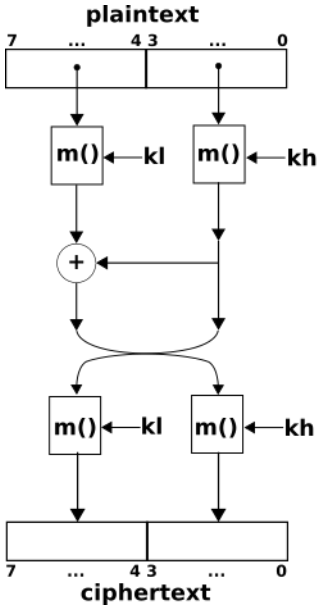
# CMPUT 333, Assignment 1, Fall 2015

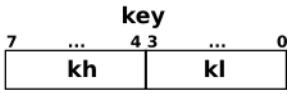
(weak encryption, cipher modes, weak passwords)

## Weak Encryption & Cipher Modes (75%)

### Part 1 (25%)

You are given an encrypted file (`ciphertext1`). The file was encrypted using a variation of the Vigenère cipher. The plaintext is an ASCII file. The key can be a combination of upper and lower case characters and numerals (no punctuation marks or any other special characters). The code is not a "textbook" Vigenère code, but rather a slightly more complicated version with respect to how a single plaintext byte and a single key byte are combined to produce a ciphertext byte. That is, the way the encryption takes place between a single plaintext byte, and a single key byte, to produce a single ciphertext byte, is shown in the diagram.




The "addition" operator in the circle is the bit-wise XOR operation. The block **m(*x*,*k*)** where ***x*** is the value fed from the top and ***k*** is the value fed from right, is implemented as a lookup table map[***x***][ ***k***]. Note that both ***x*** and ***k*** are four bit quantities. The lookup table contents are:

```
map[16][16] = {
{0x7, 0x5, 0x0, 0x4, 0x2, 0x3, 0xb, 0x6, 0xa, 0x8, 0x9, 0xd, 0xc, 0xf, 0xe, 0x1},
{0x3, 0x8, 0xd, 0xa, 0xc, 0xe, 0xf, 0xb, 0x7, 0x6, 0x4, 0x5, 0x1, 0x2, 0x0, 0x9},
{0x4, 0x0, 0x3, 0x1, 0xb, 0xa, 0x8, 0x5, 0x9, 0xd, 0xc, 0xe, 0xf, 0x6, 0x7, 0x2},
{0x9, 0xe, 0x7, 0xc, 0x6, 0x4, 0x5, 0xd, 0x1, 0x0, 0x2, 0x3, 0xb, 0x8, 0xa, 0xf},
{0x1, 0x3, 0xa, 0x2, 0x8, 0x9, 0xd, 0x0, 0xc, 0xe, 0xf, 0x7, 0x6, 0x5, 0x4, 0xb},
{0xe, 0x6, 0x5, 0x7, 0x1, 0x0, 0x2, 0xf, 0x3, 0xb, 0xa, 0x8, 0x9, 0xc, 0xd, 0x4},
{0x2, 0xa, 0x9, 0xb, 0xd, 0xc, 0xe, 0x3, 0xf, 0x7, 0x6, 0x4, 0x5, 0x0, 0x1, 0x8},
{0xd, 0xf, 0x6, 0xe, 0x4, 0x5, 0x1, 0xc, 0x0, 0x2, 0x3, 0xb, 0xa, 0x9, 0x8, 0x7},
{0xb, 0x9, 0xc, 0x8, 0xe, 0xf, 0x7, 0xa, 0x6, 0x4, 0x5, 0x1, 0x0, 0x3, 0x2, 0xd},
{0x0, 0xb, 0x8, 0x3, 0x9, 0xd, 0xc, 0x2, 0xe, 0xf, 0x7, 0x6, 0x4, 0x1, 0x5, 0xa},
{0x8, 0xc, 0xf, 0xd, 0x7, 0x6, 0x4, 0x9, 0x5, 0x1, 0x0, 0x2, 0x3, 0xa, 0xb, 0xe},
{0x5, 0x2, 0xb, 0x0, 0xa, 0x8, 0x9, 0x1, 0xd, 0xc, 0xe, 0xf, 0x7, 0x4, 0x6, 0x3},
{0x6, 0x1, 0x2, 0x5, 0x3, 0xb, 0xa, 0x4, 0x8, 0x9, 0xd, 0xc, 0xe, 0x7, 0xf, 0x0},
{0xc, 0x7, 0x4, 0xf, 0x5, 0x1, 0x0, 0xe, 0x2, 0x3, 0xb, 0xa, 0x8, 0xd, 0x9, 0x6},
{0xa, 0xd, 0xe, 0x9, 0xf, 0x7, 0x6, 0x8, 0x4, 0x5, 0x1, 0x0, 0x2, 0xb, 0x3, 0xc},
{0xf, 0x4, 0x1, 0x6, 0x0, 0x2, 0x3, 0x7, 0xb, 0xa, 0x8, 0x9, 0xd, 0xe, 0xc, 0x5}};
```

Your tasks are to:

- Show the plaintext corresponding to the ciphertext given, and the key that was used to encrypt it.
- Explain in your submission what technique(s) you used to determine the key and why.
- Explain how you automated (if you did) the process of recognizing whether you had the right key? (If you did not automate it, or automate it completely, explain why not.)

- Submit any source code you write to solve this problem. You are free to use the language of your choice, but C is preferred. Regardless of your choice of language, you must also submit a `README.txt` file explaining how to compile/run your code on the lab machines.
- Answer this question: if the plaintext was first compressed using one of the standard compression tools (zip, gzip, etc.) before it was encrypted, how would your strategy have changed? Explain your answer.

## Part 2 (25%)

You are given another encrypted file (`ciphertext2`) encrypted using the same scheme as before but using a substantially longer key than the first one. You know that the format of the plaintext file is a format used in a number of word processing applications.

Your tasks are to:

- Determine the plaintext corresponding to the ciphertext given, and the key that was used to encrypt it.
- Explain in your submission how you modified the technique you used in Part 1 to solve Part 2.
- Explain how you automated (if you did) the process of recognizing whether you had the right key? (If you did not automate it, or automate it completely, explain why not.)
- Submit any additional source code you write specifically to solve this problem. You are free to use the language of your choice, but C is preferred. Again, provide in the accompanying `README.txt` a description of the steps necessary to compile and run your code on lab machines.

In your answers make sure to comment on the benefits of known (partial) plaintext. Identify the points where the known plaintext structure helps you in the process of decrypting the ciphertext and recovering the key.

## Part 3 (25%)

In this part of the assignment you will experiment with the various cipher modes of DES. You should review the definition of ECB, CBC, CFB, and OFB modes of operation from the textbook.

- Create a `plaintext` file consisting of 10 repetitions (without any whitespace between them) of the (ASCII character) string: `01234567` That is, your file length should be 80 characters long *exactly*.
- Encrypt the file with DES using the following command: "`openssl enc -e -des-XYZ -nosalt -in plaintext -out cipherXYZ.enc`" where XYZ is `ecb`, `cbc`, `cfb`, `ofb`. In total you have produced four new ciphertext files. You are free to choose any encryption key you like (but you have to use the same key for all encryptions) as long as you submit it with your assignment. All the ciphertext files should also be submitted.
- Inspect the contents of the `.enc` files. Explain the differences (i) with respect to their sizes vs. the plaintext files, and (ii) with respect to seeing any patterns in their contents.
- Now create an "error" version of each file (name these versions "`cipherXYZerror.enc`" following the previous convention) by replacing the 19th byte in each file by a different byte. These files will also be submitted.
- Try now to decrypt each of the "`cipherXYZerror.enc`" and write your explanation about what was the impact of the "error" on the decryption outcome.

Note that explanation is not the same as observation. In responding to all the questions, you must refer back to the precise aspect of the ECB, CBC, CFB or OFB mode operation that caused the behavior you observed.

- Finally, (this part is worth 15%) you are given an encrypted file (`ciphertext3`), which you know has been encrypted using DES in ECB mode (using `openssl`, with `–nosalt` option). You also know that the

plaintext is a list of student records where (for each record) the first 8 characters are the student ID, and the remaining 24 bytes are the student name. At the end, after all the records, there is a single character which is the ASCII value for the record number of the student who was selected to receive a scholarship (that is, it is ASCII '1' if the first record/student received the scholarship, it is ASCII '2' if the second receiceived the scholarship, and so). Answer clearly and precisely the following questions:

- How many student records are in the list? How did you determine this?
- Modify the file such that the IDs of the student in the first record and in the second record are exchanged. Provide the modified file as part of the deliverables.
- Which record/student in the list received the scholarship? You are given the freedom to ask us (in a single request) to produce the ciphertext for up to 8 files of your choice (provide them as files to us) and you can then base your answer on this information. In any event, describe the process you used (even if it turns out to be unsuccessful).

# Password Cracking (25%) [Sliding Part]

Each group will be provided separate password hashes files from Unix/Linux and Windows/LANMAN.

- You are to crack as many passwords as possible from the files. We suggest that you employ at least a well known password cracking program, namely John the Ripper http://www.openwall.com/john/ but you are free to find and use additional tools if you feel it is necessary. Clearly, if you need to use a different tool than the one recommended, you have to indicate why it was a better choice. You have limited amount of time to break the passwords (until the Assignment 1 sliding deadline), so you are encouraged to use the best strategy you can come up with. Do not use any other University computers for cracking the passwords or any personal computers. This includes machines in other labs, the GPU login servers, etc.
- If you find your initial strategy not producing results (i.e., cracking passwords), you might want to consider the hints given and to study how you could provide more appropriate wordlists and rules to John the Ripper. Hints are given out in class.
- Your report must indicate which passwords you cracked (what was the plaintext of the passwords) and what kind of a strategy (if any) you followed.
- Explain why the cracked passwords were weak passwords and, additionally, compare and contrast the Windows and Unix passwords. Which ones were easier to crack and why?

In the interest of fairness, the release of the passwords will take place at a specific time and date, which will be pre-announced in class and/or via eclass.

# Deliverables

Only one of the group members need to submit on behalf of the entire group (in the event of more than one submission, the last one will be considered). The sliding part (and the sliding part only) can be submitted at any point in time prior to the deadline for the (non-sliding) part of Assignment 2. Your submission must include a report file (in .pdf format) which includes answers to the questions and should cite any resources that you used to answer the questions. It is assumed that all group members equally contribute to the assignment but you have to provide a paragraph in your report which explains how you split the workload. If you need to deviate from this model ("all equally contributing") of cooperation, explain why and indicate who/what was responsible for what.

[Optional: add a single paragraph at the end of the report indicating whether you found any difficulties with this assignment and if you think there are ways in which it could be improved. In particular, we are interested to know if the assignment forced you to learn something new that you did not know of before, and how much effort it took you. Was the workload reasonable?]

**********