Like in the last week, for class exercises in this week, **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 other search engines to get help and look for hints.

**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 Part 1 of your CW2 report.

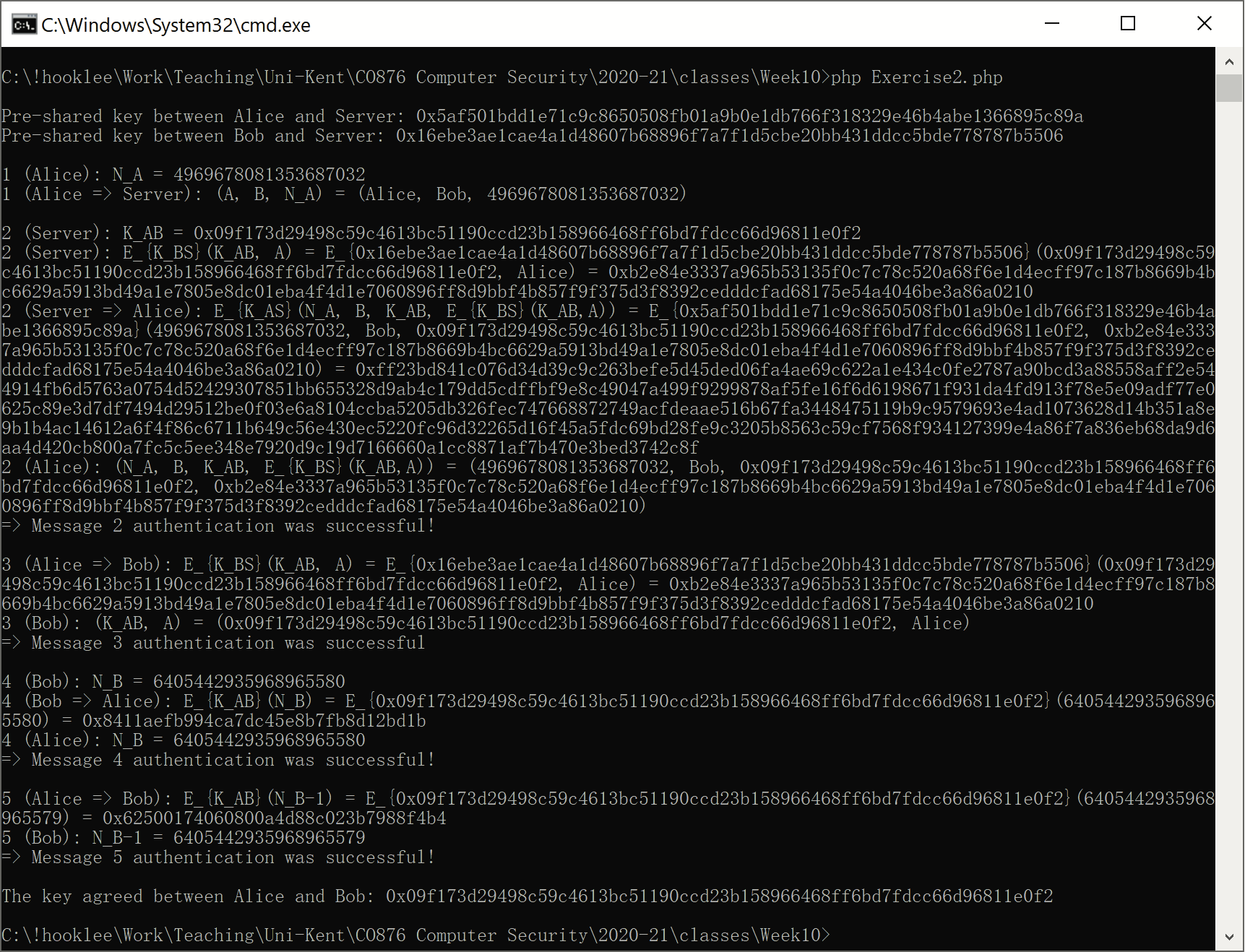
**Exercise 2 – Needham-Schroeder protocol**

Write a script to demonstrate how Needham-Schroeder protocol shown on Slide 30 of Week 10 lecture works.

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).

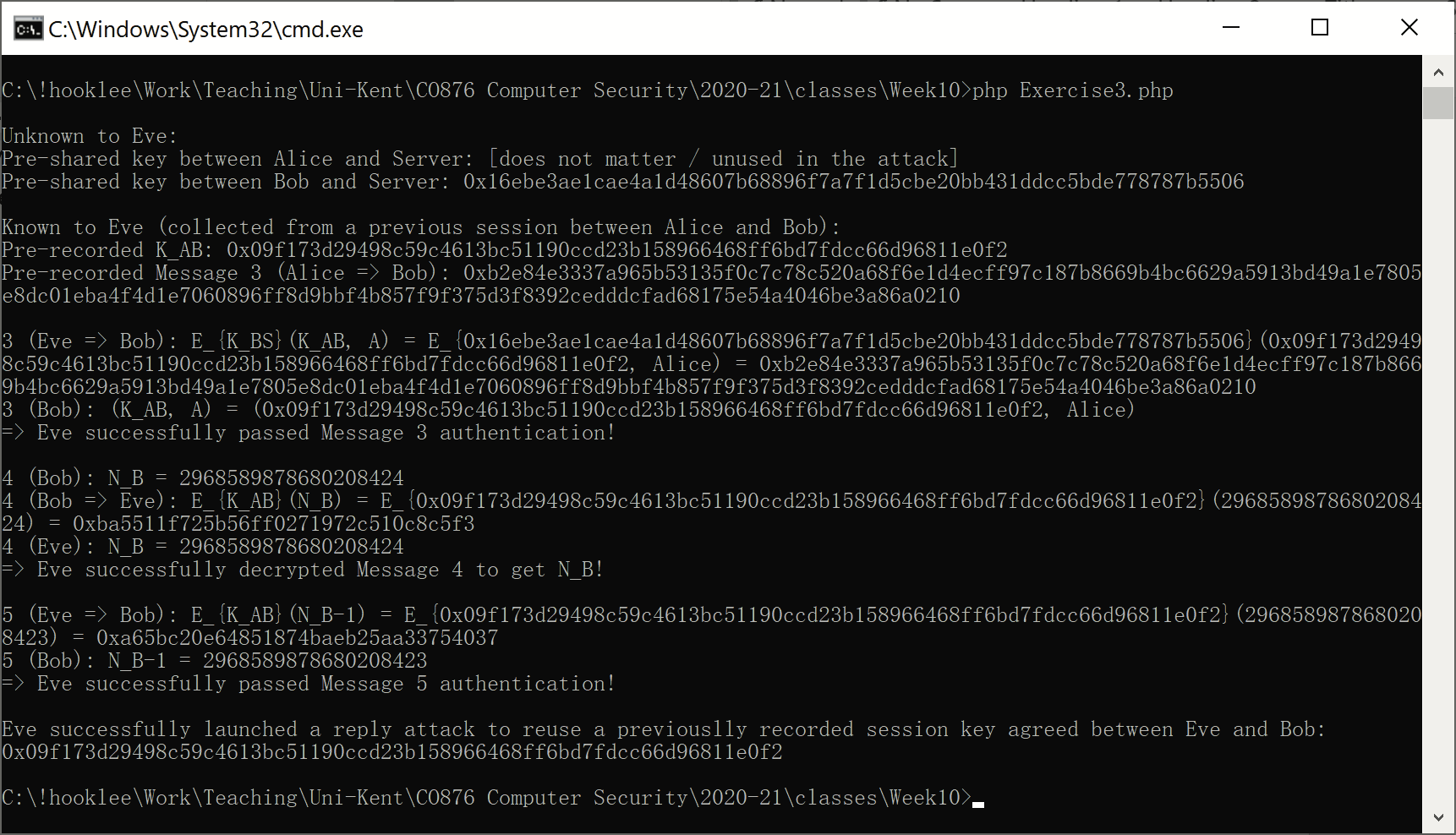
Print all useful information for all steps to show the internal working mechanisms of the protocol. Include authentication of each message when it is relevant and stop the protocol if a message does not pass authentication.

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



**Exercise 3 – Attacking Needham-Schroeder protocol**

Modify the script you implemented in the previous exercise to demonstrate how the replay attack described on Slide 31 of the Week 10 lecture work for an attacker. 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 – Setting Linux file and folder permissions**

Write a script to automatically convert a human-entered Linux permission string (like “rwxr-xr--”) into an integer that you whose octal representation is like 754.

You may want to implement it as a simple web page with JavaScript so you can take the permission string from an <input> element. If you decide to do so, feel free to make the interface even more user-friendly, e.g., implementing a user interface like the one at https://chmod-calculator.com/.

(Optional) Consider making your script fault tolerate (fixing too short or too long input string, wrong permission flag character at any position, etc.).

**Exercise 5 – Investigating a real-world example of RBAC**

Find a real-world example of RBAC and document what roles are defined. This can be your Windows user management system or a website you are involved as an administrator. If you cannot find a good one you are using, search for an online application with RBAC access control (e.g., a web forum, a Wiki, a CMS), install it on your local computer and then examine it.

**Exercise 6 – Learning about same-origin policies (SOP) and cross-site scripting (XSS) attacks**

Visit this website to play a game to learn about XSS attacks that can compromise the SOPs of a target website: https://xss-game.appspot.com/. Think about how XSS attacks circumvent the SOPs that are supposed to be mandatorily enforced.

**Exercise 7 – Playing with a sandbox (a computing one of course)**

Find a sandbox used by a computing system and examine its security settings and/or any existing restrictions on access control. Think about why such security settings and/or access control policies are needed and how they are enforced. Search for any published vulnerabilities that can compromise the security settings.

For the report, describe what sandbox you choose and what you learned about it **briefly**.

Hint: Examples include mobile apps, Docker containers, virtual machines, virtual servers (e.g., a virtual server hosted by a web hosting company), and <iframe> elements in a web page.

**Exercise 8 – Examining a real-world federated identity management (FIM) system**

Find a real-world FIM system you’ve used and examine how technically the system is / may have been implemented. Search for technical documents related to the system to understand more.

For the report, describe what FIM system you examined and what you learned about it **briefly**.

Hint: To identify the techniques used behind a FIM system, search for its name and examine any technical information you may have access to (e.g., HTML source code returned from a website, source code of the system if published under an open source license).

===== Report Template ===== (Exercise 1-4 = 1page report. Exercise 5-8 = 1page report)

**Introduction.** A brief introduction on what the exercises are about.

**How did I do the exercises and what did I learn about**? - Describe how you attempted the exercises and what you learned about from doing them.

**What difficulties / problems did I encounter?** / What observations and/or thoughts did you have on the exercises? - Describe any difficulties and/or problems you encountered (if any) and how you solved them (or if you could not solve them, explain what you did in your attempt to solve them and why it was not possible to overcome the problems). If you did not encounter any difficulties or problems, state so and describe some observations and/or thoughts you had on the class exercises and results, Basically, in this section write something beyond the exercises you attempted.

Where relevant, for both the above questions provide evidence of your work (acceptable evidence includes – but not limited to – screenshots, tables, diagrams, pictures of work done on paper).