CPSC 418 / MATH 318 — Introduction to Cryptography
ASSIGNMENT 1

SET: **Monday, Sept. 17** Total marks: **100** + 10 bonus marks
DUE: **Wednesday, Oct. 10** at **11:55 PM**

*** * * Please read the entire assignment carefully! * * ***

Requirements. Problems 1-4 are required for both CPSC 418 and MATH 318. Problem 5 is required for MATH 318 only, but CPSC 418 students may do this problem for extra credit. Problem 6 is required for CPSC 418 only, but MATH 318 students may do this problem for extra credit. Problem 7 is a bonus problem for everyone.

Bonus Credit Policy:

- Bonus credit will only be awarded for perfect or near perfect solutions.
- The maximum number of bonus marks awarded is 25.

Originality. Group work is *not* permitted on any of the problems. If you consult with other students, make sure that the answers presented are written in your own words. Be sure to show all your work.

Format. Your answers to the written problems must be *typeset* and submitted in PDF format. For uniformity, please use the LaTeX template provided with the problem sheet on the "assignments" page for your solutions. Fill in your **name** and **student id number** in the appropriate spaces on the template. Solutions that do not use LaTeX (e.g. are hand-written or generated with a word processor such as MS Word or StarOffice) will be receive a 50 percent penalty.

Your written solutions to **Problems 1-5 and 7** must be submitted in one single file in PDF format; do *not* submit LaTeX source files. If you did any programming for any of these problems and wish to submit your computer code as supporting documentation, submit it in one or more separate files, but your written solutions must include accompanying explanations and documentation.

For **Problem 6**, submit the written part of your solution as a separate README file. This file may be in TXT format and need not be done in LaTeX. Do *not* include the written portion of the programming problem in the PDF file containing your written answers to the other problems.

Submission Procedure. Instructions for assignment submissions will be published on the "assignments" page on our course website shortly. Assignments must be submitted online by the specified deadline. **Late submissions will be severely penalized.**

For the sake of fairness and respect for our TAs' time and workload, please follow all these instructions *carefully* as assignments not conforming to the submission procedure will incur penalties.

Written Problems for CPSC 418 and MATH 318

Problem 1 — Superencipherment for substitution ciphers, 12 marks

(a) Recall that for the shift cipher, we have $\mathcal{M} = \mathcal{C} = \mathcal{K} = \mathbb{Z}/26\mathbb{Z}$ (the integers modulo 26). Encryption is given by $E_K(M) \equiv M + K \pmod{26}$ (modular addition of message and key).

  i. (2 marks) Give a formal mathematical proof that double encryption for the shift cipher under two keys K_1, K_2 results again in a shift cipher. What is the key of the double encipherment?

  ii. (5 marks) Give a formal mathematical proof that *superencipherment* (i.e. multiple encryption) for the shift cipher results again in a shift cipher. Use induction on the number of encipherments (solutions that do not use induction will be penalized). What is the key of the multiple encipherment?

(b) (5 marks) Recall that encryption under the Vigenère cipher just corresponds to m shift ciphers, one for each of the letters in the key word. Formally explain why encryption under a Vigenère cipher using a key word w_1 of length m, followed by encryption under a second Vigenère cipher using another key word w_2 of length n, is again a Vigenère cipher under some new key word w. How is w obtained, and what is its length? (*Note:* mn *is generally* *not* the correct answer for the length of the new key word w.)

Problem 2 — Key size versus password size, 21 marks]

ASCII, short for *American Standard Code for Information Interchange*, is a method for encoding any character into a string of 7 bits. For example, the ASCII codes for 'A', 'a', and '?' are 1000001, 1100001, and 0111111, respectively[1].

Consider a cryptosystem where the user enters a key in the form of a password.

(a) (3 marks) What is the total number of ASCII encodings of 8-character strings?

(b) Only 94 of all the ASCII character encodings are allowed in passwords; the rest correspond to characters that are either unusable (like line feeds, tabs, spaces) or don't appear on a standard North American keyboard.

  i. (3 marks) Assuming that passwords consist of exactly 8 characters, what is the total number of ASCII encodings of permissable passwords (i.e. the size of the key space)?

  ii. (3 marks) What percentage of all 8 character ASCII encodings are usable as passwords?

(c) (3 marks) Assuming that each permissable character in a password is chosen equally likely, what is the entropy of the key space of part (b) i?

(d) (3 marks) Suppose now that users use only the 26 lower case letters for their 8-character passwords instead of the full set of 94 permissable characters. Assuming again that each key consisting of lower case letters is chosen equally likely, what is the entropy of this restricted key space?

[1]Source: `http://www.ascii-code.com`.

(e) Suppose we want a keys space with entropy 128, i.e. a total of 2^{128} keys assuming that keys are chosen equally likely (this number is typical for modern cryptosystems). What is the minimum password length (i.e. the minimum number of characters in a password) that guarantees this key space size, assuming that passwords consist of

   i. (3 marks) ASCII encodings of permissable characters?

   ii. (3 marks) ASCII encodings of lower case letters?

Problem 3 — Equiprobability maximizes entropy for two outcomes, 12 marks

Let X be a random variable consisting of two outcomes X_1 and X_2, both occurring with positive probability $p(X_1) = p$ and $p(X_2) = 1 - p$. The *(Shannon) entropy* of X is defined to be

$$H(X) = p \log_2 \left(\frac{1}{p}\right) + (1-p) \log_2 \left(\frac{1}{1-p}\right) = -p \log_2(p) - (1-p) \log_2(1-p) \ .$$

Note that $H(X) > 0$ since p and $1 - p$ are both strictly between 0 and 1, so their reciprocals exceed 1, and hence $\log_2(1/p)$ and $\log_2(1/(1 - p))$ are both positive.

(a) (2 marks) Suppose $p(X_1) = 1/4$ and $p(X_2) = 3/4$. Numerically calculate $H(X)$.

(b) (8 marks) Prove that if $H(X)$ is maximal, then both outcomes are equally likely.
Hint: First year calculus: consider $H(X)$ as a function of p and determine for which value of p it takes on its maximum. Then find that maximum.

(c) (2 marks) What is the maximal value of $H(X)$?

Problem 4 — Conditional entropy, 12 marks

Let X and Y be two random variables. Recall that the joint probability $p(x, y)$ is the probability that $p(X = x)$ and $p(Y = y)$; it satisfies $p(x, y) = p(x|y)p(y)$. The *conditional entropy* or *equivocation* of X given Y is defined to be

$$H(X|Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log_2 \left(\frac{1}{p(x|y)}\right) = \sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log_2 \left(\frac{1}{p(x|y)}\right) \ ,$$

where the sums $\sum_{x \in X}$ and $\sum_{y \in Y}$ run over all outcomes of X and of Y, respectively, such that $p(x|y) > 0$. Informally, the equivocation $H(X|Y)$ measures the uncertainty about X given Y. (Shannon measured the security of a cipher in terms of the key equivocation $H(K|C)$, i.e. the amount of information about a key K that is not revealed by a given ciphertext C.)

(a) (5 marks) Consider a plaintext space $\mathcal{M} = \{M_1, M_2, M_3, M_4\}$, with corresponding ciphertext space $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$. Suppose that each plaintext and each ciphertext is equally likely, i.e. $p(M_i) = p(C_j) = 1/4$ for $1 \leq i, j \leq 4$. Now suppose that each ciphertext C_j narrows down the choice of corresponding plaintext M_i to two of the four possibilities as follows:

C_1: M_1 or M_2
C_2: M_3 or M_4
C_3: M_2 or M_3
C_4: M_1 or M_4

Compute $H(\mathcal{M}|\mathcal{C})$.

(b) (5 marks) Suppose a cryptosystem provides perfect secrecy, and $p(M) > 0$ for all $M \in \mathcal{M}$. Prove that $H(\mathcal{M}|\mathcal{C}) = H(\mathcal{M})$.

(c) (2 marks) Does the example of part (a) provide perfect secrecy? Explain your answer?

Written Problem for MATH 318 only

Problem 5 — Perfect secrecy and joint entropy, 43 marks

(a) (One-time pad provides perfect secrecy)

Recall that for the one-time pad, we have $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$, encryption is defined via $E_K(M) = M \oplus K$ (bit-wise x-or), and decryption is given via $D_K(C) = C \oplus K$.

i. (3 marks) Prove that one-time pad encryptions are bijections (i.e. one-to-one and onto). Use this result to describe the set $E_K(\mathcal{M})$.

ii. (6 marks) Prove that for any ciphertext C, the map $\phi_C : \mathcal{K} \to \mathcal{M}$ defined by $\phi_C(K) = D_K(C) = C \oplus K$ is a bijection. Conclude that

$$\sum_{K \in \mathcal{K}} p(D_K(C)) = 1 \ .$$

iii. (6 marks) Assume that each one-time pad key is chosen with equal likelihood. Use the definition

$$p(C) = \sum_{\substack{K \in \mathcal{K} \text{ with} \\ C \in E_K(\mathcal{M})}} p(K) p(D_K(C))$$

to prove that each ciphertext occurs with equal likelihood (regardless of the probability distribution on the plaintext space).

iv. (3 marks) Prove that for every plaintext M and ciphertext C, there exist exactly one key K such that $E_K(M) = C$, namely $K = M \oplus C$. Use this result to describe for any message M the set $\{K \in \mathcal{K} \mid E_K(M) = C\}$.

v. (2 marks) Assume again that each one-time pad key is chosen with equal likelihood. Use the results of parts (b) iii and iv as well as the definition

$$p(C|M) = \sum_{\substack{K \in \mathcal{K} \\ E_K(M) = C}} p(K)$$

to prove that the one-time pad provides perfect secrecy.

(b) (Joint entropy)

Let X and Y be two random variables. Assume for simplicity that $p(x) > 0$ for all $x \in X$ and $p(y) > 0$ for all $y \in Y$. Recall the connection between joint and conditional probability:

$$p(x, y) = p(x|y)p(y) \quad \text{for all } x \in X \text{ and } y \in Y.$$

You will need this formula in part ii.

i. (2 marks) Explain why $\sum_{x \in X} p(x|y) = 1$ for all $y \in Y$.

ii. (2 marks) Use part i to prove that $\sum_{x \in X} p(x,y) = p(y)$ for all $y \in Y$.

iii. (2 marks) Use part ii to prove that $\sum_{y \in Y} p(x,y) = p(x)$ for all $x \in X$.

iv. (6 marks) The *joint entropy of X and Y* captures the combined uncertainty of X and Y and is defined to be

$$H(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \left(\frac{1}{p(x,y)} \right) .$$

Prove that $H(X,Y) \leq H(X) + H(Y)$. Use parts ii & iii and (without proof) the fact that $p(x,y) \geq p(x)p(y)$ for all x, y. (Informally, this result says that the total uncertainty of X and Y can only decrease if X and Y occur together.)

v. (3 marks) Prove that equality holds in part iv if and only if and only if X and Y are independent, i.e. x and y are independent for all outcomes $x \in X$ and $y \in Y$. (Informally, this result says that if X and Y are independent, then the fact that they occur together does not reduce their total uncertainty.)

vi. (6 marks) Use part ii to prove that $H(X,Y) = H(X|Y) + H(Y)$, where $H(X|Y)$ was defined in Problem 4.

vii. (2 marks) Combine the results of parts vi and iv to prove that $H(X|Y) \leq H(X)$. (Informally, this result says that the uncertainty of X can only decrease when some side information Y is known.)

Programming Problem for CPSC 418 only

Problem 6 — File Encryption and Authentication Implementation, 43 marks

Your solution to this problem must be implemented in `Java`. Make sure to use good coding practices. You may use whatever development platform you like, but make sure that the final version compiles and runs on one of the Computer Science department Linux servers. The TA will compile and test your programs on one of these servers, using the latest version of `javac/java` installed.

Suppose you wish to send a file to your friend by e-mail. You don't want anyone else to know the content of the file. At the same time, your friend wants to make sure that the received file is identical to the one that you sent, and that no third party altered the encrypted file. One way to accomplish this is to calculate a *message digest* of your file and append it to the file before encrypting the file along with its message digest. That is, you send

$$C = E_k(\mathit{file} \| \mathit{digest}) ,$$

to your friend. Here, E_k is encryption under key k, *file* is the file you wish to send, *digest* is its message digest, and $\|$ denotes concatenation with some encoding as described below. When your friend receives C, he or she decrypts it, separates the file from the message digest, and verifies that the message digest is correct. If the the digest is not correct, then C was modified in transit.

Design and implement a Java program using the latest version of the *Java Cryptography Architecture* (JCA) to encrypt a file of arbitrary length with its digest appended. You may assume that your input file is at most 1MB in size, so you can simply read the file content into an array and feed it into the encryption function. Use AES-128 as your encryption algorithm and a message authentication procedure of your choice. Also implement the appropriate decryption and message digest verification program. Use CBC as the mode of operation of your encryption and decryption algorithm.

A link to the JCA documentation can be found on the "references" page of the course website. The JCA has its own interface to which you are expected to adhere. Remember that the encrypted file contains the original file concatenated with its digest. Therefore, you should *encode* the file with its digest in such a way that the receiver can separate them after decrypting. Your decryption and digest verification program should output an error if the encrypted file was modified from its original. The TA will test this using `diff` and `cmp` commands.

To generate a key for your encryption algorithm, consider the user input as a random seed and use a *pseudorandom number generator* (PRNG) to convert the seed to a random key of the appropriate bit length as determined by the encryption algorithm. The key is passed to the encryption and decryption routine. Users should be able to enter an ASCII character string of any length as the seed. You may use any PRNG you like, but you must specify which one you use and its origin (JCA, Java built-in, third party source etc).

The TA will test your program with 3 type of files: a text file, a JPEG file and a ZIP file. After performing encryption, they may change some bits of the output file randomly to make sure that your program can detect unauthorized changes to the encrypted file. (*Friendly hint:* this gives you some suggestion as to how to test your program.)

Your encryption and message authentication program should be invoked by the command

```
java secureFile [plaintext-filename] [ciphertext-output-filename] [seed]
```

Your decryption and digest verification program should be invoked by the command

```
java decryptFile [ciphertext-filename] [plaintext-output-filename] [seed]
```

Programs that do not comply with these specifications will be penalized or not marked at all.

You need *not* submit paper printouts of your source code, but you must submit a description of your implementation in a *separate README file* (this file may be in TXT format and need not be done in LaTeX). **Do *not* include the written portion of the programming problem in the PDF file containing your written answers to the other problems.**

Your description must include:

- A list of the files you have submitted that pertain to the problem, and a short description of each file. In particular, specify the following:
  - which message authentication algorithm you are using

  - the type and source of your PRNG
  - your method for encoding the input to the encryption algorithm; that is, how the *file* is separated from the *digest*.
- A description of how to compile and test your program (we prefer makefiles), and the name of the department server on which you tested your code. Again, programs that do not compile will *not* be graded.
- A list of what is implemented in the event that you are submitting a partial solution, or a statement that the problem is solved in full.
- A list of what is not implemented in the event that you are submitting a partial solution.
- A list of known bugs, or a statement that there are no known bugs.
- Any other answers to questions specified in the problem.

To assist the TA in marking, please include your name and the name of the file at the beginning of each file you submit.

Bonus Problem for Everyone

Problem 7 — Mixed Vigenère cipher cryptanalysis, 10 marks

This is a hard problem. Mixed Vigenère cipher cryptanalysis is far more difficult than ordinary Vigenère cipher cryptanalysis. We did not cover the mixed Vigenère cipher in class, so you need to figure out on your own how it works.

Decrypt the following ciphertext that was encrypted using a *mixed* Vigenère cipher. Show all your work; this includes source code if you used programming. Answers without satisfactory explanation and documentation of how they were obtained will receive *no* credit. Neither will answers obtained by simply running mixed Vigenère decryption from an online crypto applet website on the ciphertext.

A text file containing the ciphertext can be downloaded from the "assignments" page.

Hint: The key word has length 6.

```
UNFDN KEPBX PXNMF IOWHM IDNHH ETEJV UNYIV OEXUF OCWVM DZRTB RETEV
XYENE GPFOV QTLFR CVPBV UNQGH YMQFE KUIOV PKYUV FTXOE VXMNA JMTCW
OEZRW BXLQV UNFAT OYNOL VXNQQ DZRXB AQVFG NJIPC ZHFUN FGHQH FXGFA
SUVPH ODZRY BFFMH CEOOK WEFII ZWBLA JTREN BHSRC XDZJF CJBVU DVGOI
UZQBB MURNC GONEU NFXVX WXEFZ IEJYI USVRQ KOWIQ RIBYN GNDZR DGOLV
QPGVX VWEWB MUREQ USVNW BYMVC VLGMO VVCMF ERNDJ VMTCV AEOAV CULNW
ZVFFJ FCJOL QVPHM JQTHX EOVCD KQFOC XGPXW QVWOL VKCKT MOPUG XVZXZ
CFDNY RXAOM TUGKE CFOGQ TNFEQ QKMWE HOKJO RLJVB QVGUD CFMVV HCFYD
AZOPG KYFUC WWASU DVBER ZOUKI IQCGC AVQFZ NFMJZ HFJVM TCHTD PSEHN
IUCNR QQVFZ GYVCH PWBVW RWFII JHPKB OUECX NZMVC BJVXV PHODZ RYBJJ
WOENB HLIUB FHJWE XFDBZ NGOLV QUGEI QVPHQ HJXSK ELCHD HJHVF GQUDV
QNGQV VVPQN XVDRP BXBXT JOESR NRULC FYCAZ MTUGB CVFYK IMBZN GOLVQ
GJFFM TWXEL CHGQU DVQGJ JXZWE GYDZX UXJKZ VWBJF OFCJJ XLXUA IIUVP
```

```
KGMJH UGGLV QNHWI OHXHR QMWZG YVVQC EAOMO EMOEZ RNLBD ZRNGQ FPDUN
FAUHC KQFOC PKBOU VPKQI GVDGY VCFYN FOIWD EBVKH CFAZM YVBNM BVJGA
HQHUN FJCQG FYFKH VLOLV QTCJV XHHXX IJRGV BVMOE MOLJW HMIDZ RSKBQ
VCNGA JBXEJ RONWA KEFPD CDUEW VPKXI VUTXH DNKEI PUCCD GYVCF YHMMX
TUBBM OZCMB VMWRH MSQHU NFAFR NKNFY OEMOL VONPB AOWNF IDZQV RHLQD
WXEEW PNKFV WOCVE XQFZV JDMDC YJSPX YKUFB SCFOL VSPHE XVREX AXCPE
CAJWO YNOMO PXROF PDUNF EMTCB OOQJC VFOBX BGQKM TCBVD MRZBA FUHCK
NIUVV IFKNO EMJVM TCGOL VQZQY IIVTG QFOCU NFRNE CXJVM TCYJS PXYKU
DZRQE BXBRZ MBPVC WFOLV EJQOL FXNDF AVHWX EQVUU FIICQ MNJSU QCXRS
NHCLV OMTCB QEJVP FIIWO CVEXX XIKXF KVVBA SPOEM IMPDG HQHMT TALKJ
WIKUE WWBJG EJRGF OLVQV HEHFO EJMIU VVHOO QNAHQ HEOBV BKVHX KRFTR
BKUXI WDWAV
```