Advanced topics in Cybersecurity 1

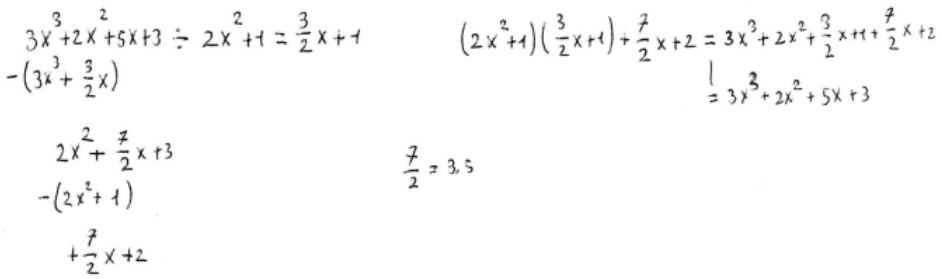
Assignment 1

Kristyan Falcon

17 March 2022

Exercise 1

Being $g(x) = 3x^3 + 2x^2 + 5x + 3$ and $f(x) = 2x^2 + 1$ the result of $g(x) \div f(x)$ is $\frac{3}{2}x + 1$ with a remainder of $\frac{7}{2}x + 2$



Exercise 2

The irreducible polynomial in $GF(2^2)$ is $x^2 + x + 1$. Because if we try to factorize it there won't be a match of polynomials in the field that will factorize it. To extend the polynomial we can try extending it to the GF(2^3) considering the new irreducible polynomial $x^3 + x + 1$. Being a 2^3 the size of the new finite field will be 8 and the possible values will be $0, 1, x, x^2, x + 1, x^2 + 1, x^2 + x, x^2 + x + 1$ corresponding to 000, 001, 010, 100, 011, 101, 110, 111

Exercise 3

Main ideas

Addition

Concerning the addition the idea is simply the XOR of the two binary vaules. Thats because in the case of 2 x added together the modulo 2 will remove that value so if we have $x^2 + x + 1$ and $x^2 + 1$ the sum of those 2 will be $2x^2 + x + 2 \equiv x \mod 2$.

Multiplication

Concerning the multiplication the operation will be a little bit more complex. The main idea is by keeping track of each 1 in the left operator and using those ones to shift the first operator equal times as the position of each 1. The different values obtained by the different shift will then be added together to represent the result (in the code i shift and add until the last bit cause was more compact instead of keeping in the memory each value of the shifts). An example is with the $(x^3 + 1) * (x^2 + 1)$ in binary 1001 0101. i will shift 1001 2 times because there are 2 ones in the second operator. so i will shift it by 2 (0100) and 0 cause (0001). So 100100 + 1001 = 1101. Considering $(x^3 + 1) * (x^2 + 1) = x^5 + x^3 + x^2 + 1$ and considering the $GF(2^4)$ the x^5 will be removed keeping $x^3 + x^2 + 1$ or 1101.

Reduction

After the multiplication there always be a reduction that checks if the value has surpassed the irreducible polynomial and if so will apply the shift and xor operations to get his equivalent by modulo. As an example $(x^3 + x^2 + x)(x^2 + x + 1) = x^5 + x^3 + x$ and $x^5 + x^3 + x \div x^4 + x + 1 = x$ with a remainder of $x^3 + x^2$. That remainder is the result of the operation and can be coded as 1100.

How does the program work

To make the program work firstly has to be compiled using gcc. Then it will return an a.out result and this one will receive 3 arguments. The first one is the operation (+ or x). The second and third one are the polynomials in binary.

Examples of commands

- ./a.out + 1000 1100
- ./a.out + 0010 1011
- ./a.out + 1110 1000
- ./a.out x 1010 0010
- ./a.out x 0110 1011
- ./a.out x 1000 1001

Exercise 4

Main ideas

The program instantiates a matrix that will keep track of each pair (alpha, beta) that will occur keeping the number of them. Than given the permutation operation the program will calculate every beta by a given alpha and x as so.

$$S(x \oplus alpha) + S(x) = beta$$

Counted every couple the program returns a matrix that will represent how many couples for each alpha (as row) and beta (as column). After that the program will ask if the user is interested in a specific case of alpha and beta. Given the 2 cases it will take the number in that position anche divide it by 8 ($2^{(n-1)}$) and return the differential probability.

I didn't reported all the probabilities directly in the matrix because that will screw the formatting during the printing. So i made a function that asks what probability the user wants.

How does the program work

To make the program work firstly has to be compiled using gcc. Then it will return an a.out result and this one will receive no arguments. But at the end of the computation it will ask the user for 2 values (alpha and beta) and calculate the probability.

The result

.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	0	2	0	0	0	0	4	2	4	2	0	0	0	0
2	0	2	0	2	0	4	4	0	0	2	0	2	0	0	0	0
3	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2
4	0	0	4	2	2	0	2	2	0	0	0	2	0	2	0	0
5	0	0	2	0	0	0	0	2	0	0	6	0	2	2	2	0
6	0	2	2	2	2	0	2	2	2	0	0	0	0	2	0	0
7	0	2	0	0	0	0	0	2	0	2	0	4	4	0	0	2
8	0	0	0	0	2	2	0	0	0	0	0	0	4	4	2	2
9	0	0	0	0	4	6	2	0	0	0	0	0	2	0	0	2
A	0	0	0	0	0	2	2	0	2	2	2	2	0	2	2	0
B	0	4	0	4	2	2	0	0	0	0	0	0	0	0	2	2
C	0	0	2	0	0	0	2	0	4	0	2	0	0	0	2	4
D	0	2	4	0	0	0	2	4	0	2	0	0	0	0	2	0
E	0	0	0	2	2	0	0	0	0	4	0	2	0	2	2	2
F	0	2	2	2	2	0	0	4	2	0	0	0	2	0	0	0

The probability for alpha = 15 and beta = 1 is 0.25 or 2/8.

Exercise 5

Main ideas

MixColumn consists in pre-multiply a pre-defined and invertible matrix to a matrix of GF(2^4) elements. The program asks the user for a matrix (row by row) and multiplies that with the static one. The multiplication operation is the same as the exercise 3 (with also the same irreducible polynomial). So after each multiplication and sum there will be a reduciton to the polynomial (with the irreducible one) until it is of grade 3.

How does the program work

To make the program work firstly has to be compiled using gcc. Then it will ask for the matrix. The matrix has to be given row by row. So has to be inserted 4 values (in the hex form) divided by a space, and after that pressing enter to insert the next row.

The result

$$\begin{matrix} 1 & 0 & 2 & 3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 3 & 0 & 1 & 0 \end{matrix} * \begin{matrix} 1 & f & c & 2 \\ 0 & 0 & 1 & 0 \\ f & 0 & 0 & 2 \\ 3 & c & 0 & 0 \end{matrix} = \begin{matrix} 9 & 8 & C & 6 \\ 1 & F & C & 2 \\ D & 0 & 1 & 4 \\ C & 2 & 7 & 4 \end{matrix}$$

Exercise 6

A fix to resolve the problem of not enough memory is to compute the table considering only the first 2^{32} elements. In this case the memory will be able to handle it because the table will have 2^{32} as dimension. This table will have all the $\alpha : \alpha = E_{k1}(x)$. So if we decrypt y using all possible k2 we will try to find a match in the table. If there is a match we can stop the process and we found k1 and k2. If there is no match is because in this particular case we did not stored all the possible k1. So after the negative result of a match in the first table we will compute the second table that starts from $2^{32} + 1$ and covers other 2^{32} possible k1. Computed the new table we will return to decrypting y with all the possible k2 until we find a match. If there is no match again we will continue until we have computed all the possible α values of k1.

Using this method we will still cover all the possible values in the table just only splitting the computation and storage in memory because of lack of space. So the value will be found as in the normal attack but with a higher cost in the computation. The new values of time memory and data complexities are:

- Time: $\mathcal{O}(2^{96})$
- Memory: $\mathcal{O}(2^{32})$
- Data: $\mathcal{O}(1)$

Considering that we will compute parts of the table and try to find a match with all the decrypted y we will consider all the decrypted y as 2^{64} and the tables as 2^{32}. There will be $2^{64}/2^{32}$ possible tables so the decryption and encryption of the y by a table will occur at most 2^{32} times so the cost will be $(2^{64} + 2^{32}) * 2^{32} = 2^{64+32} + 2^{32+32} = 2^{96} + 2^{64}$