Computer Science, Multimedia and Telecommunications Studies

Networks and Internet Applications
Activity: CA3 – Third Continuous Evaluation Test

- The solution must be delivered in a PDF file in the subject classroom.
- You must include references to the resources you consulted to answer the questions.
- The delivery deadline is **May 25th 22, 2025**.

Questions

For each question, reference all sources used to answer it. In case of using the book, reference the corresponding section. Any answer in which external material (books, web pages, artificial intelligence, all types of UOC documentation, etc.) has clearly been used, and its reference is not included, will be considered incorrect.
Answer with your own words. Any answer directly copied or translated from the source will also be considered incorrect.

1. We want to make a polyalphabetic encryption with 2 alphabets, where the first alphabet (C1) is a Caesar cipher and the second (C2) is a monoalphabetic cipher. Define the 2 alphabets, different from those in the book, set the repeating pattern, which can be any number of elements (it does not need to be 5 as in the book) and encode a short message (10-20 characters) that includes your first name (or last name).

   Your answer should clearly show the character that replaces each character in the original message. You can use a font family such as Courier new, which has a fixed width, or a table.

   Once encrypted, in addition to both the plaintext and the encrypted text, also include the texts removing spaces and punctuation marks (it is not necessary to encrypt again because they have not been taken into account), as it is usually done to make it more difficult for attackers, and so it will look more like the real encrypted texts.

   Note: as in the book, use only the English alphabet, lowercase letters, and spaces and punctuation marks are kept the same as in the ciphertext, and do not count (as if they were not there) when applying the repetition pattern.

2. The following sentences are false, since they contain contradictory information. Propose possible corrections for each of the sentences (in the solution, underline or mark the parts that you have modified with respect to the original sentence to correct them).

   a) In stream encryption, the data is split and encrypted into blocks.
   b) The best way to get source authentication of a message is to use symmetric encryption.
   c) Between Alice and Bob, the session keys mechanism uses the same symmetric key each time (connection/session/message).
   d) The length of the hash of a message is always approximately equal to the length of the original message.

3. Connect to the following website:

   https://www.devglan.com/online-tools/rsa-encryption-decryption

   and generate a 512-bit asymmetric key pair (the dropdown list says 515).

   In the *RSA Encryption* part:
   - In "*Enter Plain Text to Encrypt*" enter a very short text that includes your first name (or last name).
   - Put one of the two keys (public or private) in "*Enter Public/Private key*", and select the type of key.
   - Select "RSA" in "*Select Cipher Type*".
   - Click "*Encrypt*".

   In the *RSA Decryption* part:
   - Copy the encrypted text from the previous step to "*Enter Encrypted Text to Decrypt (Base64)*".
   - Enter the complementary key to the one you have put in the encryption section in "*Enter Public/Private key*" and select the type of key.
   - Select "RSA" in "*Select encryption type*".
   - Click "*Decrypt*".
   - Check that the decrypted text is the same as what you encrypted.

   a) Briefly explain how the RSA public key encryption algorithm works. What characteristic/relationship must the key pair meet for the mechanism to work? (Hint: combine the decryption formula with the encryption formula).

   b) Put a screenshot showing all the fields filled out.

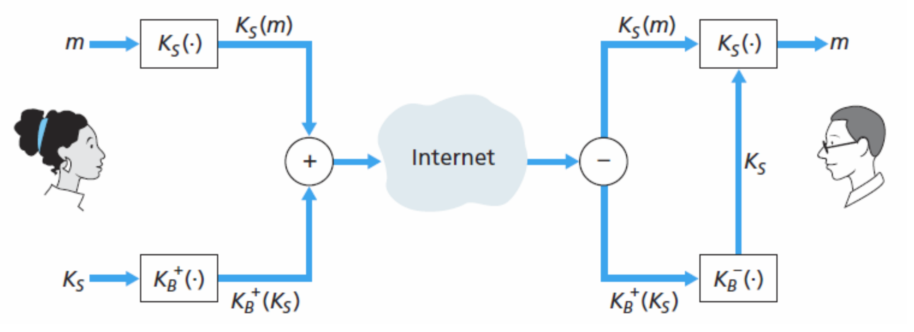
   c) Say whether you chose to use the public or private key first, and what security you will get in this case.

   d) Why is the output actually 512 bits (88 characters in base64, with 2 '=' at the end) when the input only has 12?

   e) Now, change the first character of "*Enter Encrypted Text to Decrypt (Base64)*" and click at "*Decrypt*". What message do you get? Justify it, and tell what security service it provides.

   f) Try "*Encrypt*" again. Why does it give a different string? Even though it is a different string, can you decrypt it with the same key? Put a screenshot and explain why it is like that.

4. Briefly explain the mechanism that Alice and Bob use to exchange a message in Figure 8.19 in the book, and how it works (from both Alice's and Bob's points of view). Justify the security services provided (confidentiality, integrity and/or origin authentication), and find in the book how all 3 could be obtained.



5. Table 8.4 of the book, page 664, lists the main fields found in a Certificate (X.509 certificate) with a brief explanation. List them and briefly justify what their functionality/use is.
   What is a digital signature, how is it generated, and what does it have to do with certificates?

6. IPSec is explained in the course book as a mechanism for implementing VPN networks.
   a) Given the layer at which they work (and the information that can be found within PDU fields), briefly compare Transport Layer Security (TLS/SSL) and Network Layer Security (IPSec).
   b) IPSec has 2 working modes: Transport mode and Tunnel mode (this is explained later in the book). Briefly explain (if you want, with the help of an image) what the PDUs of the 2 modes are like. There is no need to differentiate AH and/or ESP and you can use "IPSec Header" as a generic heading that includes "AH and/or ESP", as shown in Figure 8.27 of the book.
   c) Considering that a datagram that is sent must always have the destination address and also the source address accessible so that the datagram can be received and the response returned, and that in general these cannot be modified along the way, say and justify which of the 2 modes can be used to make the following 3 secure IPSec communications:
      1. **Sender <=IPSec=> Receiver**
      2. Sender <-IP-> **Router 1 <=IPSec=> Router 2** <-IP-> Receiver
      3. **Sender <=IPSec=> Router** <-IP-> Receiver

7. About the "Diffie–Hellman" mechanism:

   a) Find out what it is used for.
   b) Briefly explain how it works.
   c) What is *Ephemeral Diffie-Hellman* and what is the relationship with HTTPS and session keys?
   d) Give a simple numerical example of the "Diffie-Hellman" mechanism (different from the one you find on Wikipedia, of course, although p and/or g may be the same). For simplicity, choose "small" numbers a, b, p, and g, with p a prime number.

8. **(3 points)** You must perform a series of tasks detailed in the exercise in order to test a web service that we have implemented (see connection details in Appendix A) that uses JSON Web Token (JWT) to authenticate users of the service. Authentication is done with the OAuth 2.0 protocol (defined in RFC 6749) and to access protected resources you must use a *bearer token* (defined in RFC 6750), where the JWT to be used will go. Give reasoned answers to the following sections:

   a) Find information about the OAuth 2.0 protocol and JSON Web Token (JWT). Briefly explain how the OAuth 2.0 protocol is used, what its main operations are, and how it relates to JWTs in order to authenticate users.

   b) Prepare a call with the `curl` tool available on Linux in order to obtain a JWT (see `token` operation connection details of the service in Annex A). This operation returns a token that you must store in a text file to later use in the following sections. In the answer, put the call you prepared with the `curl` tool. Also, capture the request made with Wireshark and paste a screenshot showing the request and response information.

   c) Prepare a call with the `curl` tool in order to execute the operation `square` of the service, which calculates the square of the number passed to it as a parameter (see the connection details to the `quare` operation in Annex A). The JWT from the previous section must be used for this call. In the answer, put the call you have prepared with the `curl` tool. Also, capture with Wireshark the request made and paste a screenshot showing the request and response information.

Note: For the test in DSLab, use 2 as `{number}`.

d) You have to put the `curl` calls above in two Linux `shell` type files, `token.sh` and `square.sh`, respectively, which we give you with the statement. These files should be uploaded to DSLab (see details in Appendix A) to automatically check that your calls work. If the response is as expected, you will get successful execution. If not, you will get an error message that tells you what errors were encountered when running the commands.

e) Finally, check that the headers of the requests and the responses you obtained in the different sections match the theoretical information you found in section a.

Annex: Description of the REST web service

The REST service we provide you offers an Application Programming Interface (API) with some operations that are accessed via the HTTP protocol.

The service details are as follows:

- REST API server address: `http://labxarxes.techlab.uoc.edu`
- Port servidor API REST: `8095`

The `token` operation must be called with HTTP's POST method. The endpoint of the operation is `/dslab-api/token`. Also, on the `curl` call, the "`X-TenantID: prova`" header must be included to indicate that the request is made to the "`prova`" deployment. The `curl` user and password must also be included to be able to make the call correctly. The `token` must be saved to a file for later use.

The `square` operation must be called with the HTTP GET method. The endpoint of the operation is `/dslab-api/xai/square/{number}`. In addition, an HTTP header must be included in the curl call to indicate which OAuth2.0 Bearer token is used for authentication. This Bearer token will be in the file that was created with the call to the previous `token` operation.

To test in DSLab, we provide two templates, `token.sh` and `square.sh` where the `curl` commands needed to connect to the service must be included. The second file depends on the result of the first one. These files are the ones that have to be uploaded to DSLab. In addition, we provide the `comprovacio.sh` file, which can be used locally to perform functional checks.

To do the connection tests, randomly choose one of the users listed below. This is to minimize that if another student is testing at the same time, the operation fail because concurrent token requests with the same user are being made. All users have the same password:

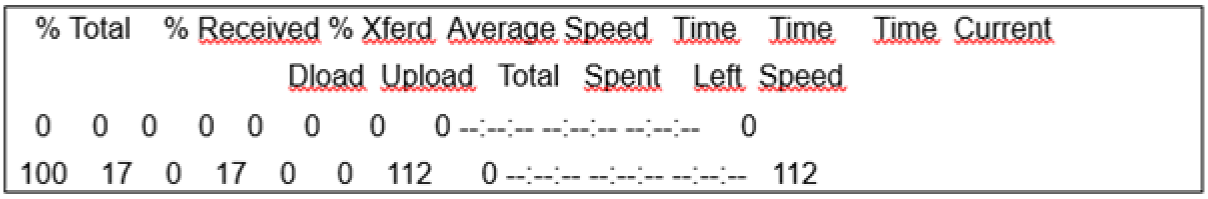
User: between `e100@uoc.edu` and `e109@uoc.edu`

Password: `TfM2023,`

It should be noted that, due to the way `curl` works, when calling `comprobacio.sh` the following files are generated:

- `resultats_/token_stderr.txt`
- `resultats_/square_stderr.txt`

These two files generate output to stderror similar to the following screenshot:



```
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    17    0    17    0     0    112      0 --:--:-- --:--:-- --:--:--   112
```

You can run the tests on a local Linux machine or remotely (DSLab). However, **only answers that have been successfully run in DSLab will be given the maximum score**.

To run your practice in the DSLab autocorrect environment you must follow these steps:

- Create a project (*Projects* > *New Project* menu).
- Select the task that corresponds to the part of the statement to be corrected: *Assignment*

and task.

- Upload the source code of this part by clicking on the project you just created. Notice that at the bottom of the screen, the files that have to be uploaded appear (*Required files)*.
- Create an autocorrect (*Submissions > New Submission* menu) of the project created in the first point. The result of the executions can be consulted in the *Submissions* Menu.