CA4005 Cryptography and Security Protocols

Assignment 1

Symmetric File Encryption Using Password and Salt

The aim of this assignment is to perform symmetric encryption of a file using the block cipher AES where the key is derived from a *password* and a *salt*.
The password you use should be encoded using UTF-8 and be considered to be "strong".
The salt will be a randomly generated 128-bit value. The password (*p*) and salt (*s*) will be concatenated together (*p*||*s*) and hashed 200 times using SHA-256. The resulting digest ($H^{200}(p||s)$) will then be used as your 256-bit AES key (*k*).
You will then encrypt an input binary file using AES in CBC mode with the 256-bit key *k* and a block size of 128-bits. The IV for this encryption (*i*) will be a randomly generated 128-bit value. You will use the following padding scheme (as given in lectures): if the final part of the message is less than the block size, append a 1-bit and fill the rest of the block with 0-bits; if the final part of the message is equal to the block size, then create an extra block starting with a 1-bit and fill the rest of the block with 0-bits.
You will then encrypt your password using RSA with the encryption exponent (*e*) and my public modulus (*N*). This is done by calculating $p^e \pmod N$. This will use your own implementation of modular exponentiation rather than a library method for this purpose. For the purpose of this assignment no padding should be added to the password before encryption (although this would normally be done).
The encryption exponent (*e*) is 65537 and my public modulus (*N*) has the following value (in hexadecimal):
c406136c 12640a66 5900a9df 4df63a84 fc855927 b729a3a1 06fb3f37 9e8e4190
ebba442f 67b93402 e535b18a 5777e649 0e67dbee 954bb021 75e43b64 81e7563d
3f9ff338 f07950d1 553ee6c3 43d3f814 8f71b4d2 df8da7ef b39f846a c07c8652
01fbb35e a4d71dc5 f858d9d4 1aaa856d 50dc2d27 32582f80 e7d38c32 aba87ba9
The implementation language must be Java. You will have to make use of the BigInteger class (java.math.BigInteger), the security libraries (java.security.*) and the crypto libraries (javax.crypto.*). You must not make use of the modular exponentiation method provided by the BigInteger class; all modular exponentiation must be done using one of the two methods described in the lectures. You can however make use of the security and crypto libraries to perform the AES encryption and the SHA-256 hashing.
Once your implementation is complete, you should create a zip file containing all your code, encrypt this file as described above, and send me the following by email:

- Your 128-bit salt (*s*) in hexadecimal.
- Your 128-bit IV (*i*) in hexadecimal.
- Your password encrypted using RSA as described above in hexadecimal.
- Your zipped code file which was encrypted.

- Your AES encryption of the above zipped code file in hexadecimal.
- A declaration that this is solely your own work (except elements that are explicitly attributed to another source).

When I receive your email I will decrypt your password *p* using my RSA private key. Using this password and your salt *s* I will then generate your AES key *k* using SHA-256. I will decrypt the AES ciphertext using your AES key and IV, which should match the submitted zipped code file for a correct submission.

This assignment is due **10am on Monday 6th November. Submissions without the declaration will not be assessed. This assignment carries 15 marks and late submissions will have 1.5 marks deducted for each 24 hours the assignment is overdue.**