Name: _____

ISM Exam January 26, 2023 (Java - JCA)

- *The Java solution is developed in a single .java file.*
- *You are responsible for defining in that file all the classes that you need to solve the problem. If the solution depends on external source code files (not Java libraries) it will not compile and it will not be evaluated.*
- *The use of Bouncy Castle library is optional*
- *All submitted solution must be without compiler errors (0 errors)*
- *All the solutions will be cross-checked with MOSS from Stanford and very similar source code files will not be evaluated.*

Each student will develop the solution in a single .java file and in a package that has his/her name in it (ex: ro.ase.ism.lastname.firstname)
Each student will use the values given in the users.pdf file. Search for your name in that file.

PART I (0,5 points)

A DB admin asks for your help to update the hash value of a user in his/her database.
He sent you that user password in an encrypted file (with a .user extension). Search for that file as you know its SHA256 hash value in Base64 format.
Print the designated file name at the console.

PART II (1,5 points)

Once you found the file, decrypt it (AES in CBC mode with a known IV - check the user's file (**the index starts at 0**). There is no need for Padding as the file has the required size) using the password sent by your friend (check the **users.pdf** file).
The decrypted content represents the user password as a string with 16 characters.
Print the user password at the console.

PART III (1,5 points)

Add to the user password the "*ism2021*" salt **at the end** and hash it with the PBKDF (Password-Based Key Derivation Function) based on HmacSHA1 algorithm **with 150 iterations**. The output must have **20 bytes**.
Store the result in a binary file (you can choose the filename name). To get the points, the value must be validated by your friend.

PART IV (1 points)

To assure your friend that no one is tampering with that value, digitally sign the previous binary file **with your private key**. Store the signature in another binary file.
Using keytool generate a RSA pair. Export the public key in a X509 .cer file. Use the private key to sign the previous file.
Send your colleague the binary files with the signature and your public certificate.
To get points the digital signature must be validated for the previous file with your public key.

Upload on Sakai

- The .java file with your solution (only 1 file)
- The binary file with the PBKDF hash value
- The binary file with the digital signature of the previous file
- The .cer file with your public key