University of London

Computing and Information Systems/Creative Computing

CO3326 Computer security 2018-19

Coursework assignment 1

IMPORTANT: all students have been allocated a unique set of cipher text, intercepted key stream fragment and seed to use for this coursework assignment. You can obtain this using your Student Reference Number (SRN) from the following URL: http://foley.gold.ac.uk/cw19/api/cw1/{srn}. For example, if your SRN is 877665544, you would obtain your data from http://foley.gold.ac.uk/cw19/api/cw1/877665544. If you have difficulties obtaining your cipher text, intercepted key stream fragment and seed, please email us at: intcomp@gold.ac.uk

This coursework assignment is designed to help you enrich your learning experience and to encourage self-study and creativity. Chapter 5 (pages 49-62) of the subject guide, including the suggested supplementary reading, is a good starting point and will help you in completing this assignment. You should read the coursework assignments carefully and pay particular attention to the Submission requirements.

You are expected to submit **two** files: a **report** and a **results sheet**. The *report* counts as **60%** of your coursework assignment mark, in which you are expected to answer the questions below. The *results sheet* counts as **40%** of your mark, in which you are expected to summarise the results of your calculations in a specific format. Please use the cipher text and keys provided when answering the questions and when compiling the results sheet.

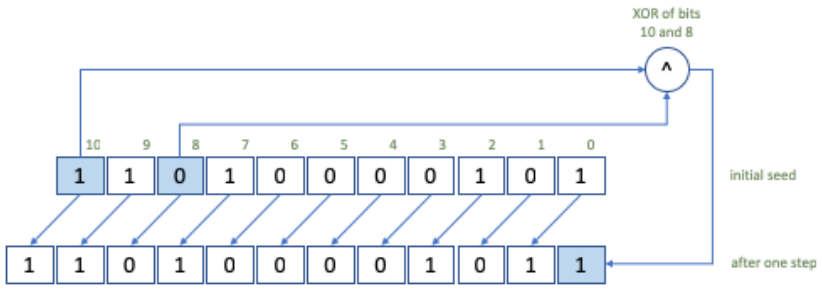
To complete the coursework assignment, it will make your life easier if you write a program. You are welcome to use any programming language. Please include key snippets of your code as an appendix at the end of your report.

Linear Feedback Shift Register

A *linear feedback shift register* (LFSR) is a type of counter that generates a "pseudo random" sequence of bits, which is particularly useful in cryptography. In simple terms, it is a register of bits that performs discrete step operations that:

- shifts all of the bits one position to the left, and
- replaces the vacated bit by the *exclusive or* (XOR) of certain bits shifted off, indicated by the *tap* positions.

A LFSR has three parameters that characterise the sequence of bits it produces: the number of bits N, the initial *seed* (the sequence of bits that initialises the register), and the the tap positions *tap*. The following picture illustrates one step of an 11-bit LFSR with initial seed 11010000101 and tap positions 10 and 8.



One step of an 11-bit LFSR with initial seed 11010000101 and tap positions at bits 10 and 8

Figure 1: One step of an 11-bit LFSR with initial seed 11010000101 and tap positions at bits 10 and 8

Please do some research around LFSRs, starting from your subject guide (Chapter 5), suggested readings and Wikipedia. Make sure you also understand the Berlekamp-Massey algorithm as this is necessary for you to complete the coursework assignment.

Assignment

Suppose you are an avid hacker working for a security agency and you intercept a *cipher text*, a *key stream fragment* and a *seed* that you know has been sent using a cryptosystem based on LFSRs. You would like to decrypt the message.

IMPORTANT: To answer the questions below, please use the *cipher text*, intercepted *key stream fragment* and *seed* that you obtained using your SRN.

Question 1 Briefly describe how stream ciphers work and give a few reasons why stream ciphers based on linear-feedback shift registers are so popular.

Question 2 Explain the following terms in the context of LFSRs: *feedback function*, *primitive polynomial* and *cycle*. What are *maximal-length polynomials*?

Question 3 Explain why the use of LFSRs on their own is insufficient to provide good security. List three schemes that have been proposed to increase the security of LFSRs.

Question 4 Briefly describe, in your own words, what the Berlekamp-Massey algorithm is used for and how it works.

Question 5 Apply the Berlekamp-Massey algorithm on the key stream fragment you were given to obtain the *feedback polynomial* that generated the key. For full marks, you have to implement the algorithm yourself, attach key parts of the algorithm to the report as an appendix and describe briefly which bit of the implementation you found the most challenging. If you use someone else's code or an online service you will not be awarded full marks; in this case make sure you acknowledge the source of the code or service you use.

Question 6 Generate the full key stream in order to be able to decrypt the cipher text you intercepted (*i.e.* what you were given).

Question 7 Decrypt and decode the cipher text to get the original plain text message. Show all your workings. If your calculations are correct, you will get an English dictionary word as the plain text.

Question 8 Briefly - in one paragraph - describe the design of your code. Attach key snippets to the annex. Do not forget to acknowledge all sources. Make sure you acknowledge any code re-use.

Submission requirements

REMINDER: It is important that your submitted coursework assignment is your own individual work and, for the most part, written in your own words. You must provide appropriate in-text citation for both paraphrase and quotation, with a detailed reference section at the end of your coursework. Copying, plagiarism and unaccredited and wholesale reproduction of material from books or from any online source is unacceptable, and will be penalised (see our guide on how to avoid plagiarism on the VLE).

You should upload **two** single files only. These must not be placed in a folder, zipped, *etc.*

The **report** should be submitted as a PDF document following a *strict naming scheme*: `YourName_{srn}_CO3326_cw1.pdf`. For example, Steve Jobs with SRN 877665544 would submit `SteveJobs_877665544_CO3326_cw1.pdf`.

The **results sheet** should be submitted as a JSON file with a *strict format* and *strict naming scheme*. This summarises the results of your calculations and will be automatically checked by an algorithm, so pay particular attention to its format. The name of the file should be `YourName_{srn}_CO3326_cw1.json`; for example, Steve Jobs with SRN 877665544 would submit `SteveJobs_877665544_CO3326_cw1.json`.

Example

You have obtained the *cipher text*, intercepted *key stream fragment* and *seed* in the following format (this is an example for illustration):

```
{
  "srn": "877665544",
  "name": "Steve Jobs",
  "lfsr": {
    "seed": "11010000101"
  },
  "keyFragment": "10100100001001101001011",
  "cipherText": "a5dc26183f945cbb6732"
}
```

For this *cipher text*, intercepted *key stream fragment* and *seed*, Steve Jobs would submit the following JSON, which reflects a correct solution:

```
{
  "srn": "877665544",
  "name": "Steve Jobs",
  "lfsr": {
    "taps": [
      10,
      8
    ],
    "seed": "11010000101"
  },
  "keyFragment": "10100100001001101001011",
  "key": "d0b24f6e5ae62fd2134b",
  "plainText": "university",
  "cipherText": "a5dc26183f945cbb6732"
}
```

Explanation

The *srn* and *name* fields are self-explanatory. The binary *keyFragment* field has been intercepted (given). The Berlekamp–Massey algorithm can be used to compute the *tap positions*, which is expected to be under the *lfsr* field in the *taps* subfield, which is an array of zero-based indices in decreasing order (*i.e.* [10, 8]). Once the *tap positions* and the *linear span* of the LFSR are known, the entire *key* can be generated starting from the *seed*, which has been intercepted (given). The *key* is the full-length key stream in hexadecimal that is necessary to decrypt the *cipherText*. The corresponding binary value for the *key* in this case is:

```
01101000010110010010011110110111001011010111100101101011110011000101111111010010000100110100101
```

The *cipherText* is also given in hexadecimal, so this needs to be converted to binary before the *key* can be applied on it, *i.e.* a simple *exclusive or* (XOR) operation, to get the encoded plain text. The binary encoded plain text can then be decoded to obtain the actual *plainText*, which is a recognisable English dictionary word.

You should notice the following:

- The intercepted *cipherText* is in hexadecimal and is 10 bytes long, exactly the same length as the generated *key stream*.

- The *key stream* starts with the given *seed*.
- The *key stream* contains the the *keyFragment*.

All strings are encoded/decoded using UTF-8. There are UTF-8 encoders/decoders available for all programming languages. If you have difficulties decoding, you can double-check your results with the following Web API call: http://foley.gold.ac.uk/cw19/api/decode?binary=11101010110111001101001. This decodes as **uni**. You can call this with any binary number.

Final note

You can use the example solution above as a template, which is a well-formed JSON, and replace the values with your data and calculation results. As the JSON will be evaluated by an algorithm, every quote, comma, colon, curly brace upper/lower case is crucial. Please pay attention to these. It would be a shame to lose a potential **40%** of the total marks for this coursework assignment because of a misplaced comma or a missing quote. There are online tools you can use for JSON formatting and validation (for example https://jsonformatter.curiousconcept.com), so double-check that your JSON is syntactically correct.