CS5099 - Information Security

Assessment 02 – Developing Secure Cryptographic Primitives

Deadline: Monday 30 November 2020 (23.59hours BST) in Blackboard
Estimated Weightage: 15% (subject to adjustments)

There are three parts to this assignment. All parts are compulsory. Credit all sources you referred to. Students found plagiarising will be reported to the disciplinary committee.

Submissions will be checked against anti-plagiarism checkers. Plagiarism can be punished severely by the University.

- Any standard style of giving references is acceptable. Please look at a few books to decide how you'd like to do it.
- Clear, unambiguous English is expected.
- Significant deviations from these requirements will be penalized. If in doubt, ask at a tutorial or optional seminar session.
- Since feedback is required to provide within 2 weeks of submission --- the available written feedback will be relatively limited.

Submission Instructions:
There should be three parts submitted as a zipped folder named according to the following convention: CS5099_Assignmen2_StudentID.zip (or tar.gz).

Within the zipped folder, each part should be as a separate file (doc, docx or pdf),

1. Part 1: Assignment02_ StudentID
2. Part 2: Assignment02_ StudentID
3. Part 3: Assignment02_ StudentID

Part 1 (40 marks) – Traditional Cryptography

You are a cryptographer in medieval times and the state-of-the-art cryptosystem is **Caesar Cipher**. In this part, you have to encrypt, decrypt and cryptanalyse given texts. For details on how **Caesar Cipher** works, please refer to the respective lecture. You have to program (using either Python or Ruby on Rails programming language) an API "call it CaesarAPI" that provides all three functionalities (encrypt, decrypt, and cryptanalysis). Your API should implement at minimum three functions: Encrypt(plaintext, key), Decrypt(ciphertext, key) and Cryptanalyse(ciphertext).

i. Encryption using **Caesar Cipher**. The user can call Encrypt(plaintext, key) and it returns the ciphertext of the corresponding input. For your report, use the following as the input parameters and detail the respective output.

Plaintext:
I am currently studying Cyber Security module at the Department of Computer Science, University of Waikato.

Key:
16

ii. Decryption using **Caesar Cipher**. The user can call Decrypt(ciphtertext, key) and it returns the plaintext of the corresponding input. For your report, use the following as the input parameters and list the respective output.

Ciphertext:
Zbydomdsxq kx SD sxpbkcdbemdebo sxmvenoc gsno bkxqo yp kmdsfsdsoc drkd rkfo dy lo zobpybwon sx cixm gsdr okmr ydrob. Sx drsc vomdebo, go ohzvybo dro lkcsmc yp lesvnsxq kx SD cicdow drkd sc cdboxqdroxon dy lo k comebo. Dro cdboxqdroxsxq zbymocc boaesboc rkbnoxsxq nsppoboxd mywzyxoxdc yp dro SD sxpbkcdbemdebo sxmvensxq Yzobkdsxq Cicdowc, Xodgybuc kxn Kzzvsmkdsyxc.

Key:
10

iii. Cryptanalyses: The user can call Cryptanalyse(ciphtertext) and it returns all possible plaintext of the corresponding input (26 in total). For your report, use the following as the input parameter and present the plaintext (only one) in the report.

Ciphertext:
Jatp uawn pda Qjeranoepu yahaxnwpao epo benop 50 uawno wjz ej pdwp odknp peia sa'ra xaykia kja kb pda xaop ej pda sknhz. Nayajphu sa dwz kqn lhwya ykjbeniaz wo kja kb pda Pkl 50 qjeranoepeao ej pda sknhz qjzan 50 uawno khz. Pda decdhu nacwnzaz QG-xwoaz Peiao Decdan Azqywpekj Nwjgejco whok lhwyaz qo ej pda pkl 2% kb whh qjeranoepeao sknhzseza.

In your submission, there should be the source code of your CasearAPI and a maximum two page report that should detail the following:

- 5 marks: Detail the design of your CasearAPI.
- 4 marks: Discuss how the design of the" Cryptanalyse(ciphertext)" function can be improved as such that it only returns the correct plaintext (rather than all 25 possible plaintexts).
- 2 marks: Ciphertext output from the "Encrypt" function for the input listed in 'i'?
- 2 marks: Plaintext output from the "Decrypt" function for the input listed in 'ii'?
- 2 marks: Plaintext output from the "Cryptanalyse" function for the input listed in 'iii'??

The API will be marked as below:

- 20 marks: Source code.
    - 4 marks: Coding style
    - 4 marks: Logical and easy to follow comments
    - 4 marks: Handling adequate exceptions
    - 4 marks: Handling the incorrect input parameters (e.g. keys cannot be negative and larger than 25)
    - 4 marks: Overall design of the API
- 5 marks: Professionalism in formatting of the report and the API source code.

Part 2 (30 marks) Modern Cryptography with AES

At present AES is the recommended symmetric key algorithm. In this exercise, you have to code an "ExtendedExperimentalAPI" based on the AES (using Python or Ruby on Rails programming language). You can use the built-in AES library for your API. The functionality of the API is that it takes a file, which can be in any format and encrypts the file contents using a fixed key (provided below). The encrypted output is stored with the same file extension as of the input file (i.e. if you encrypt a JPG file, the output is also stored as a JPG). For each input file, your API will generated three output files, using Electronic Codebook (ECB), Cipher-Block Channing (CBC) and Cipher Feedback (CFB) modes of operations.

i. Key (128bit) in Hexadecimal format for the AES encryption using ECB, CBC, and CFB modes

770A8A65DA156D24EE2A093277530142

ii. For the report, please use the image from https://icons.iconarchive.com/icons/tatice/operating-systems/256/Linux-icon.png

In your submission, there should be the source code of your ExtendedExperimentalAPI and a maximum two page report that should detail the following:

- 4 marks: Discuss the difference between ECB, CBC, and CFB modes of operation.
- 5 marks: Detail the design of your ExtendedExperimentalAPI.
- 2 marks: ECB output of the input image?

- 2 marks: CBC output of the input image?
- 2 marks: CFB output of the input image?

The API will be marked as below:

- 10 marks: Source code.
  - 2 marks: Coding style
  - 2 marks: Logical and easy to follow comments
  - 2 marks: Handling adequate exceptions
  - 2 marks: Handling the incorrect input parameters
  - 2 marks: Overall design of the API
- 5 marks: Professionalism in formatting of the report and the API source code.

Part 3 (30 marks) Secure Password Generation

In this part, you have to develop a "RandomPasswordGeneratorAPI" that generates random passwords as required by the requesting entity (user or another program). The API should have:

i. Generated passwords include alphanumeric and special characters (e.g. '@','#','$','%','^','&', '/', '?' , '[', ']', and '*' etc.).
ii. Requesting entities can request a variable length of passwords (8 to 80 characters in length).

Note: You can use the built-in function for random number generation but you are not allowed to use any built-in function for random password generation.

In your submission, there should be the source code of your RandomPasswordGeneratorAPI and a maximum three page report that should detail the following:

- 6 marks: Design document for your implemented API.
- 5 marks: Discuss the benefits and issues with an access control based on password.
- 4 marks: Elaborate on the pros and cons of random passwords for access control mechanisms in relation to human users.

The API will be marked as below:

- 10 marks: Source code.
  - 2 marks: Coding style
  - 2 marks: Logical and easy to follow comments
  - 2 marks: Handling adequate exceptions
  - 2 marks: Handling the incorrect input parameters
  - 2 marks: Overall design of the API
- 5 marks: Professionalism in formatting of the report and the API source code.