CPSC 418 / MATH 318 — Introduction to Cryptography ASSIGNMENT 1

Due: **Wednesday, Feb. 12, 2020** at **11:55 PM** Total marks: **100** + 5 bonus marks

Prior to submission, be sure to familiarize yourself with the **Policies and Guidelines** as well as the **Specifications and Submission Procedure** as detailed on the assignments course webpage

http://people.ucalgary.ca/~rscheidl/418/assignments.html.

Assignments that don't follow these instructions will incur penalties, possibly even a score of zero.

Written Problems for CPSC 418 and MATH 318

Problem 1 — Linear Feedback Shift Register Key Streams (10 marks)

Stream ciphers such as the one-time pad require a secret key stream of random bits which is bitwise x-or'ed with the plaintext to produce a ciphertext. In this problem, you will cryptanalyze one possible approach for generating such a key stream.

Let m be a positive integer and $c_0, c_1, \ldots, c_{m-1} \in \{0,1\}$ a sequence of m fixed bits. Let $z_0, z_1, \ldots, z_{m-1}$ be any sequence of m bits and define $z_m, z_{m+1}, z_{m+1}, \ldots$ via the linear recurrence

$$z_{n+m} \equiv c_{m-1}z_{n+m-1} + c_{m-2}z_{n+m-2} + \cdots + c_1 z_{n+1} + c_0 z_n \pmod{2} , \qquad (1)$$

with the usual arithmetic modulo 2. The fixed bits $c_0, c_1, \ldots c_{m-1}$ are the *coefficients* of the linear recurrence (1) and the intial values $z_0, z_1, \ldots, z_{m-1}$ are its *seed*. If the seed and the coefficients are appropriately chosen, then (1) generates a sequence of 2^m pseudorandom bits[1] $(z_i)_{i \geq 0}$ from a seed of length m. This type of construction is popular since it can be implemented very efficiently in hardware using a *linear feedback shift register*; see pp. 36-37 of the Stinson-Paterson book.

(a) (2 marks) Let $m = 4$ and consider the recurrence $z_{n+4} \equiv z_{n+3} + z_n \pmod{2}$ with seed $(z_0, z_1, z_2, z_3) = (1, 0, 1, 0)$. Write down the first 19 bits $z_0, z_1, \ldots, z_{18}$ generated by this recurrence and seed.

(b) (4 marks) A user with knowledge of the coefficients and of any m consecutive bits $z_i, z_{i+1}, \ldots, z_{i+m-1}$ (such as the seed, corresponding to the case $i = 0$) can use (1) to generate the entire sequence of bits $(z_n)_{n \geq i}$ starting at z_i. This user is then able to decrypt everything from that point in the plaintext onwards. Explain how an attacker who intercepts a sequence of any $2m$ consecutive bits $z_i, z_{i+1}, \ldots, z_{i+2m-1}$ can potentially obtain the (unknown) coefficients $c_0, c_1, \ldots, c_{m-1}$ and thus completely break the stream cipher. Your description need not be long, but it should be clear and concise.

(c) (4 marks) Suppose the sequence $(1, 1, 1, 1, 0, 0, 1, 1)$ of 8 consecutive bits was generated using an unknown linear recurrence of the form (1) with $m = 4$. Use your attack of part (b) to find the coefficients c_0, c_1, c_2, c_3 of this recurrence.

(*Suggestion:* check your answer, i.e. ensure that the coefficients you obtain define a recurrence that produces the second four bits from the first four.)

[1]Since there are only $2^m - 1$ distinct non-zero bit patterns of length m, there must be repetition after at most $2^m - 1$ bits. So in practice, m must be large.

Problem 2 — Password Counts (20 marks plus 5 bonus marks)

In each question below, provide a brief explanation, a formula and the (exact or approximate) numerical value for your answer.

There are 94 *printable characters* on a standard North American keyboard, comprised of the 26 upper case letters A-Z, the 26 lower case letters a-z, the 10 numerical digits 0-9 and the 32 special characters `‘’".,;:!?~@#$%^&*_-+=(){}[]<>\/|`. Passwords are strings consisting of printable characters.

(a) (2 marks) What is the total number of passwords of length 8?

(b) (4 marks) Suppose a user has a password of length 8 whose first four characters are the first four letters of their child's name (all in either lower or upper case) and the second four characters are the child's birthday in the format DDMM. Intelligence gathering on your part reveals that the child's first name starts with 'L' and that the child was born in 2008. Making no further assumption about a "reasonable" first name (e.g., for all you know, the kid's name could be 'Lxqscdx'), what is the minimal number of candidates that our lazy user could potentially be using as their password?

(c) (5 marks) To guard against *dictionary attacks* (where a password cracker checks if a password is a common word or phrase), passwords are typically required to contain at least one numerical digit and at least one special character. What is the total number of passwords of length 8 that satisfy this requirement?

(*Hint:* It is easier to characterize the number of passwords that violate this rule and subtract that count from the total number of 8-character passwords. But be carefully that you don't subtract some passwords twice.)

(d) (2 marks) What is the percentage of 8-character passwords that satisfy the rule of part (c)?

(e) (**Bonus Question**, 5 marks) A more stringent complexity rule requires passwords to contain at least one upper case letter, at least one numerical digit and at least one special character. What is the total number of passwords of length 8 that satisfy this rule?

(f) (3 marks) Assuming that each permissable character in a password is chosen equally likely[2], what is the entropy of the password space of

- part (a)?
- part (c)?

(g) (4 marks) Suppose we want a password space with entropy 128, assuming that keys are chosen equally likely, i.e. a total of 2^{128} passwords (this number is typical for key space sizes of modern cryptosystems). Assuming no restrictions on the characters appearing in passwords (i.e. the scenario of part (a)), what is the minimum password length that guarantees a password space with entropy 128?

[2]This assumption may be appropriate for passwords chosen by computer systems with good random number generators, but it is utterly false for passwords chosen by humans. So in practice, the minimum password length computed in part (g) is a significant underestimate.

Problem 3 — Probabilities of Non-Collisions (26 marks)

The cryptographic relevance of this question will become clear when we cover hash functions in class.

In each question below, provide a brief explanation and a formula for your answer. In addition, for parts (d) and (f), provide a numerical value that is an integer.

Let k, n be positive integers with $k \leq n$. Consider an experiment involving a group of k participants, where we assign each participant a number that is randomly chosen from the set $\{1, 2, \ldots, n\}$ (so all these assignments are independent events). Now pick your favourite number N between 1 and n.

(a) (2 marks) What is the probability that a given participant is assigned your favourite number N?

(b) (2 marks) What is the probability that a given participant is not assigned the number N?

(c) (3 marks) What is the probability that none of the k participants is assigned the number N?

(d) (4 marks) Intuitively, the more people participate, the likelier the chance that one of them is assigned the number N. In other words, higher values of k decrease the probability that no one gets assigned N.

Suppose $n = 10$. What is the maximal number k of participants in this experiment to ensure at least a 50% chance that none of then is assigned the number N?

(e) (4 marks) Going back to k participants and n numbers, what is the probability that all k participants are assigned different numbers?

(f) (4 marks) Intuitively, the more people participate, the likelier the chance that two of them are assigned the same number (this is called a *collision*; hence the title of this problem). In other words, higher values of k decrease the probability that all participants are assigned different numbers.

Suppose $n = 10$. What is the maximal number k of participants in this experiment to ensure at least a 50% chance that they are all assigned different numbers?

(g) (4 marks) Let P be the quantity of part (e). Prove that $P \approx \exp(-k^2/2n)$ when k is large but is very small compared to n. You may use without proof the following two approximations:

- $\exp(-x) \approx 1 - x$ when $x > 0$ is very small. This comes from the Taylor series

$$\exp(-x) = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \cdots + \cdots ,$$

where the terms from $x^2/2$ onwards are so small that they are negligible compared to $1 - x$.

- $k(k-1) \approx k^2$ (since $k \approx k - 1$ when k is large).

(h) (3 marks) Generalizing part (f) from $n = 10$ to any n, suppose we want to enlist sufficiently many participants to ensure a roughly 50% chance that they are all assigned different numbers. Use the result of part (g) to prove that the number k of required participants is approximately $1.177\sqrt{n}$.

(*Note:* You can solve this question even if you didn't do parts (f) and (g).)

Problem 4 — Equiprobability maximizes entropy for two outcomes, 10 marks

Let X be a random variable consisting of two outcomes X_1 and X_2, both occurring with positive probabilities $p(X_1) = p$ 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 $p(X_1) = 1/8$ and $p(X_2) = 7/8$. Numerically calculate $H(X)$.

(b) (6 marks) Prove that if $H(X)$ is maximal, then both outcomes are equally likely. (You may *not* use Theorems 3.5 and 3.6 of the Stinson-Paterson book.)

(*Hint:* First year calculus: consider $H(X)$ as a function of p and determine for which value of p it takes on its maximum.)

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

Written Problem for MATH 318 only

Problem 5 — Cryptanalysis of a class of linear ciphers, 34 marks)

Let $\mathbb{F}_2 = \{0, 1\}$ with the usual arithmetic modulo 2. The set $\mathbb{F}_2^n = \{0, 1\}^n$ consisting of all n-bit vectors with entries 0 or 1 is an n-dimensional vector space over $\mathbb{F}_2$ (with the usual canonical basis, for example). Linear algebra works exactly like linear algebra over $\mathbb{R}^n$ as a vector space over $\mathbb{R}$, except that linear combinations of vectors in $\mathbb{F}_2^n$ have coefficients 0 and 1.

For any $n \in \mathbb{N}$, let $\mathrm{Gl}_n(\mathbb{F}_2)$ denote the set of invertible $n \times n$ matrices with zeros and ones as entries. Calculations involving such matrices again work exactly the same as the familiar linear algebra over $\mathbb{R}$ you learned in first year, except that arithmetic using real numbers is replaced by arithmetic modulo 2.

Now fix $n \in \mathbb{N}$ and consider the class of linear cryptosystems with $\mathcal{M} = \mathcal{C} = \mathbb{F}_2^n$, $\mathcal{K} = \mathrm{Gl}_n(\mathbb{F}_2)$, and for all plaintexts $\vec{m}$ (interpreted as n-bit column vectors with entries 0 and 1), encryption under a key matrix K is

$$E_K(\vec{m}) = K\vec{m} \; . \tag{2}$$

Then for all ciphertexts $\vec{c}$, decryption under K is obviously

$$D_K(\vec{c}) = K^{-1}\vec{c} \; .$$

(a) (5 marks) Prove, by explicitly describing the key matrix K that encrypts an arbitrary plaintext vector $\vec{m}$ to an arbitrary ciphertext vector $\vec{c}$, that a transposition cipher operating on bit strings of length n is a special case of a linear cipher as given in (2), whose key matrices are *permutation matrices*, i.e. matrices in $\mathrm{Gl}_n(\mathbb{F}_2)$ with exactly one one and $n - 1$ zeros in each row and in each column.

(b) (4 marks) Explain how a cryptanalyst Eve can mount a chosen plaintext attack on a cipher of the form (2). The goal of this attack is to chose one or more plaintexts, obtain their encryptions under some unknown key matrix K, and derive K. How should Eve choose her plaintexts, and how many does she need to choose in order to be successful?

(c) (4 marks) Let $\vec{m}_1, \vec{m}_2, \ldots, \vec{m}_i$ be any collection of i linearly independent vectors in $\mathbb{F}_2^n$ for some i with $1 \le i \le n$. Prove that there are 2^i vectors $\vec{m}_{i+1} \in \mathbb{F}_2^n$ such that the vectors $\vec{m}_1, \vec{m}_2, \ldots, \vec{m}_i, \vec{m}_{i+1}$ are linearly dependent.

(d) (3 marks) With the notation of part (c), how many vectors $\vec{m}_{i+1} \in \mathbb{F}_2^n$ are there so that the vectors $\vec{m}_1, \vec{m}_2, \ldots, \vec{m}_i, \vec{m}_{i+1}$ are linearly independent?

(e) (5 marks) Prove that the number of sets consisting of n linearly independent vectors in $\mathbb{F}_2^n$ is

$$\frac{1}{n!} \prod_{i=0}^{n-1} (2^n - 2^i) \ .$$

(f) (5 marks) Prove that the probability that any set of n vectors in $\mathbb{F}_2^n$ is linearly independent is

$$P_n = \frac{\displaystyle\prod_{i=0}^{n-1} (2^n - 2^i)}{\displaystyle\prod_{i=0}^{n-1} (2^n - i)} \ .$$

(g) (3 marks) Suppose $n = 4$. What is the probability that any set of 4 vectors in $\mathbb{F}_2^4$ is linearly dependent?

(h) (5 marks) Part (b) considered a *chosen* plaintext attack, whereas we now consider the scenario of mounting a *known* plaintext attack on a linear cipher as given in (2). Given a set of known plaintext/ciphertext pairs where all the plaintexts were encrypted to their respective corresponding ciphertexts using the same unknown key matrix K, the goal of this attack is to find K.

Assume that linear dependence of any collection of n plaintexts in $\mathbb{F}_2^n$ represents independent events, i.e. if p is the probability that any n plaintexts are linearly dependent, then p^2 is the probability that any two collections of n plaintexts are linearly dependent. Explain how Eve can use multiple attempts at a known plaintext attack to find a key matrix. What is the minimal number of attempts to guarantee Eve a chance of success of least 99 percent when $n = 4$?

Programming Problem for CPSC 418 only

Problem 6 — Password attack on an authenticated encryption scheme (34 marks)

Overview. Suppose Bob has designed his own authenticated encryption scheme, using the hash-then-encrypt paradigm, with AES-128 in CBC mode for encryption, and SHA1 for the key derivation function and message hash tag. On input the name of a plaintext file and a password p, Bob's program does the following:

(a) Converts the plaintext file to a byte array B.

(b) Computes a hash tag t on the plaintext by applying SHA1 to the byte array B, then appends t to B to obtain an extended byte array $B' = B||t$ (here, as always, "$||$" denotes concatenation).

(c) Derives an encryption key by applying SHA1 to the password p and truncating the result to the appropriate length for use in AES-128.

(d) Generates a random 16-byte initial value IV (for use in CBC mode) and writes it to a file F.

(e) Pads the extended byte array B' using the PKCS7 format if necessary, then encrypts the padded array with AES-128-CBC and appends the resulting ciphertext to the file F.

Out of laziness, Bob is known to use *strings* of the form YYYYMMDD, which mark certain dates from his life, as his passwords. Bob was born in the year 1984, and always includes the string FOXHOUND in his communications.

Problem. Your task is to create a Python 3 program that takes as input a file produced by Bob's hash-then-encrypt routine and performs the following tasks:

(a) Determines the password used to derive the encryption key and prints it out.

(b) Decrypts the ciphertext using this key.

(c) Checks the resulting plaintext for the phrase CODE-RED. If present, the program replaces this phrase with the phrase CODE-BLUE and writes the modified plaintext to a new file. If not present, the plaintext is left unchanged.

(d) In the event that the plaintext was modified as specified in step (c), your program now does the following:

   i. Computes a new hash tag on the modified plaintext.
   ii. Generate a new 16-byte IV.
   iii. Re-encrypt the modified plaintext plus new tag using the same password and exactly the same process as Bob's hash-then-encrypt program.
   iv. Writes the result to a new file. Be aware that modified files may require padding, for which you should use PKCS7 as the padding format. Also recall that AES-128-CBC has a block size of 16 bytes.

Specifications. Design and implement your solution as two `Python 3` programs entitled `modifyFile` and `encryptFile`. The first program should perform steps (a)-(c) above, and be invoked by the command

```
python3 modifyFile [ciphertext-filename]
```

where the input file `ciphertext-filename` is the file produced by Bob's hash-then-encrypt routine. The second program should perform step (d) above, and be invoked by the command

```
python3 encryptFile [plaintext-filename] [tampered-filename] [password]
```

where the input file `plaintext-filename` contains the (potentially modified) plaintext produced in steps (b) and (c) above, the output `tampered-filename` is the file created in step (d) iv. above, and `password` is the password found in step (a).

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

Use the latest version of the `Python` *cryptography* library found at

`https://cryptography.io/en/latest/`

This link can also be found on the "references" page of our course website. Use this library for all the required cryptographic primitives, including the PKCS7 `padding` module.

The `cryptography` library has its own interface to which you are expected to adhere. You **must** make use of the *hazardous materials layer*, **not** the *recipes layer*. Make sure to use good coding practices.

You may use whatever development platform you like. The TAs will test your programs using the latest version of `python3` installed on the CPSC Linux servers. The testing inputs will be Bob's encryptions of at least two different plaintext files which may or may not contain the phrase CODE-RED.

You may assume all plaintext files are text files of at most 1 MB in size. All byte encoding is done using UTF-8.

Submission. Submit a description of your implementation in a separate README file in text format. **Do *not* include the written portion of the programming problem in the PDF file containing your solutions to the written problems.** Your description must include the following:

- A list of the files you have submitted that pertain to the problem, and a short description of each file.
- 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.

Bonus Problem for CPSC 418 and MATH 318

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 as illustrated in some of our handouts.

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
```