ECE424: Network security

Homework 2
Due: 10/09/2024

Collaboration Policy. Each student must hand in their own answers. Use of partial or entire solutions obtained from others or online is strictly prohibited. However, discussing the problems with other fellow students or forming study groups is encouraged.

Submission Format. Please submit your solutions electronically through Canvas as a **single PDF file**. This file should include both your code and clear explanations of each step involved. Ensure to also provide the outputs generated by your codes. Additionally, include answers to any non-code related questions. It is crucial that your solutions are organized in a way that allows for easy review and grading by our team.

Rationale and learning expectations. In this assignment, we delve into the One-Time Pad (OTP) technique and investigate the utilization of several pseudorandom number generators (PRNGs) that we discussed in our lectures to produce the encryption key.

1. *One-time-pad for ASCII.* ASCII (American Standard Code for Information Interchange) is a character encoding standard that assigns unique numerical values to each character, including letters, digits, and special symbols. For example, the ASCII value for 'A' is 65, 'B' is 66, and so on. In the classic form, each ASCII letter is represented by 7 bits, corresponding to 128 unique letters. An extended version of ASCII, referred to as ASCII-8, represents every letter with 8 bits.

   In the context of OTP, we can use ASCII-8 representation to map each plaintext letter and its corresponding key letter to their 8-bit binary representations and then perform XOP operation on the two binary strings as before. Here is an example: assume that the plaintext letter is 'a' and the corresponding key letter is '$'. Then, 'a' corresponds to 01100001 and '$' corresponds to '00100100'. Then,

   $$01100001 \oplus 00100100 = 01000101,$$

   which corresponds to letter 'E'.

   (a) Write an OTP encryption code that receives a key and a plaintext, both in ASCII letters, and returns the ciphertext (i.e., the binary XOR of the plaintext and the key, as we explained earlier).

   (b) Write an OTP decryption algorithm that receives a key and a ciphertext (both in ASCII) and recovers the plaintext.

   (c) Test your code on plaintext 'helloAlice!' and key '$%wB+=?Qz?4'. To check your result, decrypt the resulting ciphertext using the same key.

2. *Blum-Blum-Shub.* Recall the BBS algorithm for generating pseudo random numbers: Let p and q be two large primes such that

$$p \equiv q \equiv 3 \pmod 4),$$

and let

$$n = p \times q.$$

Choose a seed value x_0 that is relatively prime to n. The ith pseudorandom bit is generated as $b_i = x_i \mod 2$, where $x_{i+1} = x_i^2 \pmod n$.

   (a) Choose p, q and x_0 that satisfy the requirements of BBS algorithm.

   (b) Write a code that received p, q, x_0 and integer N as inputs and employs BBS method to generate N bits.

   Hint: If you are using Python, you can use '%' to perform mod *operation.*

   (c) Use the code you have written and test it with the parameters you have selected in part (a) to generate $N = 10^6$ bits. Report the frequency of 0s in your sequence. Also, report the frequency of the pair $(0, 0)$ in your sequence. What is the expected value of this number if your sequence was completely random?

   Hint: To determine the frequency of the pair $(0, 0)$ *in a sequence* $(b_1, \ldots, b_i)$*, count how often* $(0, 0)$ *appears in consecutive pairs. This is mathematically represented as* $\#\{i : (b_i, b_{i+1}) = (0, 0)\}$*. Then, divide this count by the maximum possible number of such pairs in a sequence of length* n*, which is* $n - 1$*. For instance, consider the sequence:*

$$(1, 0, 0, 0, 1, 1, 0, 1, 0, 0).$$

   Here, the frequency of the pair $(0, 0)$ *is* $\frac{3}{9}$*.*

3. *Linear feedback shift register (LFSR).* Consider the following recurrence mod 2:

$$x_i \equiv x_{i-4} + x_{i-10} + x_{i-11} + x_{i-12} \pmod 2)$$

   (a) What is the length of this recurrence (m)? What is the maximal length of the output of this LFSR before it repeats itself?

   (b) How many bits do we need to initialize this LFSR?

   (c) Write a code that receives the initialization bits and the desired length N and implements this LFSR and output bits.

   (d) Report the frequency of 0s and $(0, 0)$ pairs in your sequence when $N = 4095$.

4. *Linear feedback shift register (LFSR).* Consider an LFSR defined by a recurrence mod 2 as

$$x_i \equiv c_1 x_{i-1} + \ldots + c_m x_{i-m} \pmod 2).$$

Show that is the number of non-zero c_is is odd, then this recurrence cannot generate a sequence of maximal length.