COMP90043 Project Part A Specification

Introduction

The aim of this part of this project is for you to gain some experience writing functions to compute cryptographic keys. For this project, you will write some functions to generate private secret keys, compute public keys and perform fast modular exponentiations with big integers. This is an individual project.

Modular Exponentiation

Modular exponentiation is exponentiation performed over a modulus. Useful in the field of public key cryptography as it a fast operation despite the exponent being large.

For more information on modular exponentiation.

Visit http://williamstallings.com/Extras/Security-Notes/lectures/publickey.html

Diffie Hellman Key Exchange

Diffie Hellman key exchange is a specific method of securely exchanging cryptographic keys over a public, unencrypted channel. D-H is one of the earliest practical examples of public key exchange implemented within the field of cryptography. The D-H key exchange method allows two party that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

For more information on Diffie Hellman key exchange, read up the prescribed textbook for the course:

Cryptography and Network Security Principles and Practice. (PART TWO: Asymmetric Cipher, Diffie-Hellman Key Exchange)

First Time Users of University Engineering Servers

Please read `http://ithelp.eng.unimelb.edu.au/student/general_unix.html`

Provided Packages

- Java users are provided with an external interface to implement the necessary functions. Java users should only use Java 1.8.
- Python users are also provided with an external script to implement the necessary functions. Python users should only use Python 2.6.6

NOTE: Sorry, but these are the only versions available on the University Servers.

Instructions for retrieving source codes:

1. Use `ssh` to login to `[nutmeg, dimefox].eng.unimelb.edu.au`
2. Go to `/home/subjects/comp90043/partA` to look for the package of your choice. (Either Python or Java).
3. Use `scp` or `wget` to fetch this from the Servers. Alternatively, you may choose to develop on the University Servers using `vim`, `vi` or any other editor.

Instructions for testing:

There are TWO ways for you to test your work. You may either test it by following the submission instructions. (RECOMMENDED WAY for ensuring that you have implemented everything correctly.) Alternatively, you may simply run your client directly on the University server to connect to a testing server to get an idea of how your client is running. Using the alternative method will give you little feedback, but you will at least be able to get instanenous response if your client is working properly.

Points to note

- Python Users must not use any exponentiation functions provided by external or standard libraries. The only exponentiation operators you may use is just ** or Bitwise Operators for base 2 exponentiation.
- As a hint for you, you don't actually need to use any exponentiation operator. Usage of bitwise operator and the division operator is more than sufficient.
- Java users should be aware that by default Java does not support Integers more than 32bits in length. Usage of a BigInteger library will be mandatory. Please read the documentation for BigInteger carefully as there may be caveats such as variables being immutable and etc.
- One second timeouts are applied to all methods of your implementation. If for any reason, you received a timeout error while verifying your submission, it means your implementation is incorrect. One second should be more than enough to perform a fast modular exponentiation without any optimization.
- To protect yourself against suspicion of plagiarism, it may be a good idea to log the progress of your work with a Version Control System such as git, svn or bazaar.
- A discussion forum thread Project - Part A has been created on the LMS. Any clarification on the Project, specifically Part A will be considered as part of the specification. Alternatively, if you need help, this is also the place to post questions.

Administration

This part of the project is worth 4% of your total mark for this subject.

Mark Allocations are as follow:

Criteria	Mark
Fast Modular Exponentiation	1.5%
Diffie Hellman Functions	1.5%
Qualtity of Code and Documentation	1%

The due date for this part of the project is set to **Friday, 21st August 2015, 2359 (AEST) - Week 4**.

Submission Instruction

Only include script(s) of your preferred programming language in your archive file. For example, if you have chosen to implement in Python, only include .py file(s) in your archive file and exclude all other scripts we have provided to you. Alternatively, if you are only submitting one file, you may submit that file directly without including it in a zip archive file.

1. Zip the entire package with your work. (OPTIONAL)
2. Login to [nutmeg, dimefox].eng.unimelb.edu.au
3. run `submit 90043 A [whatever].zip`. Alternatively, if you are submitting Python scripts run `submit 90043 A part1_crypto.py`. If you are submitting Java classes, run `submit 90043 A CustomCrypto.java`. Only submit either Python or Java, DO NOT SUBMIT BOTH!
4. run `verify 90043 A` to verify your submission. It should also tell you if it passes all test cases.

Submission Opens: Mid Week 3.

Submission Closes: 21st August 2015, 2359 AEST

You must follow late submission instructions if the submission time has lapsed or else you will not be able to submit your work.

Extensions and Late Submissions

Late submissions will incur a penalty of **1 mark per day (or part thereof) late**.

Late Submission Instructions:

1. Zip the entire package with your work. (OPTIONAL)
2. Login to [nutmeg, dimefox].eng.unimelb.edu.au
3. run `submit 90043 A [whatever].zip`. Alternatively, if you are submitting Python scripts run `submit 90043 A part1_crypto.py`. If you are submitting Java classes, run `submit 90043 A CustomCrypto.java`. Only submit either Python or Java, DO NOT SUBMIT BOTH!
4. run `verify 90043 A.late` to verify your submission. It should also tell you if it passes all test cases.

Late Submission Open: 22nd August 2015, 0000 (AEST)

Late Submission Closes: 25th August 2015, 2359 (AEST)

Extensions may be granted on the provision of a valid reason. You must contact **Tutor/Lecturer** at the earliest opportunity, which in most instances should be WELL before the submission deadline. Requests for extensions are not automatic and are considered on a case by case basis. You may be required to supply supporting evidence such as a doctor certificate.