SCHOOL OF COMPUTER SCIENCE
COURSEWORK ASSESSMENT PROFORMA

MODULE & LECTURER: CM3110 Security, George Theodorakopoulos

DATE SET: 30 October 2017

SUBMISSION DATE: 4 December 2017, 9:30am

SUBMISSION ARRANGEMENTS: Online, via Learning Central

TITLE: Security Coursework

This coursework is worth 30% of the total marks available for this module. The penalty for late or non-submission is an award of zero marks. You are reminded of the need to comply with Cardiff University's Student Guide to Academic Integrity. Your work should be submitted using the official Coursework Submission Cover sheet.

INSTRUCTIONS

Your task in this coursework is to design and implement a cryptographic attack against a set of provided ciphertexts, as per the detailed instructions given later in this document.
You should submit one file with the code that you write for the implementation, and a single pdf report that describes the design and the results of the attack, including a brief justification of why you believe the attack should be effective, i.e. why you chose this design. The report should provide clear instructions to someone who wishes to perform the attack and reproduce your results.
Your code should be in Python 3. If you want to use a different language, please consult with the lecturer.
The report should be about 1000 words, with an **absolute maximum word limit of 1500 words**. Anything beyond the first 1500 words will not be marked.

SUBMISSION INSTRUCTIONS

Submit the following files (1 .pdf cover sheet, 1 .py file, 1 .pdf report) as separate files, i.e. **not** in a .zip file. Optionally, you can merge the cover sheet and the report pdfs into a single pdf.

Description		Type	Name
Cover sheet	**Compulsory**	One PDF (.pdf) file	cover.pdf
Attack code	**Compulsory**	One Python source file (.py)	attack.py
Report	**Compulsory**	One PDF (.pdf) file	[student number]-report.pdf

CRITERIA FOR ASSESSMENT

This coursework is worth 30 marks. The report and the code will be assessed jointly (a single mark for both, as they are tightly interrelated).
Credit will be awarded against the following criteria:

1. Correctness of the design and implementation as compared to the specification
2. Quality (in terms of clarity and technical correctness, where applicable) of the arguments in the report
3. Efficiency of the implementation

Feedback on your performance will address each of these criteria as necessary. These criteria are equally weighted in terms of marks, but this equal weighting will be done within reason – e.g. if the implementation is very efficient but solves a completely different problem, the mark will be 0 for the whole coursework.

FURTHER DETAILS

Feedback on your coursework will address the above criteria and will be returned in approximately four (4) weeks.

This will be supplemented with oral feedback in the revision lecture.

Individual feedback will be provided on Learning Central. Further individual feedback will be available upon request.

CM3110 Security Coursework

You are given a file (`cipher.py`) with Python 3 code that implements a stream cipher. In this coursework, your task is to perform an attack against a set of ciphertexts that were encrypted with this cipher.

How the cipher code works

The cipher code takes as input

1. An opcode character 'e' (for encryption) or 'd' (for decryption)
2. A file that contains the key (ASCII string)
3. A file that contains the input text (plaintext or ciphertext)

and encrypts or decrypts the input as appropriate, printing the result to standard output.

Regardless of whether it encrypts or decrypts, the cipher first converts the key from an ASCII string to a list of **bytes** (integers between 0 and 255 inclusive). This key (list of bytes) is then used to generate a keystream of bytes.

In encryption mode (opcode 'e'), each keystream byte is XORed with a plaintext character that has been converted to an integer using the `ord()` function. The resulting ciphertext integer is converted to **two** hexadecimal characters (0123456789ABCDEF), which are then written out.

In decryption mode (opcode 'd'), each keystream integer is XORed with a ciphertext integer that corresponds to a **pair** of hexadecimal characters. The resulting integer is converted to a plaintext character using the `chr()` function. The output is written out as an ASCII string.

Example (encryption)

Assume you execute

```
python cipher.py e key.txt plaintext.txt
```

and the contents of the files are:

`key.txt:`
Secret key

`plaintext.txt:`
Transfer 100GBP to account 1234.

Then, the correct output is
4E40B844800742DB4F9B8879E75727B244988FA23854AF93F3252EE07935FBB4

Attacking the stream cipher

You have intercepted multiple ciphertexts (file `ciphertexts.txt`, one ciphertext per line), each of which is English text that has been encrypted with the stream cipher in `cipher.py` using the same key. The key is unknown to you. Due to an error in the configuration of these multiple encryption sessions, the key stream generator was reseeded and restarted afresh for **two** of these ciphertexts.

Your task is to design and implement an attack that will identify the two mis-encrypted ciphertexts and will recover as much information as possible about the two corresponding plaintexts. In your report you should describe the design of the attack, and you should state clearly any information about the two plaintexts that you are able to recover. The following questions should help you design the attack, but you are not required to provide an explicit answer to them in your report:

1. In the One-Time Pad cipher, why is it a mistake to encrypt two plaintexts with the same key?
2. In terms of randomness, how does the XOR of two English plaintexts differ from the XOR of two correctly encrypted ciphertexts?
3. In a cryptographic context, what is “crib-dragging”?

Notes:

1. It is **not required** that your attack be completely automated. In other words, part of your attack could be done in software, but you may also need to perform some steps manually.
2. It is **not required** that your attack recover the two plaintexts completely. There is an element of randomness in how much you can recover.
3. Your report should give **clear instructions** to someone who wishes to perform the attack and reproduce your results. In other words, describe how one can use your submitted code together with some manual steps to perform the attack.
4. You are free to use all or part of the code in the Python file provided to you (`cipher.py`).