SaraBaradaran

9624193 homework 2

Question 1)

1.1

primitive polynomials : LFSRs which generate a maximum-length sequence is called primitive polynomials.

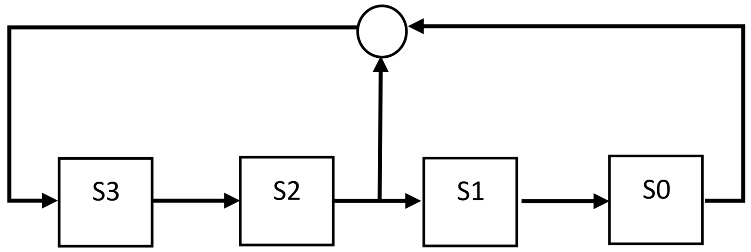
irreducible polynomials : LFSRs which their sequence length is independent of the initial value of the register is called irreducible polynomials.

(also all primitive polynomials are irreducible polynomials)

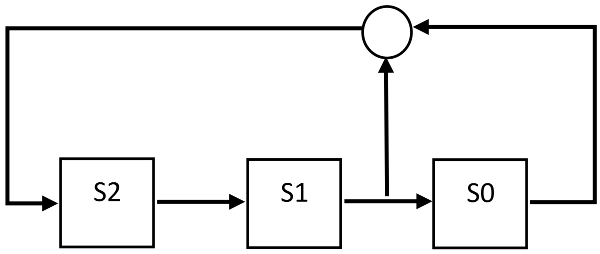
reducible polynomials : LFSRs which do not generate a maximum-length sequence and their sequence length is dependent of the initial value of the register is called reducible polynomials.

1.2

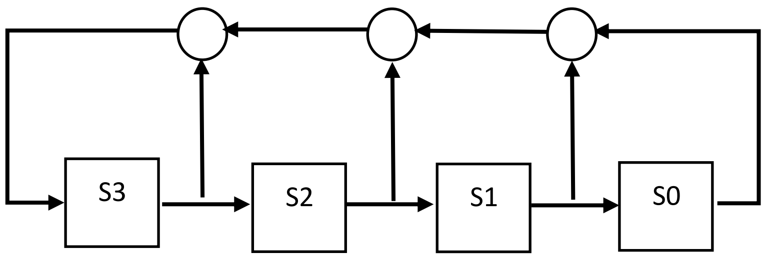
$x^4 + x^2 + 1$



$x^3 + x + 1$



$x^4 + x^3 + x^2 + 1$



1.3 , 1.4

$$1 + x^2 + x^4 = (1 + x^2 - x)\,(1 + x^2 + x)$$

initialization vector = 0011 → 1001 → 1100 → 1110 → 1111 → 0111 → 0011

initialization vector = 0110 → 1011 → 1101 → 0110

period depends on

initialization vector

reducible polynomial

$$\begin{array}{l|l}
1 & 1 + x + x^3 \\ \cline{2-2}
\underline{1 + x + x^3} & 1 + x + x^2 + x^4 \\
\quad x + x^3 & \\
\quad \underline{x + x^2 + x^4} & \\
\quad x^2 + x^3 + x^4 & \\
\quad \underline{x^2 + x^3 + x^5} & \\
\qquad x^4 + x^5 & \\
\qquad \underline{x^4 + x^5 + x^7} & \\
\qquad\qquad x^7 &
\end{array}$$

period = 7 = 2^3 - 1

primitive polynomial

irreducible polynomial

$$\begin{array}{l|l}
1 & 1 + x + x^2 + x^3 + x^4 \\ \cline{2-2}
\underline{1 + x + x^2 + x^3 + x^4} & 1 + x \\
\quad x + x^2 + x^3 + x^4 & \\
\quad \underline{x + x^2 + x^3 + x^4 + x^5} & \\
\qquad\qquad x^5 &
\end{array}$$

period = 5

irreducible polynomial

Question 2)

2.1

Period or sequence-length = 31

2.2

11011 → 01101 → 00110 → 00011 → 10001 → 11000 → 11100 → 11110 → 11111 → 01111 → 00111 → 10011 → 11001 → 01100 → 10110 → 01011 → 00101 → 10010 → 01001 → 00100 → 00010 → 00001 → 10000 → 01000 → 10100 → 01010 → 10101 → 11010 → 11101 → 01110 → 10111 → 11011

Question 3)

Advantages:

The OTP is unconditionally secure in other words it can't be broken even with infinite computational resources. In this method there is 50% chance that each bit has the value 0 or 1.

Disadvantages:

It is highly impractical for most applications because the key length has to be equal the message length , it is difficult to transfer And This key can't be reproduced.

Question 4)

4. 1

The program is attached to this file → LFSR.py

4.2

Plaintext → wpi?????? → w = 10110, p = 01111, i = 01000

Ciphertext → j5a0edj2b → j = 01001, 5 = 11111, a = 00000

w (xor) j = 11111, p (xor) 5 = 10000, i (xor) a = 01000

$s_0\, s_1\, s_2\, s_3\, s_4 \quad s_5\, s_6\, s_7\, s_8\, s_9 \quad s_{10} s_{11} s_{12} s_{13} s_{14}$ …

keystream = 1 1 1 1 1 _ 1 0 0 0 0 _ 0 1 0 0 0 …

sequence-length of keystream = 6 → initialization vector = 111111 = $s_0 s_1 s_2 s_3 s_4 s_5$

4.3

$s_6 = p_0 s_0 + p_1 s_1 + p_2 s_2 + p_3 s_3 + p_4 s_4 + p_5 s_5 \pmod 2$

$s_7 = p_0 s_1 + p_1 s_2 + p_2 s_3 + p_3 s_4 + p_4 s_5 + p_5 s_6 \pmod 2$

$s_8 = p_0 s_2 + p_1 s_3 + p_2 s_4 + p_3 s_5 + p_4 s_6 + p_5 s_7 \pmod 2$

$s_9 = p_0 s_3 + p_1 s_4 + p_2 s_5 + p_3 s_6 + p_4 s_7 + p_5 s_8 \pmod 2$

$s_{10} = p_0 s_4 + p_1 s_5 + p_2 s_6 + p_3 s_7 + p_4 s_8 + p_5 s_9 \pmod 2$

$s_{11} = p_0 s_5 + p_1 s_6 + p_2 s_7 + p_3 s_8 + p_4 s_9 + p_5 s_{10} \pmod 2$

$0 = p_0 + p_1 + p_2 + p_3 + p_4 + p_5 \pmod 2 \quad \rightarrow p_5 = 0$
$0 = p_0 + p_1 + p_2 + p_3 + p_4 \pmod 2 \quad \rightarrow p_4 = 0$
$0 = p_0 + p_1 + p_2 + p_3 \pmod 2 \quad \rightarrow p_3 = 0$
$0 = p_0 + p_1 + p_2 \pmod 2 \quad \rightarrow p_2 = 0$
$0 = p_0 + p_1 \pmod 2 \quad \rightarrow p_1 = 1$
$1 = p_0 \pmod 2 \quad \rightarrow p_0 = 1$

➔ Feedback = $x^6 + x + 1$

4.4

➔ Plaintext = Keystream (xor) ciphertext

Now we have initialization vector and feedback so we can generate keystream.

keystream: 1 1 1 1 1_1 0 0 0 0_0 1 0 0 0_0 1 1 0 0_0 1 0 1 0_0 1 1 1 1_0 1 0 0 0_1 1 1 0 0_1 0 0 1 0
cipher text: 0 1 0 0 1_1 1 1 1 1_0 0 0 0 0_1 1 0 1 0_0 0 1 0 0_0 0 0 1 1_0 1 0 0 1_1 1 1 0 0_0 0 0 0 1

---

plain text: 1 0 1 1 0_0 1 1 1 1_0 1 0 0 0_1 0 1 1 0_0 1 1 1 0_0 1 1 0 0_0 0 0 0 1_0 0 0 0 0_1 0 0 1 1

W P I W O M B A T

4.5

Type of attack → known plaintext attack

<u>Question 5)</u>

<u>The program is attached to this file → Trivium.py</u>

<u>Optional Question)</u>

BARACKOBAMA

B → 66 = 1000010
A → 65 = 1000001
R → 82 = 1010010
C → 67 = 1000011
K → 75 = 1001011
O → 79 = 1001111
M → 77 = 1001101

1.

Plaintext :

01000010_01000001_01010010_01000001_01000011_01001011_01001111_01000010_01000001_01001101_01000001
66 65 82 65 67 75 79 66 65 77 65

Ciphertext :

01000011_00011011_00010010_00110000_11111000_10100111_10001110_11101001_00010100_00011101_01100100

Plaintext (xor) ciphertext = keystream

01000010_01000001_01010010_01000001_01000011_01001011_01001111_01000010_01000001_01001101_01000001
01000011_00011011_00010010_00110000_11111000_10100111_10001110_11101001_00010100_00011101_01100100
Key stream:

00000001_01011010_01000000_01110001_10111011_11101100_11000001_10101011_01010101_01010000_00100101

Key stream2:

00000010_01011011_01000001_01110010_10111100_11101101_11000010_10101100_01010110_01010001_00100110

Ciphertext2:

01000110_00010100_00001111_00110011_11110000_10101001_10010110_11111110_00000011_00011100_01110110

Keystream2 (xor) ciphertext2 = plaintext2

00000010_01011011_01000001_01110010_10111100_11101101_11000010_10101100_01010110_01010001_00100110
01000110_00010100_00001111_00110011_11110000_10101001_10010110_11111110_00000011_00011100_01110110
Plaintext2:

01000100_01001111_01001110_01000001_01001100_01000100_01010100_01010010_01010101_01001101_01010000

68	79	78	65	76	68	84	82	86	77	80
D	O	N	A	L	D	T	R	U	M	P

2.

Since we know that nonce of 1 was added to each byte for this specific message we cab subtract that from each byte of keystream to get the fixed key. But we don't need to do this because we know the second message was encrypted using a nonce of 2. So the keystream which is used to encrypt the second message can be obtained by adding 1 mod 256 to the first key. Then we can obtain second plaintext using second keystream XOR second ciphertext.