A3 INF2310

Aslak Vik Sørvik and Aslak Røstad Bjordal

March 2023

1 Problem 1

Yes, A and B can reach the wrong conclusion with an attacker present. The given protocol is vulnerable to a simple replay attack as is. This attack could be something like this:

1. The attacker intercepts the first message between A and B obtaining {N_A}K.
2. The attacker can then replay this message at a later time, and by doing this pretend to be A.
3. If B receives this message, B will respond with a fresh nonce {N_B}K and will also include the N_A in the response.
4. The attacker can then intercept this message to obtain N_A unencrypted.
5. The attacker can then replay this message to A at a later time, and by doing this, pretending to be B.
6. When A receives the replayed message, it reponds with the nonce N_B.
7. The attacker can then intercept the third message from A to B and obtain N_B unencrypted.

By doing this, the attacker has convinced both A and B that they share the common key K, even though the attacker does not have the said key K.

2 Problem 2

The given protocol seems to be secure. To see how it would be safe, we can imagine an attacker who wants to trick A and B into using a different common key K_AB. The attacker has a possibility to intercept the messages between A and KDC, or B and KDC, and even modify the message. But the attacker is not able to break the encryption or generate new keys.

Now assume that the attacker intercepts the message from KDC to A and changes the the value of N to a new value N_new. For the attacker to be able to change the key it will have to generate a new pair of keys encrypted with K_A, but as the key K_A is considered secure, this is not possible without breaking or obtaining K_A. And since K_A is assumed secure, the attacker will not be able to change the message to A.

The same goes for communication between B and KDC. This means that as long as the key used between KDC and A/B (K_A/K_B) is secure, and virtually impossible to break, we see that the protocol is secure.

3 Problem 3

In this issue there is a timer that we have to get around. The objective is to try to send an old key: K_AB to B. In this case Old is older than 5 seconds. To do this we act as T and intercept the message after A has sent it to S. Then when S tries to send it to B, we intercept it and send it back posing as B. Then because of the way the message is defined, it will send the message back to A again. Then we intercept it on the other side acting as A. Once again we send it back to S which will make S send it to B once more. Then we can do this dance for as long as we would like. When we want to end we just let the message pass through to B and he will have an old version of K_AS.

In this illustration, the interceptor is T, and the symbol in the () is who T is acting as. There are also multiple T_S denoted by v1,v2. These are to separate different timers. T_Sv(x) is timer version X where x is the number of times this loop has been done.

1. $$A \rightarrow S : A, \{T\_A, B, K\_AB\}K\_AS$$

2. $$S \rightarrow T(B) : \{T\_Sv1, A, K\_AB\}K\_BS$$

3. $$T(B) \rightarrow S : B, \{T\_Sv1, A, K\_AB\}K\_BS$$

4. $$S \rightarrow T(A) : \{T\_Sv2, B, K\_AB\}K\_AS$$

5. $$T(A) \rightarrow S : A, \{T\_Sv2, B, K\_AB\}K\_BS$$

6. *On and on until you want the loop to end.*

7. $$S \rightarrow T(S) \rightarrow B : \{T\_Sv(x), A, K\_AB\}K\_BS$$