ECE 404 - HW 4

Parth R. Doshi

Theory Problems

(1)

(a) $9x^5 + 4x^4 + 8x^3 + 2x^2 + 3x + 4 + 6x^5 + 2x^4 + 9x^3 + 7x^2 + 5x + 7$

$= (15x^5 + 6x^4 + 17x^3 + 9x^2 + 8x + 11) \bmod 11$

Ans $= 4x^5 + 6x^4 + 6x^3 + 9x^2 + 8x$

(b) $(8x^3 + 6x^2 + 8x + 1) \times (3x^3 + 9x^2 + 7x + 5)$

$= (24x^6 + 72x^5 + 56x^4 + 40x^3) + (18x^5 + 54x^4 + 42x^3 + 30x^2) + (24x^4 + 72x^3 + 56x^2 + 40x) + (3x^3 + 9x^2 + 7x + 5)$

$= 24x^6 + x^5(72 + 18) + x^4(56 + 54 + 24) + x^3(40 + 42 + 72 + 3) + x^2(30 + 56 + 9) + x(40 + 7) + 5$

Ans $= 2x^6 + 2x^5 + 2x^4 + 3x^3 + 7x^2 + 3x + 5$

(c) $\dfrac{(3x^3 - 5x^2 + 10x - 3)}{(3x+1)}$

$3x^3 - 5x^2 + 10x - 3 = 3x^3 + 6x^2 + 10x + 8$

$$
\begin{array}{r}
x^2 + 9x + 4 \\
(3x+1) \,\big)\, 3x^3 + 6x^2 + 10x + 8 \\
-\ (3x^3 + x^2) \\
\hline
5x^2 + 10x + 8 \\
-\ (5x^2 + 9x) \\
\hline
x + 8 \\
-\ (x + 4) \\
\hline
4
\end{array}
$$

Ans: $x^2 + 9x + 4 + \dfrac{4}{3x+1}$

$\dfrac{5}{3} = 5 \times \dfrac{1}{3}$

$= 5 \times 4 = 20$

20 mod 11 = 9

27 mod 11 = 5

(2)

(a) $x^3 + x + 1$ = Irreducible Polynomial

$(x^2 + x + 1) \times (x^2 + x)$

$= x^4 + x^3 + x^2 + x^3 + x^2 + x$

$= (x^4 + x) \bmod (x^3 + x + 1)$

Ans $= x^2$

$$
\begin{array}{r}
x \\
x^3 + x + 1 \,\big)\, x^4 + x \\
-\ (x^4 + x^2 + x) \\
\hline
-x^2
\end{array}
$$

$= +x^2$

(2)

(b) $x^2 - (x^2 + x + 1)$

$= x^2 - x^2 - x - 1$

$= x^2 + x^2 + x + 1$

Ans $= x + 1$

(c) $\frac{x^2 + x + 1}{(x^2 + 1)}$ irreducible polynomial $= x^3 + x + 1$

$\frac{111}{101}$ $= 111 \times MI(101)$

$= 111 \times (010)$

$= (x^2 + x + 1) \times (x)$

$= (x^3 + x^2 + x) \bmod (x^3 + x + 1)$

$$x^3 + x + 1 \;\overline{)\; x^3 + x^2 + x \;(\; 1}$$

$$- \cancel{x^3} + x + 1$$

$$x^2 - 1$$

Ans $= x^2 + 1$

Ans $= 101$

HW04 – ECE 404

1. Encrypt()

a. Encrypted Text :

2bd280a572d58f866b407a63e2ac60a4a58e4f16d71808c75b85a3188aa78de70453883720af22591
5d84feff6fc415edfd642d338f4d61f1d8b696e47a0e2f3769c340a5d249ebaae0fd1817f6db4166b2b
9e32c7a9c93dcf801f52946997ba0f0584ee0b118e3335a5efabf959e799736ec47b6df311c0f05ede
6c2ae6a130d33722616b931f1982d9039f7609f77d734d54b495016d43c5e22e7f9d4b7f9d3fbf031f
af35f93de2178d6b7b1281db88be2c3708441843af5ab489dabde7ddefd3407c4b895fa18bb803259
e4c292536017682376f140070dec722414b5c971b144be144ccbd55169ca58c8785393ab6023ca02
c62e3184dacc3598ed9027a9ef4debd3dbf04b953eabee5ee753046c695ff58206fabcc29e59d4917c
eddc0f791dd3790be6a55dad78c25fb35924c9e3ab50e50fd268ab9c20338a4098aacfb3053534ac9
737828be7a615b609196ec23cf880fa1ae2407ba15a4c4c305f612181320100e5b87649e4eb9565c8
3e1d0898312461e38d63c8452e38abe8099c4cb17964a0d4dd3bbde0ec018d37c2aaa9fe33e1f69a9
d886a7c3fa0f03554965f572d90506bb3c07fc8d8af0d0f10ce1b6eef25f64e4c0a0d8ece2958b860a3
c14e84993511caad9e5f5611f7516d82d89e5680cb8a248b5c3a686d26164c98dc9dd4f8336390afd
a6503b79dce3e9e561b0f006bf32a7071e16fd7e7da6a72a884afce43f42a61c85926a17056f54084f
6355fbe34d6d05eb6cedef0864b8

b. Brief Explanation of Code : AES is short for Advanced Encryption Standard which uses either a 128, 192 or 256 bit key encryption algorithm. In this assignment we use a 256 bit key. Primarily we need to convert the plaintext and the key text into bit vectors. Before any input processing of the plaintext, we performing a XOR operation on the plaintext bit vector with the first round key. To obtain a list of the round keys we use the gen_key_schedule_256() function. This returns the key words. Using these key words we generate a list of 15 round keys for each round. Before any input processing we ensure that the plaintext bit vector is padded correctly (takes 128 bits at a time for encryption). Then we run a for loop for 13 rounds to perform the encryption. The first step is to generate a 4 by 4 byte state array where each byte is ordered column wise. We then perform the substitute byte step which entails using a substitution table given to manipulate the state array. After that we perform the shift rows function where we shift the second, third and fourth row by 1,2 and 3 bytes to the left, respectively. Then we convert the state array which is a list of list of integers to a list of bit vectors. This is imperative since we need to perform the XOR function which the round keys at the end. Once we have converted it to a list of bit vectors, we then perform the mix columns operation. The mix column operation uses the formulae provided in the lecture to mix the columns. Finally we XOR the output bit vector from the mix column step with the first round key (assuming we are on the first round of encryption). This happens for 13 rounds. For the last round we only perform the same functions except mixing columns. Lastly, we convert it into a bit vector and print it to the output file.

2. Decrypt()

a. Decrypted Text : As a constructor in Formula One, Ferari has a record 16 Constructors' Championships. Their most recent Constructors' Championships was won in 2008. The Team also holds the record for the most Drivers' Championships with 15, won by nine different drivers: Alberto Ascari, Juan Manuel Fangio, Mike Hawthorn, Phil Hill, John Surtees, Niki Lauda, Jody Scheckter, Michael Sschumacher and Kimi Raikkonen. Raikkonen's title in 2007 is the most

recent for the team. The 2020 Tuscan Grand Prix marked Ferrari's 1000th Grand Prix in Formula One.

c. Brief Explanation of Code : The logic for the decrypt function is almost identical to the encrypt function with a few caveats. Primarily we need to convert the ciphertext and the key text into bit vectors. Before any input processing of the ciphertext, we performing a XOR operation on the ciphertext bit vector with the first round key. To obtain a list of the round keys we use the gen_key_schedule_256() function. This returns the key words. Using these key words we generate a list of 15 round keys for each round. However we need to reverse this list of round keys. Before any input processing we ensure that the ciphertext bit vector is padded correctly (takes 128 bits at a time for decryption). Then we run a for loop for 13 rounds to perform the decryption. Then we generate a 4 by 4 byte state array where each byte is ordered column wise. The first step is to perform the shift rows function where we shift the second, third and fourth row by 1,2 and 3 bytes to the right, respectively. We then perform the inverse substitute byte step which entails using an inverse substitution table given to manipulate the state array. Then we convert the state array which is a list of list of integers to a list of bit vectors. This is imperative since we need to perform the XOR function which the round keys at the end. Once we have converted it to a list of bit vectors, we then add the round key corresponding to that round. Next, we perform the inverse mix column operation uses the formulae provided in the lecture to inverse mix the columns. This happens for 13 rounds. For the last round we only perform the same functions except inverse mixing columns. Lastly, we convert it into a bit vector and print it to the output file.