Data Security, Assignment 1, Group 23

Mukhtiar Khizar (s182696)
Sarker Tama (s232913)
Turki Yassine (s231735)
Zdrale Dimitrije (s231734)

October 2023

[1]

[1] All the relevant code is included in the Appendix of this document.

1 Exercise 1: AMP

1.1 Describe the attack found by OFMC: what does the attacker do and what goal is broken here?

The attack trace found by OFMC shows that the communication between A and B in this protocol is intercepted by an intruder who impersonates B, thus, performing a man in the middle attack. Here is an overview of the attack :

- A initially contacts the intruder, thinking it is B, and uses the intruder's public key to encrypt its own identity, the intruder's identity, the request, and a shared symmetric key.
- The intruder decrypts A's request, alters the request (excludes itself, adds B's identity, creates a new request number and symmetric key), and forwards it to B using B's public key.
- B assumes the message is from A because it includes A's identity, and prompts A to authenticate itself through a trusted third party (s) using a shared password.
- B uses the symmetric key sent by the intruder to encrypt identities of A, B, s, and a request ID, sending it to the intruder.
- The intruder replaces B's identity and request ID, encrypts it with A's earlier sent symmetric key, and forwards it to A.
- A communicates again with the intruder, thinking this time that the intruder is s. The intruder receives the signed message from the trusted third party by impersonating A and sends it to B.
- B assumes that A was successfully authenticated and sends confidential data to the intruder, thinking it's A.

In conclusion, the attack is a result of weak authentication. and the goal that is broken is *B authenticates A on Request*.

1.2 How could this attack be prevented? Try to find a fix by extending some messages with more information. You should not introduce more crypto- graphic operations.

This attack can be avoided if A is more precise in its message to the trusted third party about who it wants to prove its identity to. Therefore, in our case, we add B to the request that A sends to the server, and the server adds B to his message to B as well (as shown below):

```
A->s:  {| A,s,B |}pw(A,s)
s->B:  { {A,s,B}inv(pk(s)) }pk(B)
```

This prevents the intruder from using the signed message because B can tell that the message was not intended for it. However, we can now face the possibility of a replay assault since the intruder can just take this message and ignore authentication. Thus, in order to fix this, we use a ReqID Nonce in order to make sure that the messages are always recent and not generated by a replay attack.

```
A->B: {A,B,Request,K}pk(B)
B->A: {|A,s,B,ReqID|}K

A->s:  {|A,s, ReqID, B|}pw(A,s)
s->B:  {{A,s,B,ReqID}inv(pk(s))}pk(B)
```

1.3 Suppose now pw(A, s) is not a strong cryptographic key, but a poorly chosen password. The specification contains the commented-out goal pw(A,s) guessable secret between A,s Explain the attack that OFMC finds on this goal and explain what is the problem.

Since the password shared between A and s is not cryptographically strong enough, OFMC finds that it is easily guessed by the intruder and, therefore, all the communication between A and s is compromised. The password is no longer a shared secret between A and s, and the intruder can easily get access to the message A wants to send to s by impersonating s and use that info to authenticate as A to B.

1.4 Try to fix also this second problem, i.e., that even in presence of the password the protocol works correctly. Here you have to change a bit the use of cryptography.

Any message that A transmits initially can potentially become known to the intruder. However, to communicate with s, A could encrypt an already weakly encrypted message with the public key of s. Furthermore, the message can include a nonce generated by A and a request ID for this specific session, so even if the intruder gets this message and impersonates A, s can determine that the request ID is from a previous session and the communicating entity is an intruder.

```
A->s:  { {| A,s,B,ReqID,Nonce |}pw(A,s)}pk(s)
s->B:  { {A,s,B,ReqID }inv(pk(s)) }pk(B)
```

Moreover, another possible way to prevent the attack would require A to generate another symmetric key ($K1$) and encrypt the message (that is already encrypted with a weak password) with this key. This symmetric key and the encrypted message are encrypted again with the public key of the server s. Thus, only s can decrypt this entire message from A using its private key and gain access to the symmetric key.

```
A->s:  { {| {| A,s,B,ReqID |}pw(A,s) |}K1, K1 }pk(s)
s->B:  { {A,s,B,ReqID }inv(pk(s)) }pk(B)
```

1.5 Suppose one wants to implement the single-sign-on with a TLS channel from exercise 1 to secure the connection between A and B. Describe how that relates to the current formulation of the protocol, in particular if that has any advantages/disadvantages. You do not need to make any experiments in OFMC (the problem might be too complex when combining the two protocols).

The very first step in this protocol requires that both communicating entities support TLS. Therefore, when A tries to communicate with B, the first exchange of messages between these two entities would comprise the former requesting a TLS session to be established. If the latter does not provide a relevant response, which requires B to send its valid digital certificate upon receiving a half symmetric key, A can suspect an impersonator and terminate the connection. This initial step is called the "hand-shake" and it is during this phase that a unique symmetric session key is generated by A upon receiving the other half key from B and vice-versa. Thus, all the data during the entire session is encrypted using this key. Furthermore, before the transmission of any data, A will authenticate itself to B with a single sign-on using the third party server s, which is the trusted identity provider of the former. The potential drawback with TLS is verifying the actual identity of A, since the latter does not have a digital certificate that it can authenticate itself on. However, the TLS channel itself is secure (even more secure in this case) enough to provide communication that is not compromised between the two entities, thus, preventing an intruder to perform an attack when the session is established. Moreover, for any "witness" X, with TLS the "request" service provider can have a secure communication channel to transmit and receive data from the former.

2 Exercise 2: Selfie.AnB

2.1 Try to describe and explain the protocol: explain the shape of the initial key, what the new key is, what the purpose of the nonces and MACs are, and why the goals are meaningful.

In the protocol, A generates a nonce and sends it to B, who appends a new nonce and uses a key derivation function to create a new key. B also generates a MAC for the nonces using the new key and sends it, along with the nonce, back to A. A can verify B's authenticity by following similar steps. This process ensures the shared initial key is used to create a new key and MACs for nonces, allowing both entities to authenticate each other and ensure message integrity.

2.2 OFMC returns an attack on Selfie.AnB. Describe the attack: what hap- pens, what is each agent "thinking" what happens, why does this violate the goals? Essentially: what went wrong here?

In the initial session, the intruder pretends to be B and obtains a nonce from A. Within the same session, the intruder then returns the nonce to A. In the subsequent session, A assumes that it is communicating with B and believes this to be the first message it sent. Consequently, A generates another nonce and shares it with the intruder, along with the hashed MAC of the newly generated key and the nonces.

The issue here is that the protocol breaches weak authentication because, in any given session, one of the entities is completely absent from the communication. This breach is denoted in the statement as "if there is a request fact without a matching witness fact," and it becomes apparent in the attack trace when A successfully completes the protocol without involving B. [Reference: pg. 10 of the OFMC documentation]

2.3 Fix the protocol and verify the fixed version for two sessions with OFMC.

In the initial exchange of messages in the communication, the nonce is not encrypted and it is vulnerable upon transmission on an insecure channel. Moreover, B doesn't know about the identity of A in the first transmission of the message. A could include its identity along with the nonce in a message and encrypt it using the shared secret key between A and B which is: exp(exp(g,secretk(A)),secretk(B)).

Thus, we change:

```
A->B: N1
```

to

```
A->B: {|N1,A|}exp(exp(g,secretk(A)),secretk(B))
```

2.4 Suppose that after the agents have executed the protocol, the intruder would find out secretk(A) (for some honest agent A $\neq$ i). Explain why this would break the secrecy goal of the protocol. Is it possible to modify the protocol so that secrecy would still hold as long as secretk(A) is only discovered after the execution of the protocol? Note that you cannot check this in AnB/OFMC because of the "after" restriction.

With the secret key of A, the intruder can generate the secret key shared by only A and B. So altering the protocol from the previous step would serve no purpose because intruder can access any data in transit between A and B that is encrypted with the now exposed key.

The solution to this issue is that A and B should not send their nonces (N_1, N_2) in raw form. Instead, they can send them as half-keys; exp(g, N_i). Now, even if the intruder gets the half-keys, getting N_i would be computationally hard for them. Moreover, since $exp(exp(g, N_1), N_2) = exp(exp(g, N_2), N_1)$ A and B can use this as an argument in the kdf to generate a new shared secret key. This new key is not affected even when secretk(A) is compromised.

3 Appendix

In this section, we include our final code for both exercises:

3.1 AMP-given.AnB : Solution 1 with a Nonce

```
Protocol: AMP

Types: Agent A,s,B;
       Number Request,ReqID,Data,Nonce;
       Symmetric_key K;
       Function pk,pw;

Knowledge:
       A: A,s,pw(A,s),pk(s),B,pk(B);
       B: B,pk(B),inv(pk(B)),s,pk(s);
       s: s,pk(s),inv(pk(s)),B,pk(B),A,pw(A,s);
       where B!=s

Actions:

A->B: {A,B,Request,K}pk(B)
B->A: {|A,s,B,ReqID|}K

A->s:  { {| A,s,B,ReqID,Nonce |}pw(A,s)}pk(s)
s->B:  { {A,s,B,ReqID }inv(pk(s)) }pk(B)

B->A: {|Request,Data|}K

Goals:

B authenticates A on Request
A authenticates B on Data
Data secret between B,A
pw(A,s) guessable secret between A,s
```

3.2 AMP-given.AnB : Solution 2 with a new additional Symmetric Key generated by A

```
Protocol: AMP

Types: Agent A,s,B;
       Number Request,ReqID,Data;
       Symmetric_key K,K1;
       Function pk,pw;

Knowledge:
       A: A,s,pw(A,s),pk(s),B,pk(B);
       B: B,pk(B),inv(pk(B)),s,pk(s);
       s: s,pk(s),inv(pk(s)),B,pk(B),A,pw(A,s);
       where B!=s

Actions:

A->B: {A,B,Request,K}pk(B)
B->A: {|A,s,B,ReqID|}K

A->s:  { {| {| A,s,B,ReqID |}pw(A,s) |}K1, K1 }pk(s)
s->B:  { {A,s,B,ReqID }inv(pk(s)) }pk(B)

B->A: {|Request,Data|}K

Goals:

B authenticates A on Request
A authenticates B on Data
Data secret between B,A
pw(A,s) guessable secret between A,s
```

3.3 Selfie.AnB

```
Protocol: Selfie


Types: Agent A,B;
       Number N1,N2,outer;
       Function secretk,mac,kdf

Knowledge: A: A,B,exp(g,secretk(B)),g,secretk(A),mac,kdf,outer;
     B: A,B,exp(g,secretk(A)),g,secretk(B),mac,kdf,outer;

Actions:

A->B: {|N1,A|}exp(exp(g,secretk(A)),secretk(B))
B->A: N2, mac(kdf(exp(exp(g,secretk(A)),secretk(B)),N1,N2),
               N1,N2)
A->B: A,B,    mac(kdf(exp(exp(g,secretk(A)),secretk(B)),N1,N2),outer,
               N1,N2,mac(kdf(exp(exp(g,secretk(A)),secretk(B)),N1,N2),N1,N2))
Goals:

B authenticates A on N1
A authenticates B on N2
kdf(exp(exp(g,secretk(A)),secretk(B)),N1,N2) secret between A,B
```