UNIVERSITY OF HULL

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY
COURSEWORK ASSESSMENT DESCRIPTION 2020/21

MODULE DETAILS:

Module Number:	500088	Trimester:	1
Module Title:	Software Engineering		
Lecturer:	Dr P A Robinson		

COURSEWORK DETAILS:

Assessment Number:	2	of	2		
Title of Assessment:	Software development assignment				
Format:	Report	Program			
Method of Working:	Individual				
Workload Guidance:	Typically, you should expect to spend between	20	and	30	hours on this assessment
Length of Submission:	This assessment should be **no** more than: *(over length submissions **will be** penalised as per University policy)*	Code: No limit Report: 2500 **words** *(excluding diagrams, appendices, references, code)*			

PUBLICATION:

Date of issue:	10/11/2020

SUBMISSION:

ONE copy of this assessment should be handed in via:	Canvas		If Other (state method)	
Time and date for submission:	**Time**	2pm	**Date**	17/12/2020

Will submission be scanned via TurnitinUK?	No	If submission is to be scanned byTurnitin, these should be one of the allowed types e.g. Word, RT, PDF, PPT, XLS etc. Specify any particular requirements in the subumission details Students MUST NOT submit ZIP or other archive formats unless specified. Students are reminded they can **ONLY** submit **ONE** file and must ensure they upload the correct file. Normally only the LAST submission will be considered (and if late incur a late penalty).

The assessment must be submitted **no later** than the time and date shown above, unless an extension has been authorised on a *Coursework Extension Form: s*ee the Canvas site: Help&Support > Student Forms

MARKING:

Marking will be by:	Student number

ASSESSMENT:

The assessment is marked out of:	100	and is worth	50	% of the module marks
N.B If multiple hand-ins please indicate the marks and % apportioned to each stage above (i.e. Stage 1 – 50, Stage 2 – 50). It is these marks that will be presented to the exam board.				

ASSESSMENT STRATEGY AND LEARNING OUTCOMES:

The overall assessment strategy is designed to evaluate the student's achievement of the module learning outcomes, and is subdivided as follows:

LO	Learning Outcome	Method of Assessment *{e.g. report, demo}*
1	*Show evidence of comprehension of object oriented and component based software engineering.*	Program, output, report
2	*Discern design patterns in systems, critically analysing and producing designs based on these patterns.*	Program, report
3	*Design and implement a system by exploiting patterns and relevant object-oriented software engineering methods and explain and justify the approaches used*	Program, output, report

Assessment Criteria	Contributes to Learning Outcome	Mark
Report content – analysis, explanation of design and logical use of patterns	1,2,3	50
Extent to which submitted program executable fulfills requirements.	3	20
Program implementation – readability, structure, organization and design	1,2,3	30

FEEDBACK

Feedback will be given via:	Canvas feedback	Feedback will be given via:	
Exemption (staff to explain why)			
Feedback will be provided no later than 4 ‘teaching weeks’ after the submission date.			

This assessment is set in the context of the learning outcomes for the module and does not by itself constitute a definitive specification of the assessment. If you are in any doubt as to the relationship between what you have been asked to do and the module content you should take this matter up with the member of staff who set the assessment as soon as possible.

You are advised to read the **NOTES** regarding late penalties, over-length assignments, unfair means and quality assurance in your student handbook, which is available on Canvas.

In particular, please be aware that:

- Up to and including 24 hours after the deadline, a penalty of 10%
- More than 24 hours and up to and including 7 days after the deadline; either a penalty of 10% or the mark awarded is reduced to the pass mark, **whichever results in the lower mark**
- More than 7 days after the deadline, a mark of zero is awarded.
- The overlength penalty applies to your written report (which includes bullet points, and lists of text. It does not include contents page, graphs, data tables and appendices). 10-20% over the word count incurs a penalty of 10%. Your mark will be awarded zero if you exceed the word count by more than 20%.

Please be reminded that you are responsible for reading the University Code of Practice on Academic Misconduct through the Assessment section of the Quality Handbook (via the SharePoint site). This govern all forms of illegitimate academic conduct which may be described as cheating, including plagiarism. The term ‘academic misconduct’ is used in the regulations to indicate that a very wide range of behaviour is punishable.

In case of any subsequent dispute, query, or appeal regarding your coursework, you are reminded that it is your responsibility to produce the assignment in question.

The assignment

This assignment requires you to design and create a software application. You will be expected to use the patterns and principles we have considered in the lectures and labs, and to explain how and why you have done so. The design of your software is more important than its ultimate performance and functionality (as you will see from the allocation of marks). The software specification is created to give you the opportunity to use appropriate design patterns, and you will be expected to report on how you have done so. The content of the report is an important part of this assessment.

The software

The software you are to produce will be used to encrypt text using a number of different encryption algorithms. Encryption takes a an input string (called the plaintext) and transforms it into a different form (the ciphertext). In some cases, a key is used to configure the encryption. Decryption processes the ciphertext to retrieve the plaintext. In symmetrical encryption, the same key is used for encryption and decryption. In asymmetrical encryption, different keys are required for encryption and decryption (you will not be implementing this in this assignment).

There are three algorithms which your code should implement: simple substitution, MD5 and TEA. You will find many resources online to help you understand these, and there are papers listed in the table of references which give full details.

The simple substitution algorithm

This is a simple and wholly insecure method of encrypting text. Each alphabetical character in the string is repaced with one which is offset from it in the alphabet by *n* characters, where *n* is the key value. Thus if *n* is 5, the string "`Hello World!`" becomes "`Mjqqt Btwqi!`". Notice how the letter "W" has been encrypted to "B", because if the length of the alphabet is exceeded the index rolls over to the start of the alphabet again. Decryption is a reverse of this process.

The MD5 hash algorithm

The message digest algorithm MD5 is an irreversible encryption algorithm. It calculates a fixed-length hash value from any length of input data. This hash value cannot be converted back to the original text. The MD5 algorithm is described fully in (Rivest 1992). MD5 hashes have been used to ensure the integrity of data, because any change in the original data will have a high probabability of resulting in a different hash[1].

The TEA algorithm

The Tiny Encryption Algorithm (Wheeler and Needham 1995) was developed as a simple symmetrical encryption algorithm which required only limited resources. It was quickly shown to have weaknesses, and has been replaced with more secure versions (XTEA and XXTEA). It is not considered secure enough for serious protection, but has the benefit of being straightforward to implement. The key is 128-bits long, which can represent a UTF-8 string of 16 characters. Shorter strings should have space characters appended. Longer key strings should be truncated. TEA uses the same key for encryption and decryption.

[1] MD5 has been shown to be vulnerable to certain attacks, and is no longer considered suitable as a serious security measure, though it is useful for detecting accidental corruption of data. Better hash algorithms are available.

Your software task

You must write a program which implements the three algorithms previously mentioned. Your program should be a console application (no GUI) written in C#. It must run on a PC under Windows 10. It must have the name "crypto.exe".

Your program should accept a range of command-line arguments as described in the table below. Not all of them need to be used at the same time (and, as should be obvious, some of them cannot be used at the same time). Note that if neither -o nor -O options are specified, output should go to the console.

Your program must produce no output other than that described in the table unless the "d" flag is provide on the command line (in which case you may output whatever information you find helpful for development to the console). The application must not wait for a key press before terminating. No user input other than that on the command line may be used to operate it.

Flag and parameter	Purpose
-e	encrypt input
-d	decrypt input
-a SUB|MD5|TEA	Use the relevant algorithm
-s <integer>	If using SUB, specify the character offset for substitution
-k <key>	If using TEA, specify the key string for encryption/decryption
-f <filename>	read data from <filename> as the input
-o <filename>	Write the output to file <filename> only
-O <filename>	Write the output to both the console and file <filename>
-i "your text string"	use the provided string as input
-d	Output debugging information to the console. This is optional.

The following shows some examples of usage:

`crypto.exe -e -i "This is a test" -a TEA -p "John Suchet"`

Encrypts the text 'This is a test' using the TEA algorithm with the key phrase "John Suchet" and outputs the result to the console

`crypto.exe -e -f "c:\temp\file1.txt" -a SUB -s 2`

Encrypts the text in the file c:\temp\file1.txt using the the substitution algorithm with an offset of 2 and outputs the result to the console

`crypto.exe -d -f "c:\temp\file1.txt" -a TEA -k "shibboleth" -o "c:\temp\outfile.txt"`

Decrypts the text in the file c:\temp\file1.txt using the the TEA algorithm with a key phrase of "shibboleth" and writes the result to the file c:\temp\outfile.txt

`crypto.exe -e -i “stop the count” -a MD5 -O “c:\temp\outfile.txt”`

> Calculates the MD5 hash of the text “stop the count”, writes the result to the file c:\temp\outfile.txt and also to the console.

Constraints

Input plaintext must be in UTF-8 format, as must key phrases. Encrypted output must be Base64 encoded. You should assume that encrypted data read from files or the command line will also be Base64 encoded. You are encouraged to use the built-in libraries and conversions in .NET to handle Base64 and UTF-8 encoding.

You may not use cryptographic algorithm implementations in other libraries for TEA or MD5. This includes Microsoft’s own implementations. You may not directly use or copy open source implementations of the encryption algorithms.

Submission and Assessment

Write a report giving a description of your program. Explain the architecture and design of the program. Be clear about which design patterns you have used, why you have used them and how they are implelented. Very short program fragments may be included to illustrate the description. The report must be a pdf document file and be submitted to the main assignment. Do not exceed the word count specified above. Your software solution must be submitted to a separate assignment.

Note that although you are expected to produce some working software, its functionality is not the most important thing. Your program output will be checked against various tests, and the code will also be examined. You should make sure that it is readable and appropriately commented. You must submit a Visual Studio solution for your software, compressed into a single ZIP file. It must be possible to compile the solution with a standard university installation of Visual Studio.

Report

References

Rivest, R., 1992. *The MD5 Message-Digest Algorithm*. URL https://tools.ietf.org/html/rfc1321#section-3.4

Wheeler, D.J., Needham, R.M., 1995. *TEA, a tiny encryption algorithm*, in: Preneel, B. (Ed.), Fast Software Encryption, Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 363–366

These papers are available for download on the module’s Canvas site.