

Computer Security  
Coursework Exercise CW2  
Solutions

**Cryptography and Protocols**

School of Informatics  
University of Edinburgh  
<http://www.inf.ed.ac.uk/teaching/courses/cs>

This is an individual assessed coursework exercise. It will be awarded a mark out of 25. It is one of two assessed exercises in the Computer Security course. Each exercise is worth 12.5% of the final result for the course. The deadline for completing this coursework is 16:00, 25th March 2016. The final page summarises submission instructions.

**Question 1: Hashes, MACs, and integrity**

**[10 marks]**

1. Consider two hash functions  $h_1, h_2 : \{0, 1\}^* \rightarrow \{0, 1\}^n$ . Build a hash function  $h : \{0, 1\}^* \rightarrow \{0, 1\}^m$  that is collision resistant provided that at least one of the two hash functions  $h_1, h_2$  is collision resistant. Prove that if a collision on  $h$  can efficiently be found than collisions for  $h_1$  and  $h_2$  can also efficiently be found. [3 marks]

We define  $h$  as the concatenation of  $h_1$  and  $h_2$ . That is

$$\forall m \in \{0, 1\}^*. h(m) = h_1(m) || h_2(m)$$

Assume that a collision on  $h$  can efficiently be found. That is we can efficiently find messages  $m_1, m_2 \in \{0, 1\}^*$  such that  $m_1 \neq m_2$  but  $h(m_1) = h(m_2)$ . By definition of  $h$ , we have that  $h_1(m_1) || h_2(m_1) = h_1(m_2) || h_2(m_2)$ . But then it must be that  $h_1(m_1) = h_1(m_2)$  and  $h_2(m_1) = h_2(m_2)$ . So  $m_1, m_2$  are also a collision for both  $h_1$  and  $h_2$  that can efficiently be found.

To be considered secure, a MAC function must resist existential forgery under chosen-plaintext attacks: even if an attacker knows some  $(m_1, MAC(m_1, k)), \dots, (m_n, MAC(m_n, k))$  message/tag pairs under some key  $k$  not known to him, it should be hard to compute a valid pair  $(m, MAC(m, k))$  without knowing  $k$  (with  $(m, MAC(m, k)) \neq (m_i, MAC(m_i, k))$  for all  $1 \leq i \leq n$ ).

2. You are given a MAC function  $f(\cdot, \cdot)$ . For efficiency purposes, instead of applying the  $f$  to the key  $k$  and the entire message  $m$ , you proceed as follows:
- First break  $m$  into 128-bit blocks (for simplicity we assume that the size of  $m$  is a multiple of 128);
  - Then XOR all the blocks together to obtain a 128-bit block  $m^\#$ ;
  - Finally apply function  $f$  to  $k$  and  $m^\#$ .

In other words if  $m = m_1 || \dots || m_n$  with  $|m_i| = 128$  for all  $1 \leq i \leq n$ , then you compute  $f(k, m_1 \oplus \dots \oplus m_n)$ . Show that this construction is subject to existential forgery under chosen plaintext attacks even if the initial MAC function  $f$  is secure. [3 marks]

There are many possible attacks. I only list here a few.

Let  $(m, MAC(m, k))$  be any valid message/digest pair such that  $m = m_1 || \dots || m_n$  with  $|m_i| = 128$  for all  $1 \leq i \leq n$  and  $m^\# = m_1 \oplus \dots \oplus m_n$ . The attacker can produce the following different from  $(m, MAC(m, k))$  valid message/digest pairs:

- $(m^\#, MAC(m, k))$  if  $n > 1$
- $(m || \bar{0}, MAC(m, k))$  where  $\bar{0}$  denotes the 128-bits zero message
- $(m || m' || m', MAC(m, k))$  where  $m'$  is any 128-bits message
- $(m', MAC(m, k))$  where  $m' = m_{\pi(1)} || \dots || m_{\pi(n)}$  with  $\pi$  being any permutation of  $\{1, \dots, n\}$

3. In class we discussed several modes of encryption for block ciphers, such as the ECB-mode or the CTR-mode. We are here going to consider a different mode of encryption called Output Feedback (OFB) mode.

- (a) Briefly explain how does the OFB mode of encryption work. [1 marks]

See [https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation#Output\\_Feedback\\_.28OFB.29](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Output_Feedback_.28OFB.29)

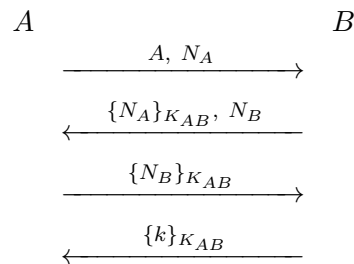
- (b) To ensure both confidentiality and integrity of a message  $m$ , we compute the ciphertext  $c = E(k, m || h(m))$  where  $E$  is a block cipher and  $h$  a hash function. Show how an attacker given a plaintext/ciphertext pair  $(m, c)$  under some key  $k$  that he doesn't know, is able to encrypt and provide message integrity to another message  $m'$ , i.e., compute  $c'$  such that  $c' = E(k, m' || h(m'))$  without knowing  $k$ . [3 marks]

The attacker can retrieve the keystream  $ks = c \oplus m$  used to encrypt  $m$  and encrypt any other message  $m'$  as follows  $c' = ks \oplus (m' || h(m'))$  provided that  $|m'| \leq |m|$ .

## Question 2: Authentication and key-agreement protocols [10 marks]

Consider the following two-party authentication and key agreement protocol. Alice (A) and Bob (B) want to establish a session key using a long-term symmetric key  $K_{AB}$ . First Alice

generates a nonce  $N_A$  and sends it along with her identity to Bob. Bob generates his own nonce  $N_B$  and sends it together with the encryption of Alice's nonce under the long-term key  $K_{AB}$ . Alice acknowledges receipt of this message by sending the encryption of Bob's nonce under the long-term key. Finally Bob generates the session key  $k$  and sends it to Alice encrypted under  $K_{AB}$ .



1. This protocol is flawed. Show how Eve could learn a session key that Alice thinks she has securely established with Bob. (You will assume that nonces and keys have the same length) [3 marks]

The following diagram depicts such an attack.

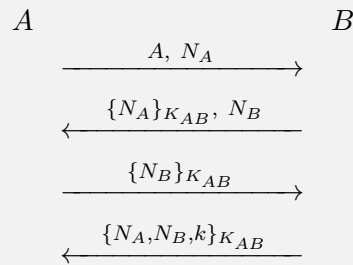
```

sequenceDiagram
    participant A
    participant E
    participant B
    A->>E: A, N_A
    E->>B: A, N_A
    B-->E: {N_A}_{K_{AB}}, N_B
    E-->A: {N_A}_{K_{AB}}, N_B
    A->>E: {N_B}_{K_{AB}}
    E->>B: {N_B}_{K_{AB}}
    B-->E: {k}_{K_{AB}}
    E-->A: {N_A}_{K_{AB}}
    
```

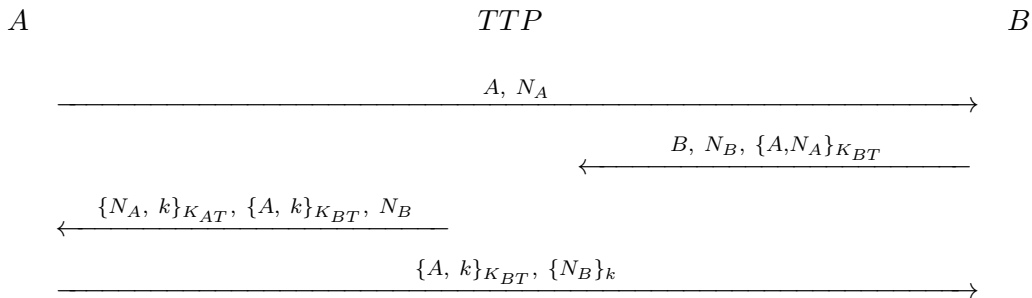
At this point  $A$  thinks she has securely established the key  $N_A$  with  $B$ .

2. Propose a way to fix the protocol to defend against this attack. Explain why your fix prevents this attack. [3 marks]

Of course having nonces and keys be of different size would thwart this attack. But several other possibilities too to fix this protocol. For example include  $N_A$  and/or  $N_B$  in the message that contains the key

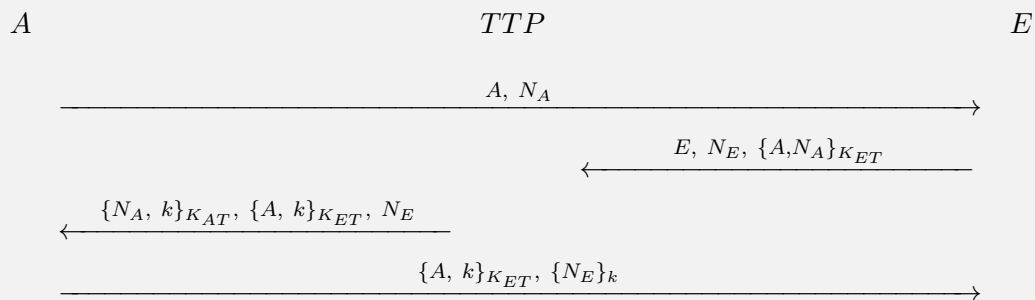


If Alice and Bob do not share a long-term symmetric key they could use the following three-party authentication and key agreement protocol that relies on a trusted third party (TTP). Alice and Bob both share a long-term symmetric key  $K_{AT}$  and  $K_{BT}$  respectively with the TTP.



3. This protocol is flawed. Show how Eve could learn a session key that Alice thinks she has securely established with Bob. (You will assume that nonces and keys have the same length) [2 marks]

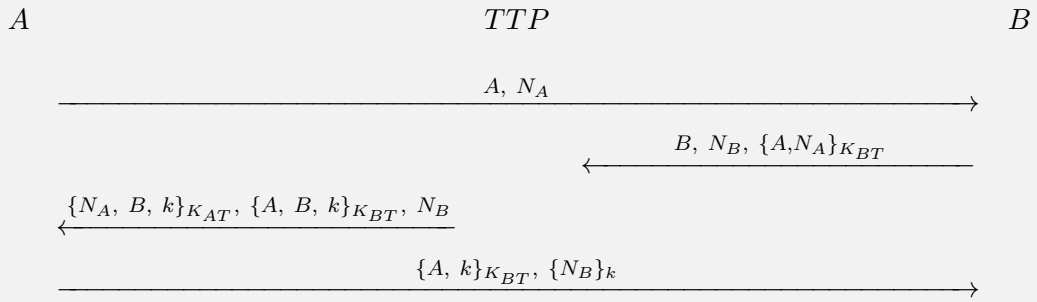
The following diagram depicts such an attack.



At this point  $A$  thinks she has securely established the key  $k$  with  $B$ .

4. Propose a way to fix the protocol to defend against this attack. Explain why your fix prevents this attack. [2 marks]

The identity of  $B$  should be included in the ciphertext from the  $TTP$  to  $A$



Similarly, to avoid an attack on Bob's perspective the identity of  $A$  and  $B$  should be included in the ciphertext from the  $TTP$  to  $B$ .

### Question 3: Verify a cryptographic signature

[5 marks]

In this part of the coursework you will be verifying PGP signatures on files. Each student has their own set of files to verify. The answer to this section will be different for every student, please make sure you follow directions and turn in the answer associated with your student number.

Alice uses her bank to ask Bob to send some money, but she is suspicious that Eve will send a similar looking request at the same time and confuse Bob. To handle this issue Bob and Alice agree to use three code words in the "comment" field provided by the bank. That way Bob will be able to identify which request is really from Alice. Or if there are more than one request with the same code, he will know that he has been attacked and can resort to more secure methods.

Alice has sent Bob a file that contains the three word code. Confidentiality is not important here, anyone can read the code Alice sent and that is not a problem. Integrity is VERY important, it is vital that Bob gets the correct three words and that the words are not changed in transit. Eve wants to confuse Bob so that he uses the code words she sends instead of the ones Alice sent, so Eve sends Bob four other files all of which look like they were sent by Alice.

Your task is to identify which file is correctly signed by Alice.

- Using a command line go to: `/group/teaching/cs/CW2/`
- Here you will find Alice's public key. You can assume that this is actually Alice's public key and that her and Bob properly verified it out-of-band in advance.
- Change the directory to your student number.
- You should see five files named `file1.txt.asc` through `file5.txt.asc`
- Use the `gpg` command to verify the signatures on all five files. To do this you will need to look at the manual for `gpg` and look online for guidance on how to correctly use it.

**Turn in:** The name of the file that is correctly signed by Alice and the three words in the file.