INF-2310 Assignment 3
Authentication Protocols

Magnus Dahl-Hansen, Sigurd Myrnes Uhre, Jørgen Eriksen Løken

March 2023

1 Problem 1

The following protocol is designed to prove to A and B that they share a key K with each other.

```
A --> B:  {N_A}K        where N_A is a fresh nonce
B --> A:  {N_B}K, N_A   where N_B is a fresh nonce
A --> B:  N_B
```

Figure 1:

Can A and/or B reach the wrong conclusion if attackers are present? If so, give the attack; otherwise, give an informal explanation of the protocol's correctness.

An assumption for our answer is that the encryption techniques are symmetric since there is use of nonce.

1.1 Answer: Nonces and replay attacks

The use of nonces are a way to prove the originality of a message. A nonce is a random or semi-random number used only once for each message sent in a cryptography communication, to defend against replay attacks in such a way that if a message is received with the same nonce more than once, the additional messages will be discarded as invalid.

All the messages in figure 1 are protected against replay attacks since the nonces: N_A and N_B are fresh nonces for the receiving parts.

1.2 Answer: Man in the middle attack (MITMA)

One type of attacks that nonces alone don't defend against are man-in-the-middle-attacks. As explained above, the nonce only checks for the messages originality and not whether it has been tampered with or viewed by an MITMA-agent.

Our perception of the protocol is that A encrypts a nonce (N_A) and sends it to B. B will then receive and decrypt this nonce and send it back to A along with its own encrypted nonce (N_B). If B's result of the decrypted nonce (N_A) matches with the original nonce of A then A will know that it shares the same key with B. To let B know that it also shares the same key, A will decrypt B's nonce (N_B) and send it back so that B knows that is share the same key with A.

The problem with this protocol is that the decrypted nonces (N_A and N_B) is sent back without any decryption, which in turn means that a third part C can pick up this result, modify it and forward it to A and B. This way both A and B will receive wrong results of the decryption and believe that they do not share the same key.

To summarise: In the first message, the nonce is encrypted with a key and the MITMA-agent have no way of interfering. In message two and three however, the confirming nonce (not the encrypted nonce) is vulnerable for modification by an MITMA-agent. The agent can modify the decrypted nonce and make it so that the A and B reach the conclusion that they don't have the same key, while in reality, they actually have the same key.

2 Problem 2

Does this protocol provide A and B with a shared key K_AB that they can trust for their secret communications? If not, describe the attack; otherwise, give an informal explanation of the protocol's correctness.

```
1.  A --> KDC: A,B,N

2.  KDC --> A: {N,A,B}K_A,  {N,K_AB}K_A

3.  KDC --> B: {N,A,B}K_B,  {N,K_AB}K_B

4.  A --> B:  {N+1}K_AB   compare value N+1 with contents of message 3

5.  B --> A:  {N-1}K_AB   compare value N-1 with contents of message 2
```

Figure 2: Here KDC is a Key Distribution Center, and K-A denotes a key shared between A and the KDC. Key K-AB is shared between A and B.

2.1 Answer:

The first message exposes the variables A,B and N to MITM-agents and the possibility for the KDC to receive an altered value N (which we assume is a nonce). The KDC sends the variables and the shared key of A and B (K_AB) encrypted and secure to both A and B. Our understanding of this exchange, is that both A and B can decrypt this K_AB key and use is safely in interactions between A and B.

However, the N variable that the KDC sends out which is used for comparison, may have been tampered with in the first stage. In return, the KDC will send the tampered nonce to A and B for them to use. One could think that sending a tampered nonce to A and B is not secure, but in reality it does not matter what the attacker switched the nonce to in the first stage, because when both A and B sends messages to each other in stage 4 and 5, they both use a fresh nonce with either $N+1$ or N-1. Overall, this means that the "originality" of the messages is between A and B is unique because the nonce is different. This also protects against replay attacks, when the nonce will never be the same because of stage 4 and 5.

We assume that the KDC and B shares a key KDC_B. This protocol will provide A and B with a reliable shared key (Key_AB), because KDC provides this key to both parties as an encrypted message that can only be decrypted between KDC-¿B and only between KDC-¿A. However, A and B must blindly trusts that KDC is a reliable key distribution source which will not modify any messages between A and B because KDC also knows the shared key between A and B. There is also important to know that even if KDC provides a reliable key between A and B, there may be no way for A and B to know that the key they share are secure. This is because we assume that the variable N that A sends to KDC is a variable used to prove that A and B shares the same key. This N variable is sent from A to KDC without any encryption and may therefore be picked up in a MITM attack and modified, but in reality this does not matter because of the freshly created nonces with the interaction between A and B, meaning that the originality of the messages is contained even though A's message to the KDC was tampered with. Therefore, the protocol provides A and B with a shared key K-AB that they can trust for their secret communcations.

3 Problem 3

The following protocol allows a client A to suggest a shared key K-AB for communicating with B rather than depending on a server to generate that key. So server S is used in the protocol only as a secure communications channel from A to B, because:

K-AS is shared between A and S
K-BS is shared between B and S

```
A --> S:  A, {T_A, B, K_AB}K_AS   where T_A is current time
S --> B:  {T_S, A, K_AB}K_BS      where T_S is current time
```

Figure 3:

Assume that principals A, B, and S ignore any message they receive that contains a timestamp that is more than 5 seconds old.

3.1 Discussion and solution

There are some aspects about this problem that we need to discuss before providing our solution. We assume that the time stamp used in the protocol is a unique time-stamp (unix time) and that it is not possible to resend the same message on the next day at the same time.

This protocol is flawed and an intruder T can force B to adopt an instance of K_AB that is arbitrarily old. This attack is based on two essential weaknesses in the protocol: 1. The message does not include a nonce which verifies the originality of the message. 2. the server S updates the timestamp each time a message is transported through the server.

In this protocol a MITM-attack can occur if an intruder T intercepts a message from server S to B, copy it, and send it back to the server S, imitating B wanting to send a message to A. B still receives this message as normally. If T intercepts and replies to S within 5 second's, the time-stamp will be valid in the server S. Server S will now update this timestamp and send a message to A. If the intruder T now again intercepts this message from S to A, and again replay it back to S, the server S will assume that A wants to establish connection with B. This involves verifying the message and once again updating the time-stamp.

The intruder will again intercept the the message from S to B, but will now not forward it to B, but send it back to S again and restart the loop. The intruder T have now established a loop in-between A and B which goes back and fourth between the server, updating the time-stamp at every interception with S and not letting the message end up at either A or B.

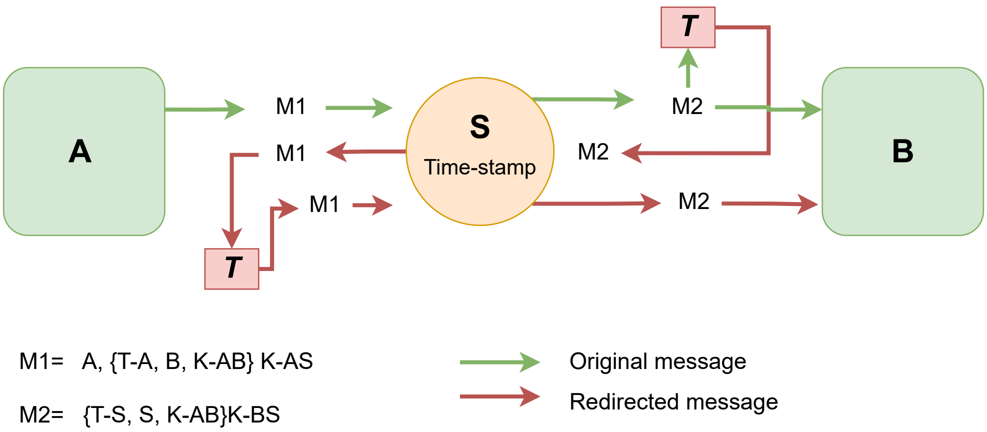

Figure 4: Illustration on how the Dolov-Yao attacker (T) intercept and copies the message before sending it back to the server S.

Lets say that A sends a message to the server S: M1 (A, *[T_A, B, K_AB]K_AS*), where the time-stamp *T_A* is at 12:00 PM. The message contains the key *K_AB*. The server validates the timestamp and forwards the *K_AB* key to B in a new message: M2 (*[T_S, A, K_AB]K_BS*.) B receives this message and can use the *K_AB* key. The message from server S to B is however intercepted by an intruder T that copies the message *M2* and sends it back to the server within 5 second's. The server receives the *M2* message, approves the timestamp, and forwards a message *M1*, with destination A, which is a request that A and B establish communication through S. T intervenes and "takes" the entire message, so that A does not receive it. T sends the message back to S, And S interprets this message as normal, validates it and again updates the timestamp before creating a message *M2* with the destination of B.

The intruder can keep this loop active for as long as it wants since the timestamp is updated continuously at the server S. If the intruder T can at a later time, lets say 19:00 PM, decide to stop the loop and let the message go trough to B, it will cause a replay attack. B will now receive a message with valid time-stamp containing a key *K_AB* that was created at 12:00 PM.

4 References

1. Cryptographic nonce [Internet]. Wikipedia. Wikimedia Foundation; 2023 [cited 2023Mar7]. Available from: https://en.wikipedia.org/wiki/Cryptographic-nonce