Linnaeus University
Department of Computer Science
OF

2021-11-16

Encryption

Practical work #1, 1DV700, HT21

In this practical work you are to investigate how to implement different encryption algorithms and test how resilient they are to crypto analysis.

This practical work should be done individually. Read all related information in MyMoodle before you start the work and use the uploaded template for your report.

The report you write for this practical work should include your results on all the tasks. None of the tasks should be left empty. All students are required to try to solve every task. Even if the task was not in the form of a question, you need to explain what you have done for it and which steps you have taken.

Your answers to the questions should elaborate on the answers. For instance, when you write about your program you should describe the algorithms you have implemented and how the program works. For task 5 you should explain the tools or methods used and what worked as well as what didn't work during the crypto analysis.

Do not copy the text from the tasks, or questions in your report. Just include your solutions. Make sure to structure your report the same as task numbers. Save the report in pdf format. Make a zip archive with report and source code for the program and post it in the upload folder assigned before the deadline.

1. The first task is to investigate different terms within cryptography and related areas.
a) What are the differences between the following pairs of methods;
Symmetric encryption – Asymmetric encryption
Encryption algorithms – Hash algorithms
Compression – Hashing
b) What are the differences between Steganography (read document about steganography in MyMoodle), Encryption and Digital Watermarking? What is the purpose of each method and when are they used?

2.
a) Try to decrypt the message "HKPUFCMHY BHDDXZH" knowing only that is a simple substitution cipher. If you give up do it with the help of this substitution key (cipher line).

plain	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
cipher	X	G	P	Y	H	Q	Z	I	R	A	J	S	B	K	T	C	L	U	D	M	V	E	N	W	F	O

b) Here is another short message encrypted with simple substitution but with another (unknown) key; "QMJ BPZ B XPJZ RZWJPAXQ LAD". Can you decrypt this message?

c) How difficult is it to decrypt the messages without having the key? Motivate!

3. Write a Python (or Java) program, implementing encryption/decryption without using any special security packages. The program should ask the user for encryption method, if you want to perform encryption or decryption, secret key and a text file to process. The output should be a processed file. You should implement at least two simple encryption methods, one substitution and one transposition method. For the substitution method, keep key sizes at maximum equivalent to eight bits, i.e. 2^8=256 different possible keys that corresponds to one character. Make sure that you can both encrypt and decrypt files with your program.

4. Download the file "plaintext.txt" found on the course home page. Add a secret message at the end of the file, at least one page long, and the names of the student that created the file. Encrypt the updated file using your program (you may select any method you have implemented). Post the encrypted file in the folder "Cipher texts". The name of the file should be the name of the student uploading the file.

5. In the folder mentioned above you will find cipher texts from the other students. Download some of these and try to perform crypto analysis on them. You may use any tool or method to perform this task. When you have successfully analysed at least one of the files, include a description of how you did it in the report. (Extra credits if you can decrypt several files that implemented different encryption algorithms).

6. A good hash functions should have several properties. See https://en.wikipedia.org/wiki/Hash_function#Properties

a) Create a very simple hash function and implement it in Python (or Java). The hash function should be able to accept input of different sizes and produce a hash value that is 8 bits.

b) Implement statistical tests of your hash function to test if it seems to be a good hash function. You should at least run tests for Uniformity and that small changes in input still give very different hash values. Consider using matplotlib to make it easy to analyse the result.

Uniformity can be tested with random input strings (both in terms of size and content). You should test with at least a couple of thousand test strings.

To test the second property, you should again run at least 1000 tests with input strings that are very similar (only differ in one bit) and investigate what distribution you get now for the hash values.

You can analyse the results manually or using different statistical tools, but you need to discuss how well your function meet the different properties.

c) There is a difference between normal hash functions and secure has functions. What is that difference? Prove that your hash function is not a secure hash function. What is the easiest way to prove this?