COMP8760 Week 18 Class Worksheet

Budi Arief (b.arief@kent.ac.uk) – Based on materials by Shujun Li and Pierre Mondon
Class Supervisors: Imad Mahaini (M.I.Mahaini@kent.ac.uk) and Joshua Sylvester (jrs71@kent.ac.uk)

Thursday 5 December 2024

In this week's class, you will work on some exercises beyond user authentication to gain a better understanding of authentication in the wider sense. The first three are easier and mandatory (and should be included in your CW2 practical report), while the rest are more advanced and optional.

Like in the last week's class, for this week's class exercises, **you are encouraged to use existing third-party libraries and tools as much as you can**, as long as you will still write some source code yourself to get the exercises done. You should use Google and/or other search engines to get help and look for hints. Work independently first before asking for help.

For all exercises, feel free to show the work you have completed to class supervisors; but if you are confident with your results and do not feel the need to get them checked, that is also fine.

Note: This class is the third class that you will need to include in your CW2 practical report (i.e. the log book). *Please keep any evidence of the work you achieved (e.g. as screenshots or answers to specific exercises), and include them in the practical report that you will submit as part of CW2.*

Exercise 1 – Message authentication with HMAC (hash-based message authentication code)

Define a shortened HMAC algorithm to produce 16-bit HMACs. (Hint: Look for a library supporting HMAC algorithms and then truncate the returned MAC.)

Imagine that Alice is sending £10 to her friend Bob via her bank. Assume that Alice shares a secret key with the banking server, and agrees with the banking server to use the above 16-bit HMAC algorithm to authenticate any transaction requests.

Now assume that there is an attacker Eve who controls the communications channel between Alice and the banking server, and can see the messages transmitted in clear and make changes to the communications. Eve did not know Alice's key shared with the banking server. Write a script to answer the following questions:

- If Eve just changes the message to something like "Alice, Eve, £1000" without changing the HMAC, will the server reject the manipulated message?
- If Eve keeps changing the transfer amount and sending a manipulated message to the server, how many times does he need to try so that he can finally produce a message accepted by the server? How does this relate to the size of the HMAC values (16 bits)? Assume the banking server does not block repeated unsuccessful attempts. (In real world, the server will normally tolerate up to a fixed number of consecutive failures.)

Assuming the key shared between Alice and the banking server was derived from a weak password chosen by Alice following a public (known) key derivation algorithm. Can Eve find a way to crack the key (password)? No need to write a separate script to crack an example weak password (which you should have done in previous weeks). Just include a discussion in your summary of this exercise in your CW2 report.

Exercise 2 – Needham-Schroeder protocol

Write a script to demonstrate how Needham-Schroeder protocol shown on Slide 30 of Lecture 19 (covered on Monday 2 December 2024, i.e. the first lecture in Week 18) works.

Needham-Schroeder protocol (1978)

University of Kent

- Assumptions:
  - A and B both trust a server S (an identity provider).
  - A and S share a secret key K_{AS}.
  - B and S share a secret key K_{BS}.

Symmetric-key version

- The goal
  - For A and B to establish a session key K_{AB}.
- The protocol
  1. A → S: A, B, N_A
  2. S → A: $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
  3. A → B: $\{K_{AB}, A\}_{K_{BS}}$
  4. B → A: $\{N_B\}_{K_{AB}}$
  5. A → B: $\{N_B - 1\}_{K_{AB}}$

N_A is a nonce.

$\{x\}_K$ = using K as the key to encrypt x

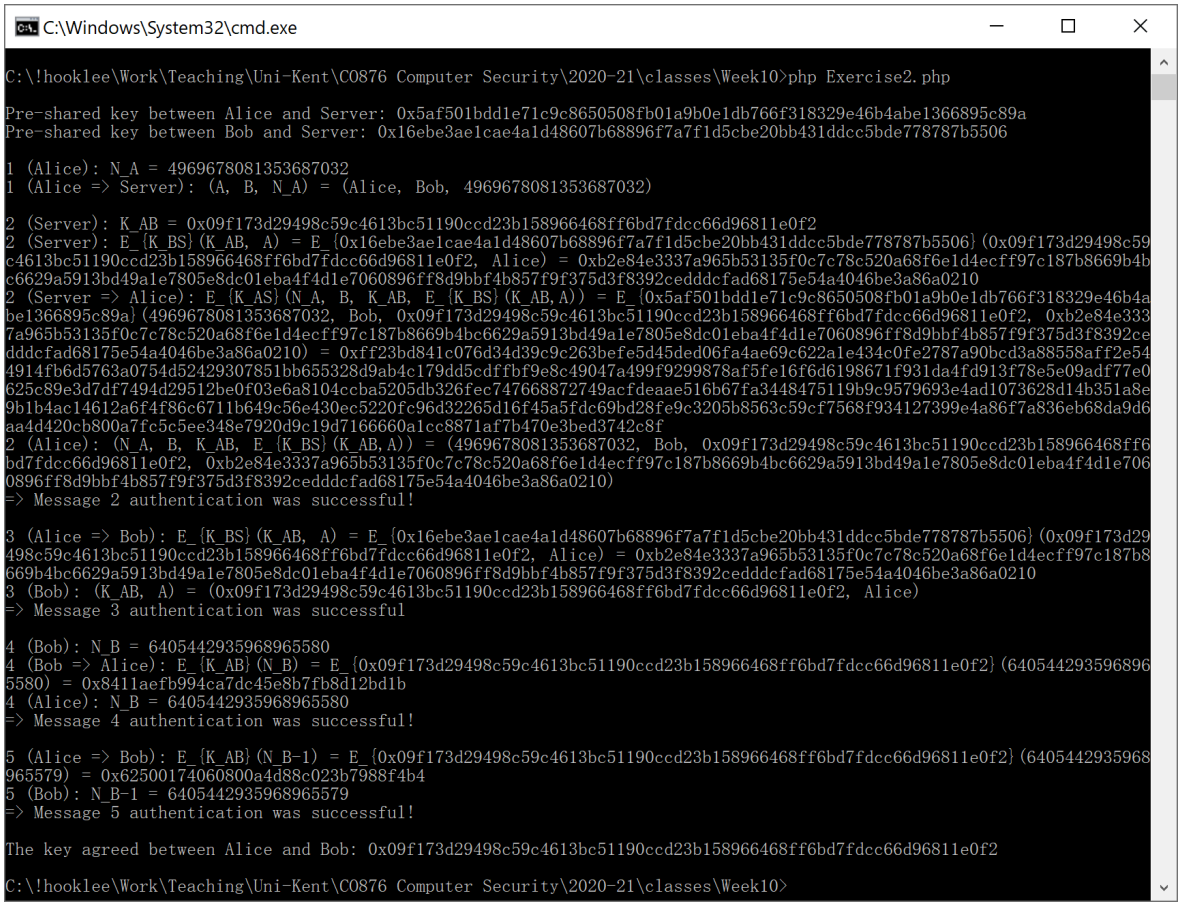
N_B is also a nonce.

30

No need to use three separate processes to simulate the protocol. It is sufficient to use a single script that simulate how the three parties (A, B and S) interact with each other to finally allow A and B to agree on a key (which is generated by S for A and B).

Print all useful information for all steps to show the internal working mechanisms of the protocol. Include authentication of each message when it is relevant and stop the protocol if a message does not pass authentication.

The output of the script should look like what is shown in the screenshot of an example PHP script (which was run on a Windows platform).



Exercise 3 – Attacking Needham-Schroeder protocol

Modify the script you implemented in Exercise 2 above to demonstrate how the replay attack described on Slide 31 of Lecture 19 (in Week 18) works for an attacker.

Needham-Schroeder protocol: an attack

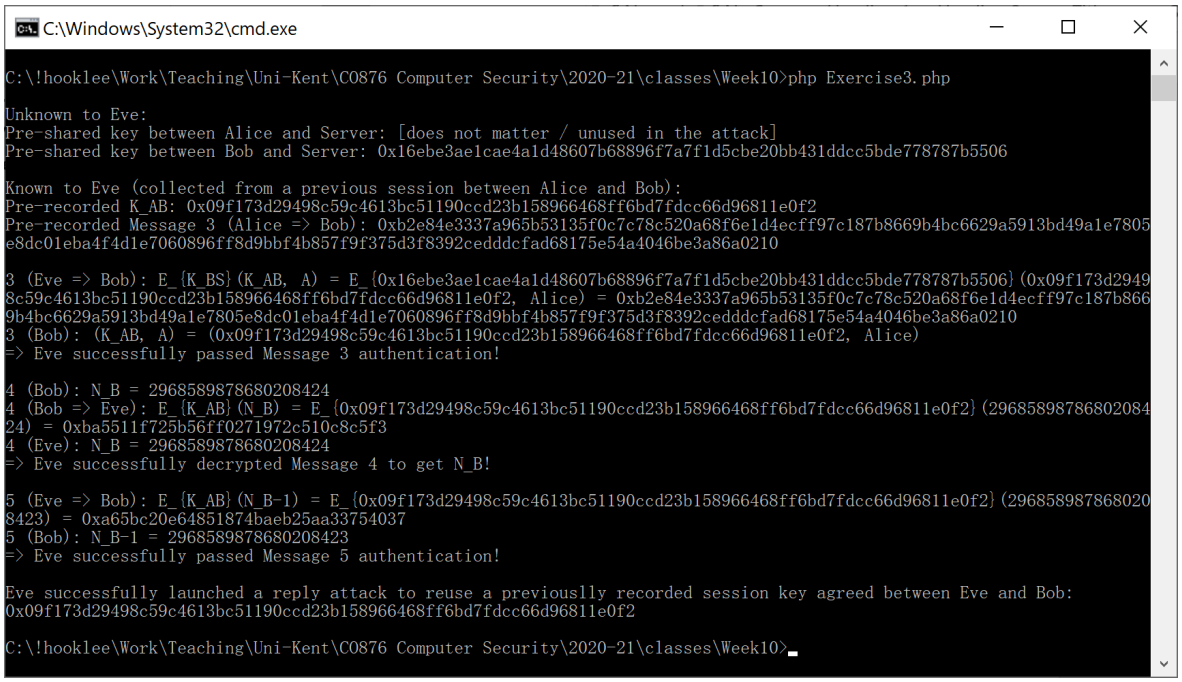
University of Kent

- Assumptions
  - E (an attacker) somehow got K_{AB}.
    - How?
  - E observed the session between A and B when K_{AB} was established.
- The goal
  - E wants to reuse K_{AB} to impersonate A to communicate with B in future.
- The (replay) attack
  3. E → B: $\{K_{AB}, A\}_{K_{BS}}$
  4. B → E: $\{N_B\}_{K_{AB}}$
  5. E → B: $\{N_B - 1\}_{K_{AB}}$

***B* will believe he is communicating with A after this step.**

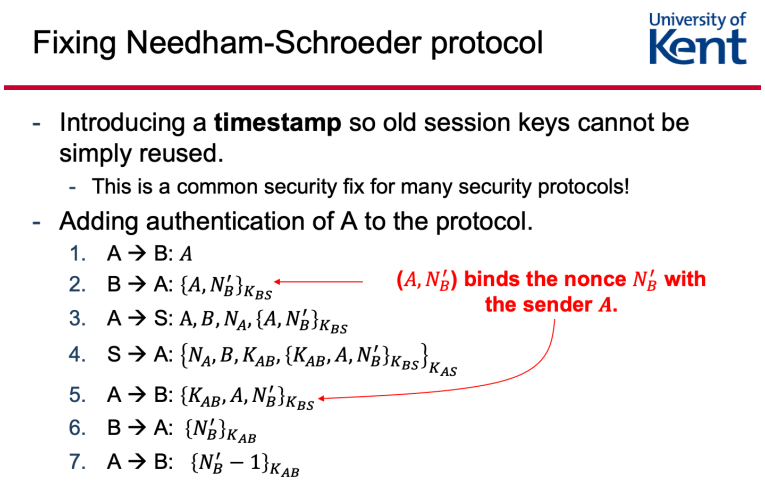
31

Note that you need to record K_AB and Message 3 from a previous session of the protocol. You will also need the same K_BS assuming B has not changed her shared secret with S.



Exercise 4 (optional) – Fixing Needham-Schroeder protocol

Revise the script you wrote for Exercise 2 to implement a more secure version of Needham-Schroeder protocol shown on Slide 32 of Lecture 19. Show that the replay attack demonstrated in Exercise 3 will not work anymore.



Exercise 5 (optional) – Breaking a selected CAPTCHA scheme

Select an existing CAPTCHA scheme used by one or more real world websites, and write a script to break it with a sufficiently high accuracy.

References:

[1] Shujun Li, http://www.hooklee.com/CAPTCHA.asp, web page

[2] Shujun Li, http://www.hooklee.com/eBanking_CAPTCHAs.asp, web page

[3] Antreas Dionysiou and Elias Athanasopoulos, "SoK: Machine vs. machine – A systematic classification of automated machine learning-based CAPTCHA solvers," *Computers & Security*, vol. 97, Article No. 101947, Elsevier, 2020, https://doi.org/10.1016/j.cose.2020.101947