CIS 3362 Final Exam - Part B (Modern Private Key Cryptography) - 25 pts Solutions

1) (8 pts) Let the input to the S-boxes in DES be 8EF943C62DA1, represented in hexadecimal. What is the output from the S-boxes, represented in hex? (Your grade will be 1 point per hex character.)

Solution

Converting the input into 8 six bit blocks we get the following and then put them into the appropriate S boxes:

S_1(100011) = row 3, col 1 entry of S_1 = 12 (C)
S_2(101111) = row 3, col 7 entry of S_2 = 2 (2)
S_3(100101) = row 3, col 2 entry of S_3 = 13 (D)
S_4(000011) = row 1, col 1 entry of S_4 = 8 (8)
S_5(110001) = row 3, col 8, entry of S_5 = 6 (6)
S_6(100010) = row 2, col 1, entry of S_6 = 14 (E)
S_7(110110) = row 2, col 11, entry of S_7 = 8 (8)
S_8(100001) = row 3, col 0, entry of S_8 = 2 (2)

Output in HEX is **C2D86E82.**

Grading: 1 pt per HEX char no exceptions

2) (7 pts) What is the result of the multiplication 03 x B6, in the AES field? In order to get credit, you must show all of your work by hand.

Solution

03 x B6 = 02 x B6 + 01 x B6

02 x B6 = 1 0110 1100 01 x B6 = 1011 0110

= 0110 1100
+ 1 1011

0111 0111

Adding we get 0111 0111
+ 1011 0110

1100 0001, in HEX we have **C1.**

Grading: 1 pt for listing out bits on 01 x B6, 4 pts for doing the 02 x B6 multiplication (1 pt for shift, 2 pts for 11011, 1 pt for answer), 2 pts for the final XOR.

3) (10 pts) The code included below has produced the following output:

```
Gpqef3Yx8/+0LTrq4nmlgrj
```

What was the string entered by the user that produced this output?

To answer the question**, you may run and edit this code as you see fit.** Alternatively, you may solve the problem by hand. There are reliable ways to do either but the former is almost definitely faster. Regardless of which method you choose, explain what you did and explain why it works.

Solution

What the code does is take the input, convert the regular ascii characters to their corresponding integer RADIX-64 values, XOR that integer with the position of the integer mod 64 (this is what i&63 really does, it isolates the last 6 bits of the variable i, which is mathematically equivalent to mod 64). Then that resulting integer is converted back to its equivalent RADIX-64 character. Thus, this is like a block cipher with 6 bit blocks where each block is XORed with a fixed key based on the letter position. To undo an XOR, you just XOR it with the same value, since each bit will get canceled with itself in the second XOR, revealing the plaintext. So, it's good enough to run this same code, but enter in the ciphertext, to reveal the plaintext, which is:

Goodbye2020/Hello2020+1

Grading: 8 pts for the answer any which way, 2 pts for explaining that to undo an XOR, you just XOR with the same thing.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* makeCode(char* word);
char intToChar(int n);
int charToInt(char c);

int main(void) {
    char* ptr = malloc(25);
    scanf("%s", ptr);
    char* res = makeCode(ptr);
    printf("%s\n", res);
    free(ptr);
    free(res);
    return 0;
}

char* makeCode(char* word) {
    int n = strlen(word);
    char* res = malloc(sizeof(char)*(n+1));
    for (int i=0; i<n; i++)
        res[i] = intToChar(charToInt(word[i])^(i&63));
    res[n] = '\0';
    return res;
}
```

```
char intToChar(int n) {
    if (n < 26) return 'A'+n;
    if (n < 52) return 'a'+n-26;
    if (n < 62) return '0'+n-52;
    if (n == 62) return '+';
    return '/';
}

int charToInt(char c) {
    if (c >= 'A' && c <= 'Z') return c - 'A';
    if (c >= 'a' && c <= 'z') return c - 'a' + 26;
    if (c >= '0' && c <= '9') return c - '0' + 52;
    if (c == '+') return 62;
    return 63;
}
```