SP 24 INFO-231

Homework 4 - Rnadom Number Generators and Hill Cipher

Due Date: 11:59pm, Thursday, 02/15/2024

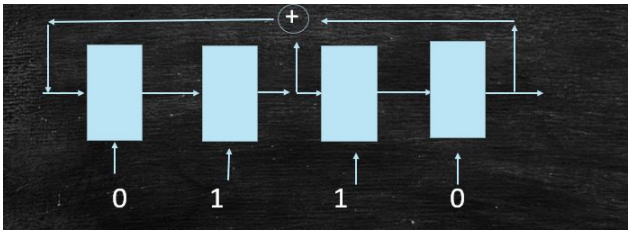
This homework contains 8 questions.

Grade Table (for grading use only)

Question	Points	Score
1	14	
2	10	
3	20	
4	30	
5	10	
6	8	
7	4	
8	4	
Total:	100	

IU user name:_____

1. The linear feedback shift registers are described by a polynomial. These polynomials are called primitive polynomials because they can not be factored into two polynomials. Here is an example of a factored polynomial:



(a) (8 points) What is the polynomial that describes this LFSR?
$x^3 + x + 1$

(b) (6 points) What are the next three outputs?
The polynomial $x^3 + x + 1$ indicates that the feedback taps are at positions 3 and 1 (from the left, starting at position 0). Thus, we need to XOR the bits at positions 3 and 1.
initial state: 0110
1. 0 xor 1 = 1
2. Shift all bits to the right, pushing in the result of the XOR operation at the leftmost bit: 1011
3. Rightmost bit, 1, is the output bit after the first iteration.
4. Repeat steps 1-3 for the next two output bits.
other two bits: 1101, 0110

2. For each polynomial below, what is the degree of the polynomial? (If you ever needed to use a primitive polynomial, look one up! Do not make one up! These are not all primitive polynomials!)

   For each polynomial below, what is the maximum number of unique states of a LSFR represented by each of these polynomials? That is, if these were primitive polynomials how many outputs until the bits begin to repeat?

   Number of unique states of a LSFR $= 2^n - 1$ where n is the degree of the polynomial

   (a) (2 points) $x^3 + x + 1$
   Degree: 3 Number of unique states: 7

   (b) (2 points) $x + x^4 + 1$
   Degree: 4 Number of unique states: 15

   (c) (2 points) $x^3 + x + x^2 + x + 1$
   Degree: 3 Number of unique states: 7

   (d) (2 points) x
   Degree: 1 Number of unique states: 1

   (e) (2 points) $x + 1$
   Degree: 1 Number of unique states: 1

3. Calculate the determinants of the following 2×2 matrices. If the inverse of the matrix exists, calculate the inverse too. Show your work.

Determinant of a 2×2 matrix is $(a \times c) - (b \times d)$

$\begin{pmatrix} a & b \\ d & c \end{pmatrix}$

(a) (4 points) $\begin{pmatrix} 3 & 2 \\ 1 & 5 \end{pmatrix}$

$3 \times 5 - 2 \times 1 = 13$

(b) (4 points) $\begin{pmatrix} 4 & 10 \\ 2 & 5 \end{pmatrix}$

$4 \times 5 - 10 \times 2 = 0$

Inverse does not exist since the determinant is 0

(c) (4 points) $\begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix}$

$1 \times 2 - (-1 \times 0) = 2$

(d) (4 points) $\begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix}$

$4 \times 1 - (3 \times 1) = 1$

(e) (4 points) $\begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix}$

$2 \times 1 - (2 \times 1) = 0$

Inverse does not exist since the determinant is 0

4. (a) (15 points) Encrypt the plaintext "RONALDO" using the Hill Cipher with the key matrix. Pad the plaintext with extra Zs when needed:

$$\begin{pmatrix} 10 & 17 & 5 \\ 21 & 6 & 20 \\ 2 & 2 & 11 \end{pmatrix}$$

$$\begin{bmatrix} 10 & 17 & 5 \\ 21 & 6 & 20 \\ 2 & 2 & 11 \end{bmatrix} \cdot \begin{bmatrix} 17 \\ 14 \\ 13 \end{bmatrix} = \begin{bmatrix} (10)\cdot(17) + (17)\cdot(14) + (5)\cdot(13) \\ (21)\cdot(17) + (6)\cdot(14) + (20)\cdot(13) \\ (2)\cdot(17) + (2)\cdot(14) + (11)\cdot(13) \end{bmatrix} = \begin{bmatrix} 473 \\ 701 \\ 205 \end{bmatrix}$$

$$\begin{pmatrix} 473 \\ 701 \\ 205 \end{pmatrix} \mod 26 = \begin{pmatrix} 5 \\ 25 \\ 23 \end{pmatrix}$$

$$\begin{bmatrix} 10 & 17 & 5 \\ 21 & 6 & 20 \\ 2 & 2 & 11 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 11 \\ 3 \end{bmatrix} = \begin{bmatrix} (10)\cdot(0) + (17)\cdot(11) + (5)\cdot(3) \\ (21)\cdot(0) + (6)\cdot(11) + (20)\cdot(3) \\ (2)\cdot(0) + (2)\cdot(11) + (11)\cdot(3) \end{bmatrix} = \begin{bmatrix} 202 \\ 126 \\ 55 \end{bmatrix} \mod 26$$

$$\begin{pmatrix} 202 \\ 126 \\ 55 \end{pmatrix} \mod 26 = \begin{pmatrix} 20 \\ 22 \\ 3 \end{pmatrix}$$

$$\begin{bmatrix} 10 & 17 & 5 \\ 21 & 6 & 20 \\ 2 & 2 & 11 \end{bmatrix} \cdot \begin{bmatrix} 14 \\ 25 \\ 25 \end{bmatrix} = \begin{bmatrix} (10)\cdot(14) + (17)\cdot(25) + (5)\cdot(25) \\ (21)\cdot(14) + (6)\cdot(25) + (20)\cdot(25) \\ (2)\cdot(14) + (2)\cdot(25) + (11)\cdot(25) \end{bmatrix} = \begin{bmatrix} 690 \\ 944 \\ 353 \end{bmatrix}$$

$$\begin{pmatrix} 690 \\ 944 \\ 353 \end{pmatrix} \mod 26 = \begin{pmatrix} 14 \\ 8 \\ 15 \end{pmatrix}$$

So when you map the matrices to the corresponding keys, you get **FZXUWDOIP**

(b) (15 points) Encrypt the plaintext "PASSWORD" using the Hill Cipher with the key matrix. Pad the plaintext with extra Zs when needed:

$$\begin{pmatrix} 6 & 20 & 1 \\ 17 & 16 & 19 \\ 21 & 14 & 15 \end{pmatrix}$$

$$\begin{bmatrix} 6 & 20 & 1 \\ 17 & 16 & 19 \\ 21 & 14 & 15 \end{bmatrix} \cdot \begin{bmatrix} 15 \\ 0 \\ 18 \end{bmatrix} = \begin{bmatrix} (6)\cdot(15)+(20)\cdot(0)+(1)\cdot(18) \\ (17)\cdot(15)+(16)\cdot(0)+(19)\cdot(18) \\ (21)\cdot(15)+(14)\cdot(0)+(15)\cdot(18) \end{bmatrix} = \begin{bmatrix} 108 \\ 597 \\ 585 \end{bmatrix}$$

$$\begin{pmatrix} 108 \\ 597 \\ 585 \end{pmatrix} \mod 26 = \begin{pmatrix} 4 \\ 25 \\ 13 \end{pmatrix}$$

$$\begin{bmatrix} 6 & 20 & 1 \\ 17 & 16 & 19 \\ 21 & 14 & 15 \end{bmatrix} \cdot \begin{bmatrix} 18 \\ 22 \\ 14 \end{bmatrix} = \begin{bmatrix} (6)\cdot(18)+(20)\cdot(22)+(1)\cdot(14) \\ (17)\cdot(18)+(16)\cdot(22)+(19)\cdot(14) \\ (21)\cdot(18)+(14)\cdot(22)+(15)\cdot(14) \end{bmatrix} = \begin{bmatrix} 562 \\ 924 \\ 896 \end{bmatrix}$$

$$\begin{pmatrix} 562 \\ 924 \\ 896 \end{pmatrix} \mod 26 = \begin{pmatrix} 16 \\ 14 \\ 12 \end{pmatrix}$$

$$\begin{bmatrix} 6 & 20 & 1 \\ 17 & 16 & 19 \\ 21 & 14 & 15 \end{bmatrix} \cdot \begin{bmatrix} 17 \\ 3 \\ 25 \end{bmatrix} = \begin{bmatrix} (6)\cdot(17)+(20)\cdot(3)+(1)\cdot(25) \\ (17)\cdot(17)+(16)\cdot(3)+(19)\cdot(25) \\ (21)\cdot(17)+(14)\cdot(3)+(15)\cdot(25) \end{bmatrix} = \begin{bmatrix} 187 \\ 812 \\ 774 \end{bmatrix}$$

$$\begin{pmatrix} 187 \\ 812 \\ 774 \end{pmatrix} \mod 26 = \begin{pmatrix} 5 \\ 6 \\ 20 \end{pmatrix}$$

So when you map the matrices to the corresponding keys, you get:
EZNQOMFGU

5. (10 points) From the readings explain the different types of pseudorandom number generators and their properties.

   Pseudorandom number generators (PRNGs) are algorithms that produce sequences of numbers that mimic the statistical properties of random numbers. PRNGs are used in a wide range of applications, including cryptography, simulation, and gaming. There are several types of PRNGs, including:

   1. Linear Congruential Generators (LCGs): These are the simplest type of PRNGs, and they work by using a simple linear equation to generate the next number in the sequence. LCGs are efficient and easy to implement, but their output can be predictable if the seed value is known.

   2. Shift-Register Generators: These PRNGs use a linear feedback shift register (LFSR) to generate the next number in the sequence. They are more complex than LCGs, but they produce longer sequences of random numbers before repeating. Each type of PRNG has its own strengths and weaknesses, and the choice of PRNG depends on the specific requirements of the application. For applications that require high levels of security, such as cryptography, CSPRNGs are typically preferred. For other applications, such as simulation or gaming, other types of PRNGs may be more suitable.

6. (8 points) In the context of True Random Number Generators (TRNG), discuss the different possible sources of entropy and how TRNGs are different from Psuedo Random Number Generators (PRNG).

Sources of Entropy for TRNGs:

Mouse Movement: When you move your mouse, the exact path and speed can be somewhat unpredictable. TRNGs can measure these movements and convert them into random numbers.

Keyboard Typing Patterns: Everyone has a unique typing style, including how fast they type and the rhythm of their keystrokes. This variation can be used as a source of randomness.

System Clock Jitter: The clock in your computer isn't perfect and can sometimes have tiny variations in timing. TRNGs can capture these fluctuations to generate random numbers.

Environmental Noise: The sounds around you, like the hum of a fan or the ticking of a clock, are never exactly the same. TRNGs can use microphones to pick up these noises and turn them into randomness.

Network Packet Timing: When data is sent over a network, the timing of each packet's arrival can vary slightly. TRNGs can measure these timing differences to generate random numbers.

Difference between TRNGs and PRNGs:

True Randomness vs. Pseudorandomness: TRNGs generate truly random numbers based on unpredictable physical processes, like the ones mentioned above. PRNGs, on the other hand, use mathematical algorithms to generate sequences of numbers that only appear random. However, PRNGs are deterministic, meaning if you know the algorithm and the starting point (called the seed), you can predict all the numbers it will generate.

Unpredictability: TRNGs provide higher unpredictability since their randomness is derived from physical phenomena. In contrast, PRNGs, while they might seem random, are entirely predictable given the seed and algorithm used.

Applications: TRNGs are typically used in applications where true randomness is critical, such as cryptography, secure communication, and gambling. PRNGs are more commonly used in simulations, numerical analysis, and other applications where statistical randomness is sufficient.

Efficiency: PRNGs are usually more efficient and faster than TRNGs because they rely on mathematical formulas rather than physical measurements. TRNGs might require additional hardware and processing, making them slower and potentially more expensive to implement.

In summary, while both TRNGs and PRNGs are used for generating random numbers, TRNGs rely on physical processes for true randomness, while PRNGs use algorithms for

pseudorandomness. TRNGs provide higher unpredictability but can be slower and more resource-intensive compared to PRNGs.

7. (4 points) Show that each digit is its own additive inverse modulo 2. To show that each digit is its own additive inverse modulo 2, we need to prove that for any digit d, adding d to itself modulo 2 gives 0.

   In modular arithmetic, we define the additive inverse of an integer a as the integer that, when added to a, results in the modular additive identity 0. That is, the additive inverse of a is the integer b such that $a + b \equiv 0 \mod n$, where n is the modulus.

   In the case of modulo 2 arithmetic, the modulus is 2, and the only two possible remainders are 0 and 1. Therefore, each digit is either equivalent to 0 or 1 modulo 2.

   We can then simply show this for all the digits 0 to 9.

8. (4 points) Define confusion and diffusion.

Confusion

—Relation between the key and the ciphertext is obfuscated

—This is implemented by substitution

Diffusion

—Relation between any bit of the plaintext is spread over the ciphertext

—This is implemented with permutation

The Diffusion and Confusion are used to protect the encryption key or protect the plaintext. These two operations increase challenges for attackers so that they cannot easily use the statistical properties of the plaintext to deduce the key.