Protocol Security Lab

Authors

Joao Mena - s223186
Tomas Estacio - s223187
Aidana Nursultanova - s212994
Renjue Sun - s181294

October 12, 2022

Contents

1 Exercise 1: Selfie

1.1 Question 1

In this question we are asked to describe and explain the protocol Selfie.AnB. The initial key is $exp(exp(g, secretk(A)), secretk(B))$, which was generated before the execution of the protocol. This key was generated using the Diffie-Hellman algorithm between agent A and agent B, using a private secret of A and a private secret of B (their private keys) to build a shared secret of A and B which is believed to be hard to break. The Diffie-Hellman method starts by agent A creating half of the key $exp(g, secretk(A))$ authenticated with the digital signature of A to ensure that agent B knows they are really communicating with A. This half of the key is created using the cyclic group algorithm, in which we use a generator g and the private key of A to build $g^{secretk(A)}$, which is considered hard to compute without the private key of A. For the second half of the key, agent B receives the message, decrypts it, and sends back their own half of the key, generated the same way and resulting in $g^{secretk(B)}$, with their digital signature to ensure authenticity. At this point, A has received $g^{secretk(B)}$ and B has received $g^{secretk(A)}$ which means they are able to build their shared key: for A $(g^{secretk(B)})^{secretk(A)}$, and for B $(g^{secretk(A)})^{secretk(B)}$. Both resulting keys are equal and can be used.

The key that is being built in the new protocol is a new secret shared between A and B using a key derivation function with the secret shared key and the two Nonces used as inputs. The Nonces are used in the protocol for authentication purposes, mainly to ensure that old communications can not be used in replay attacks. These are created by the agents A and B, respectively. The integrity of the message is also assured by the use of a Message Authentication Code which uses a cryptographic hash function and the new shared secret key generated with the KDF as input and outputs a code that can't be reversed.

The goals of the Selfie protocol are for A and B to authenthicate each other on their respective Nonces. These goals are important because it is necessary to properly authenticate the agents in order to ensure a secure communication between them.

A representation of this protocol can be found in Figure 1 for further clarification.

```
Protocol: Selfie

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

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

Actions:

A->B: N1
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),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
```

Figure 1: Selfie.AnB Protocol

1.2 Question 2

When testing this protocol with the OFMC protocol analyzer, there is an Attack Found. The Attack Trace is the following:

```
ATTACK TRACE:
(x501,1) -> i: N1(1)
i -> (x501,1): N1(1)
(x501,1) -> i: N2(2),mac(kdf(exp(exp(g,secretk(x502)),secretk(x501)),N1(1),N2(2)),N1(1),N2(2))
i -> (x501,1): N2(2),mac(kdf(exp(exp(g,secretk(x501)),secretk(x502)),N1(1),N2(2)),N1(1),N2(2))
(x501,1) -> i:
x501,x502,mac(kdf(exp(exp(g,secretk(x501)),secretk(x502)),N1(1),N2(2)),wa,N1(1),N2(2),mac(kdf(exp(exp(g,secretk(x501)),
secretk(x502)),N1(1),N2(2)),N1(1),N2(2)))
```

Figure 2: Selfie.AnB Attack Trace by OFMC

Breaking down this attack found by OFMC, we see that the Intruder agent starts by intercepting the first Nonce N1 sent by agent x501, and sending it back to agent x501. Now, agent x501 is receiving a Nonce, that is still N1, but now thinks their role is of a B in this protocol, so they send another Nonce N2 to the Intruder, alongside of the encoded shared key generated to communicate with x502. After this, the Intruder sends the same message, N2 and the encoded shared key to x501, making x501 think they are once again in the role of agent A in the protocol, and replying as such, sending their name, x502's name, and the encoded shared keys. At this point, the Intruder has managed to authenticate on x501

pretending to be agent x502 without knowing any of the private keys.

This attack is possible due to a Weak Authentication mechanism, meaning that the Intruder does not actually need to know any of the private keys of either agent. Since the messages do not specify the sender nor the receiver, the Intruder can just intercept the communications and create their own messages to authenticate. This attack is called Man In The Middle.

1.3 Question 3

To fix the issue previously mentioned,the protocol was modified in order to strenghten the authenticity: the names of A and B were added to the first two messages exchanged, and alongside with the respective Nonces, encrypted using their shared secret as a key.

Figure 3 illustrates how Selfie.AnB can be fixed with no attack found by the OFMC.

```
1  Protocol: Selfie
2
3  Types: Agent A,B;
4         Number N1,N2;
5         Function secretk,mac,kdf
6
7  Knowledge: A: A,B,exp(exp(g,secretk(A)),secretk(B)),mac,kdf;
8        B: A,B,exp(exp(g,secretk(A)),secretk(B)),mac,kdf;
9
10 Actions:
11
12 A->B: {| A, exp(g,N1) |}exp(exp(g,secretk(A)),secretk(B))
13 B->A: {| B, exp(g,N2) |}exp(exp(g,secretk(A)),secretk(B)),mac(kdf(exp(exp(g,
       secretk(A)),secretk(B)),exp(exp(g,N1),N2),exp(g,N1),exp(g,N2))
14 A->B: mac(kdf(exp(exp(g,secretk(A)),secretk(B)),exp(exp(g,N2),N1)),exp(g,N1)
       ,exp(g,N2),mac(kdf(exp(exp(g,secretk(A)),secretk(B)),exp(exp(g,N1),N2))),
       exp(g,N1),exp(g,N2)))
15
16 Goals:
17
18 B authenticates A on N1
19 A authenticates B on N2
20 kdf(exp(exp(g,secretk(A)),secretk(B)),exp(exp(g,N1),N2))) secret between A,B
```

Figure 3: Fixed Selfie.AnB with No Attack Found

1.4 Question 4

Assuming now that secretk(A) is no longer actually a secret, the Intruder would now be able to use the shared key between A and B and decrypt the messages we they trying to send in our protocol to authenticate both agents, corrupting the secrecy of the Nonces used. In this case, if we tried to use this protocol now, a MITM attack would allow the Intruder to extract information that we need to remain a secret.

However, it is still possible to ensure that our secrecy goals are accomplished by changing the messages sent between the agents A and B, because instead of sharing the Nonces N1 and N2, we can use the Diffie-Hellman algorithm to replace every use of the Nonces to g^{N1} in case of agent A and g^{N2} in case of agent B. That way we can still create a key that contains a shared secret between A and B, because N1 is still a secret only known by A and N2 is still a secret only known by B. This means that agent A can use the key $(g^{N2})^{N1}$ and agent B can use the key $(g^{N1})^{N2}$ to create the new shared key between them because $(g^{N2})^{N1} = (g^{N1})^{N2}$. The resulting protocol would have the following structure:

```
1  Protocol: Selfie
2
3  Types: Agent A,B;
4         Number N1,N2;
5         Function secretk,mac,kdf
6
7  Knowledge: A: A,B,exp(exp(g,secretk(A)),secretk(B)),mac,kdf;
8             B: A,B,exp(exp(g,secretk(A)),secretk(B)),mac,kdf;
9
10 Actions:
11
12 A->B: {| A,exp(g,N1) |}exp(exp(g,secretk(A)),secretk(B))
13 B->A: {| B,exp(g,N2) |}exp(exp(g,secretk(A)),secretk(B)),mac(kdf(exp(exp(g,
      secretk(A)),secretk(B)),exp(exp(g,N1),N2),exp(g,N1),exp(g,N2))
14 A->B: mac(kdf(exp(exp(g,secretk(A)),secretk(B)),exp(exp(g,N2),N1)),exp(g,N1)
      ,exp(g,N2),mac(kdf(exp(exp(g,secretk(A)),secretk(B)),exp(exp(g,N1),N2))),
      exp(g,N1),exp(g,N2)))
15
16 Goals:
17
18 B authenticates A on N1
19 A authenticates B on N2
20 kdf(exp(exp(g,secretk(A)),secretk(B)), exp(exp(g, N1), N2))) secret between
      A,B
```

Figure 4: Modified Selfie.AnB for leaked secretk(A)

2 Exercise 2: Calling Home

2.1 Question 1

In this section we are asked to describe the protocol call-home.AnB. By analyzing the *Actions* section, we see that A communicates with *home* using B as a communication medium. Home communicates with A through B as well. The protocol here is trying to achieve a secure communication between A and B, using agent *home* as a way to authenticate both of them and create a shared secret key using the Diffie-Hellman protocol to generate it. Below explains *Actions* step by step according to the initial protocol file:

1. A -> B: In this step we can see, that A tells B that she would like to talk with him and sends him half of the key g^X generated using Diffie-Hellman algorithm. She also sends him message m1 intended for *home* by encoding it using MAC function, where $pw(A, home)$ is a key shared between only A and *home*. Therefore, B cannot read the message m1 and takes the mediator role. Here, the MAC function is used to maintain information integrity, so that *home* can safely assume that the data integrity of the message is intact.

2. B -> *home*: B sends *home* the message generated by A and all data sent by A. He also sends his key g^Y and message m2 to *home* by encoding it using MAC function, where $pw(B, home)$ is a key shared only between B and *home*. All data sending in this step are as parameters so that *home* can be sure that data is unchanged.

3. *home* -> B: *home* sends B message m3 intended for A, encoding it using MAC function, where $pw(A, home)$ is a key shared only between A and *home*, and both g^X and g^Y are inputs. B also sends message m4 to *home*, enconding it with MAC function.

4. B -> A: B sends A his key g^Y, the message m3 from *home*, encoded with MAC, and message m5 also encoded using MAC function, where $(g^X)^Y$ is a key between A and B, generated using the Diffie-Hellman algorithm. Other input parameters of the function are data of this step, so that A can be sure that data is unchanged.

5. A -> B: In this step, A sends B secret M, by encrypting it with symmetric key shared between A and B.

2.2 Question 2

OFMC found the attack with type 'Weak Authentication'. The attack is caused by wrong sender – the intruder 'i' who can actually take the role of A to send message M to B, by making B believe it is A who sent the message, so we get a violation of the strong authentication aspect of the goal for this protocol. This attack is a form of multiple replay attack. The attack trace is the following:

```
ATTACK TRACE:
(x802,1) -> i: x802,x801,exp(g,X(1)),mac(pw(x802,home),m1,x802,x801,exp(g,X(1)))
i -> (x801,1): x802,x801,g,x306
(x801,1) -> i: x802,x801,g,x306,x801,g,exp(g,Y(2)),mac(pw(x801,home),m2,x802,x801,g,x306,x801,g,exp(g,Y(2)))
i -> (x801,2): x802,x801,exp(g,X(1)),mac(pw(x802,home),m1,x802,x801,exp(g,X(1)))
(x801,2) -> i:
x802,x801,exp(g,X(1)),mac(pw(x802,home),m1,x802,x801,exp(g,X(1))),x801,exp(g,X(1)),exp(g,Y(3)),mac(pw(x801,home),
m2,x802,x801,exp(g,X(1)),mac(pw(x802,home),m1,x802,x801,exp(g,X(1))),x801,exp(g,X(1)),exp(g,Y(3)))
i -> (home,1):
x802,x801,exp(g,X(1)),mac(pw(x802,home),m1,x802,x801,exp(g,X(1))),x801,exp(g,X(1)),exp(g,Y(3)),mac(pw(x801,home),
m2,x802,x801,exp(g,X(1)),mac(pw(x802,home),m1,x802,x801,exp(g,X(1))),x801,exp(g,X(1)),exp(g,Y(3)))
(home,1) -> i:
x801,x802,mac(pw(x802,home),m3,x801,exp(g,X(1)),exp(g,Y(3))),mac(pw(x801,home),m4,x801,x802,mac(pw(x802,home),m3,
x801,exp(g,X(1)),exp(g,Y(3))))
i -> (x801,1):
x801,x802,mac(pw(x802,home),m3,x801,exp(g,X(1)),exp(g,Y(3))),mac(pw(x801,home),m4,x801,x802,mac(pw(x802,home),m3,
x801,exp(g,X(1)),exp(g,Y(3))))
(x801,1) -> i:
x801,x802,exp(g,Y(2)),mac(pw(x802,home),m3,x801,exp(g,X(1)),exp(g,Y(3))),mac(exp(g,Y(2)),m5,x801,x802,exp(g,Y(2))
,mac(pw(x802,home),m3,x801,exp(g,X(1)),exp(g,Y(3))))
i -> (x801,1): {|x709|}_(exp(g,Y(2)))
```

Figure 5: CallHome.AnB Attack Trace by OFMC

Below explains the attack step by step according to the attack diagram automatically generated by OFMC GUI. Succinctly, the attack works like this:

1. (A,1) -> i: intruder i starts by intercepting the message sent by agent A to agent B.
2. i -> (B,1): i modifies the key g^X to g (can be either an arbitrary number or the result of g^X), and sends it to B in the role of A.
3. (B,1) -> i: i intercepts the message sent by agent B to *home*. It is important to note here, that B sends his half of the key g^{Y2} in this session 1.
4. i -> (B,2): i creates a new session 2 and sends B the initial message of A he got at Step 1.
5. (B,2) -> i: i intercepts the message sent by B to *home* in session 2. In this session, B sends his key g^{Y3}.
6. i -> (home,1): i impersonates B and sends the message received in Step 5 to *home*
7. (home,1) -> i: i intercepts the message sent by *home* to B.
8. i -> (B,1): i impersonates *home* and sends the message received in Step 7 to B.
9. (B,1) -> i: i intercepts the message sent by B to A. This step is executed in session 1, where B believes that the key of A is g. B creates a new shared key $exp(exp(g, X), Y2)$, where $exp(g, X)$ is g. Therefore, a new shared key between A and B will be $exp(g, Y2)$. Now intruder can read encrypted messages between A and B.
   Intruder can not send this step to A, because it contains the message encoded by MAC function: $mac(pw(A, home), m3, B, g^X, g^{Y3})$. Since he does not know the key between *home* and A, it is not possible for him to change the content, and one of the parameters of this MAC function is g^{Y3} - the key of B generated in session 2. Therefore, A will

not be able to read the message, because she believes that the key of B is g^{Y2}, and will reveal the attack.

10. i -> (B,1): i impersonates A and sends B a secret by encrypting it with the key generated in Step 9.

2.3 Question 3

To fix the issue previously mentioned, we made changes to the message from *home* to B. The protocol was modified in order to strengthen the authenticity: the key of B $exp(g, Y)$ was added as a parameter of MAC function authenticating the message m4 from *home* to B. Since one of the parameters of this MAC function is a secret key between *home* and B $pw(B, home)$, it is not possible for the intruder to change the message. Only agent B and *home* can read this MAC function. This step runs in session 1, which means that the key of B is $exp(g, Y2)$. But with our fix, *home* sends the key of B he knows as $exp(g, Y3)$. Therefore, B will not be able to read the message, because he believes his key is $exp(g, Y2)$, and will reveal the attack.

Figure 6 illustrates how CallHome.AnB can be fixed with no attack found by the OFMC.

```
19 home ->B:    B,A,mac(pw(A,home),m3,B,exp(g,X),exp(g,Y)),
20 mac(pw(B,home), exp(g,Y), m4,B,A,mac(pw(A,home),m3,B,exp(g,X),exp(g,Y)))
```

Figure 6: Fixed CallHome.AnB with No Attack Found

It is also a good idea to add timestamps in the protocol used, so that we can detect if the intruder attempts to perform any replay attack, which he does on the attack found.

2.4 Question 4

After changing $pw(A, home)$ from a strong cryptographic key to a poorly chosen password which is quite guessable, OFMC finds the attack with goal 'guesswhat', which is called a guessing attack. This happens because now the protocol relies on a potentially low-entropy secret, the password shared between A and home.

Following the attack trace, if intruder i intercepts the message A intends to send to B in Step 1 of Actions explained in 2.1 Question 1, he can try brute force attack until having the correct password for calculating the same MAC() as A sends out. This is because the intruder i has access to A, B, the half key of A g^X and m1 and the guessable password is the only thing lacking for reproducing the MAC().

Once the intruder knows the password, he can easily impersonate A to communicate with B. Then the protocol confidentiality is broken and the secrecy goal has been violated. The protocol confidentiality of CallHome.AnB is strongly dependent on the criteria that both $pw(A, home)$ and $pw(B, home)$ are strong cryptographic keys which are only shared between A and the trusted third party *home*, and between B and the trusted third party *home*.

One possible way to protect the weak password again is to use a key shared between A and *home* and encrypt the message with it, since the password is weak in terms of protection but the key is not.

2.5 Question 5

OFMC finds an attack in this case, because *home* is now capable of being replaced by the intruder. If he has the knowledge of the protocol and the MAC functions with the inputs, he can create a shared secret key with A, authenticating A the same way *home* usually does (with MAC function), but this time not using a key he got from B, but a key that the intruder knows, like the constant g.

In this step $i- > (x20, 1) : step4$, we can see that A can read the messages encoded by MAC function, and, therefore, she believes that she is talking with B, and can trust *home* and B. She receives a shared secret key with B in this step. Then she sends B the secret message M, but the intruder is able to open and read the secret message M. The agent *home* is impersonated by the intruder, and the intruder can read the inputs of MAC functions and change its content. Therefore, the secrecy goal is broken.