $Auth_C$

$Auth_C = E_{K_{c,tgs}}[ID_C, R_C, TS_1]$

4: $TGS \rightarrow C$: R_C, ID_C, $Ticket_V$, $E_{K_{c,tgs}}[K_{C,V}$, Times, N_2, R_V, $ID_V]$

$Ticket_V = E_{K_V}[$Flags, $K_{c,v}$, R_C, ID_C, AD_C, Times$]$

5: $C \rightarrow V$: Options, $Ticket_V$, $Auth_C$

$Auth_C = E_{K_{c,v}}[ID_C, R_C, TS_2$, Subkey, Seq#$]$

6: $V \rightarrow C$: $E_{K_{c,v}}[TS_2$, Subkey, Seq#$]$

- Freshness:
  - Times indicated lifetimes.
  - Time-stamps avoid replay.
  - Nonce indicates the session and avoid replay.
  - In 5/6 the Sequence Number can be regarded as a nonce for the subsequent communications. The number will be increased every time.

- Confidentiality:

  1. Not encrypted
  2. Encrypted with K_c and K_{tgs}.
  3. Some encrypted with $K_{c,tgs}$.

     Options, ID_V,Times, Nonce2 are not encrypted.

  Steps 4, 5 and 6 have encryption.

- Authenticity:
  - In step 2 C knows the session key comes from AS, but there isn’t any authentication on the ticket part of this.
  - In step 3 the ticket contains the session key. TGS will accept the client if the authenticator is decrypted to the correct information. Note that the ticket is encrypted so that only TGS and AS could decrypt it.
  - In step 4 it is reasonable for C to believe that the session key, encrypted with $K_{c,tgs}$, comes from TGS. Again, there isn’t any authentication on the ticket part of this.
  - In step 5 the server V believes the sender is C provided the session key decrypts the authenticator appropriately.
  - In step 6 the client accepts the message from V, and now accepts that the ticket must have been valid, since only V should have access to the key in it.

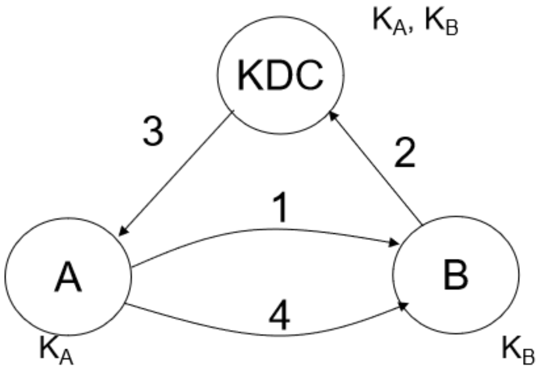
What advantages could a hybrid (mixed public-key and symmetric) Kerberos provide?

A hybrid Kerberos may provide some advantages.

- In step 1, a user password can be encrypted under the AS’ public-key to provide user authentication. Alternatively, a signature on the message (under the user’s public key) can also provide authenticity.
- In Steps 2 and 4 the tickets could be signed so the client believes they are issued by AS and TGS, respectively.
- In Steps 3 and 5 the client could sign the authenticator to enhance authenticity.

3. Protocol Analysis and Fix

3.Modified Needham-Schroeder (Neuman et al. 93)

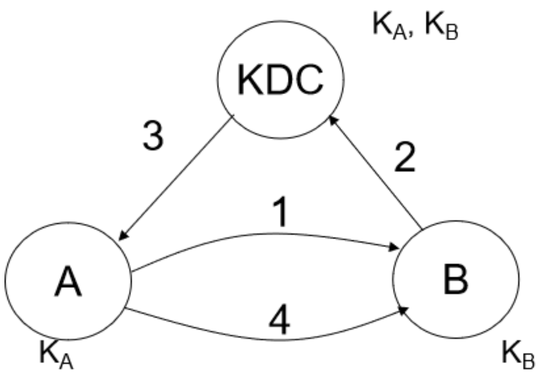


1. A → B: A, N_A
2. B → KDC: $B, N_B, E_{K_B}(\ A, N_A, T_B)$
3. KDC → A: $E_{K_A}(B, N_A, K_{AB}, T_B)\ ,\ E_{K_B}(A, K_{AB}, T_B), N_B$
4. A → B: $E_{K_B}(A, K_{AB}, T_B), E_{K_{AB}}(N_B)$

3.1 If KDC is curious, what could happen?

❖ The communications between A and B will be captured and known by KDC.

3.Modified Needham-Schroeder (Neuman et al. 93)

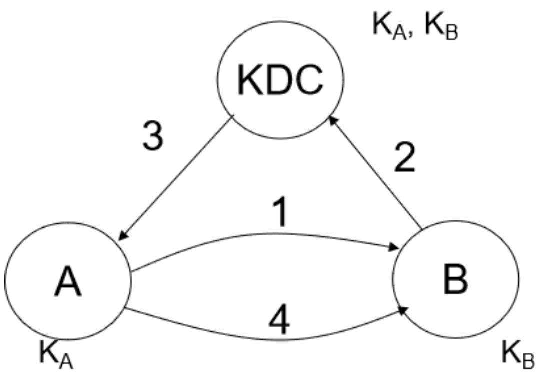


1. A → B: A, N_A
2. B → KDC: B, N_B, E_{K_B}(A, N_A, T_B)
3. KDC → A: E_{K_A}(B, N_A, K_{AB}, T_B) , E_{K_B}(A, K_{AB}, T_B), N_B
4. A → B: E_{K_B}(A, K_{AB}, T_B), $E_{K_{AB}}$(N_B)

3.2 If KDC cannot generate K_{AB} random and K_{AB} is always same for all request, what could happen?

❖ C and D could pretend to do centralized authentication protocol with KDC to get K_{AB} and know communications between A and B.

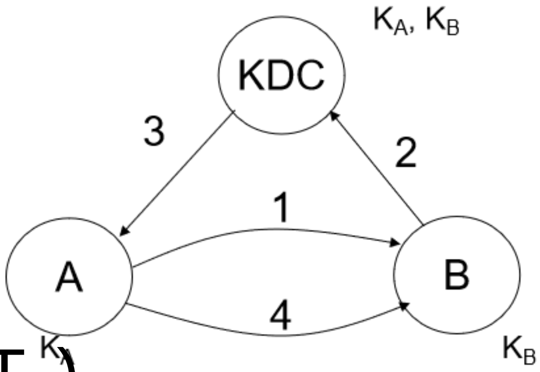
3.Modified Needham-Schroeder (Neuman et al. 93)



1. A → B: A, N_A
2. B → KDC: B, N_B, E_{K_B}(A, N_A, T_B)
3. KDC → A: E_{K_A}(B, N_A, K_{AB}, T_B) , E_{K_B}(A, K_{AB}, T_B), N_B
4. A → B: E_{K_B}(A, K_{AB}, T_B), $E_{K_{AB}}$(N_B)

3.3 If KDC cannot generate K_{AB} random and K_{AB} is always same for all request, how to fix this into secure?

3.Modified Needham-Schroeder (Neuman et al. 93)



1. $A \rightarrow B$: A, N_A
2. $B \rightarrow KDC$: $B, N_B, E_{K_B}(A, N_A, K_{AB}, T_B)$
3. $KDC \rightarrow A$: $E_{K_A}(B, N_A, K_{AB}, T_B)$, $E_{K_B}(A, T_B)$, N_B
4. $A \rightarrow B$: $E_{K_B}(A, T_B)$, $E_{K_{AB}}(N_B)$

Solution (above B chooses for KDC)

3.3 If KDC cannot generate K_{AB} random and K_{AB} is always same for all request, how to fix this to make the protocol secure?