Instituto Politécnico Nacional
Escuela Superior de Cómputo
Cryptography

Practice 2: Affine Cipher

By:
Moreno Zárate Víctor Gibrán

Professor:
M. en C. NIDIA ASUNCIÓN CORTEZ DUARTE
September 2016

Contents

Problem

What is the behavior of the affine cipher?

What are the main characteristics of the affine cipher?

How does the output behave according to the input and the key?

Affine cipher is a type of monoalphabetic substitution cipher who uses modular arithmetic to encrypt and decrypt messages. [1]

These functions are:

$$E(x) = (ax + b)\ mod\ m$$

$$D(x) = a^{-1}(x + (-b))\ \ mod\ m$$

Where “a” is a coprime number with “m” (number of elements in the alphabet), a^{-1} is the multiplicative inverse, and -b is the additive inverse.

Two numbers are coprime if gcd (a, m) =1. This ensure the existence of all the inverses.

Hypothesis

A possible solution to the problem is to write a computer program in order to analyze the behavior of the affine cipher, this program will be able to encrypt and decrypt a text using this technique. It also will validate the entered alpha and find the multiplicative and additive inverses.

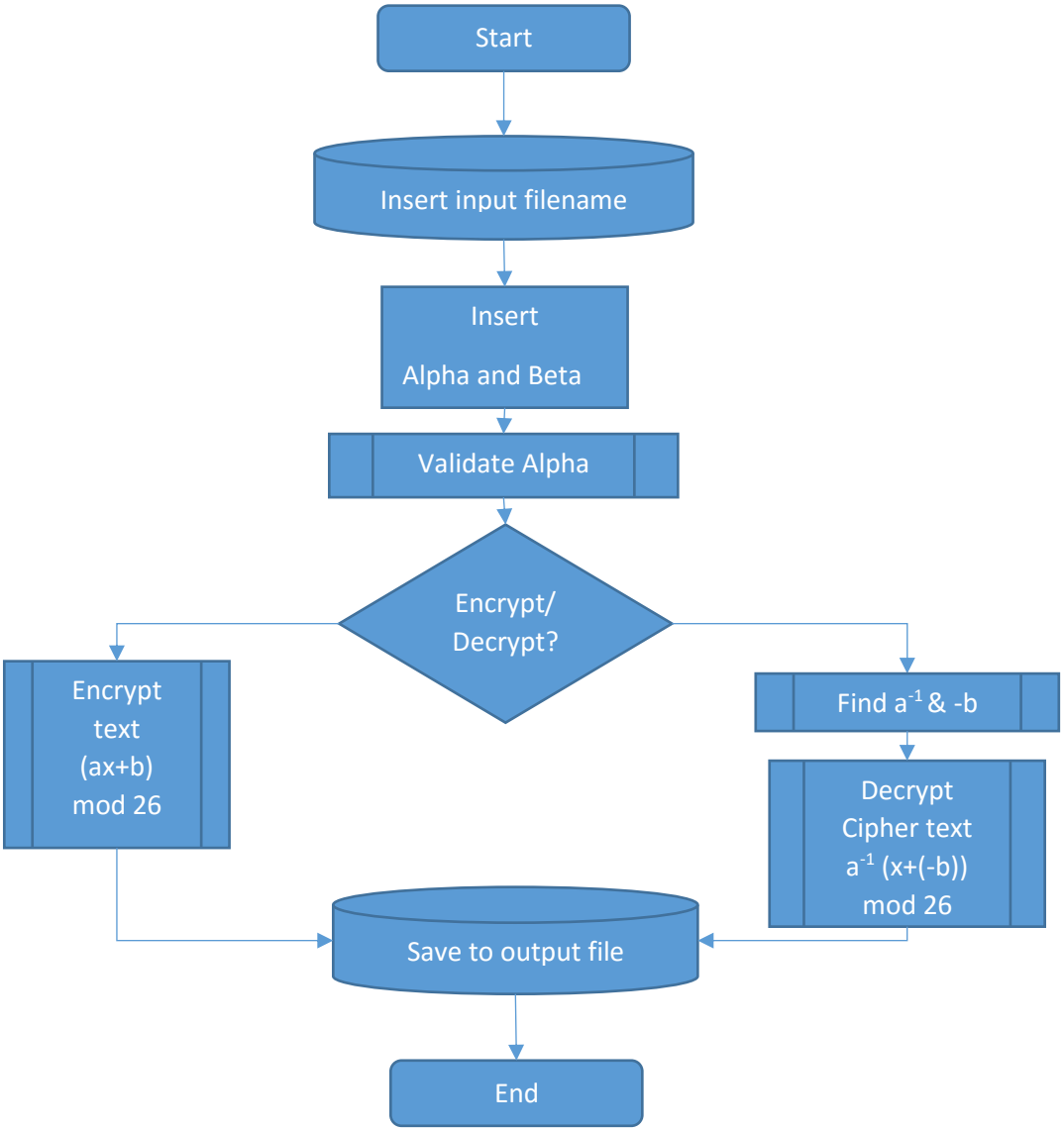
If we program this software correctly, we will see how the character data changes according to the given formulas. Then we will observe the encrypt/decrypt of the cipher text if we put the correct key.

Software (libraries, package, tools)

In order to do this practice, we used:

- Personal Computer
- Linux Operating System
- Text Editor
- GNU C Compiler

Procedure



The program asks to the user to input the file name of operation, alpha and beta values to use, and the operation to perform (encrypt/decrypt), then calls the correct procedure (encrypt or decrypt) and stores the output file with the processed data. It also validates if the file name exists, if it is possible to open/write the file and if the inserted alpha is valid (being coprime with the alphabet in this case 26).

In case that the user wants to decrypt, the program has to calculate a^{-1} & -b. The first version of the program does it by using brute force, the second version uses the Extended Euclidean algorithm to find them.

Results (Data)

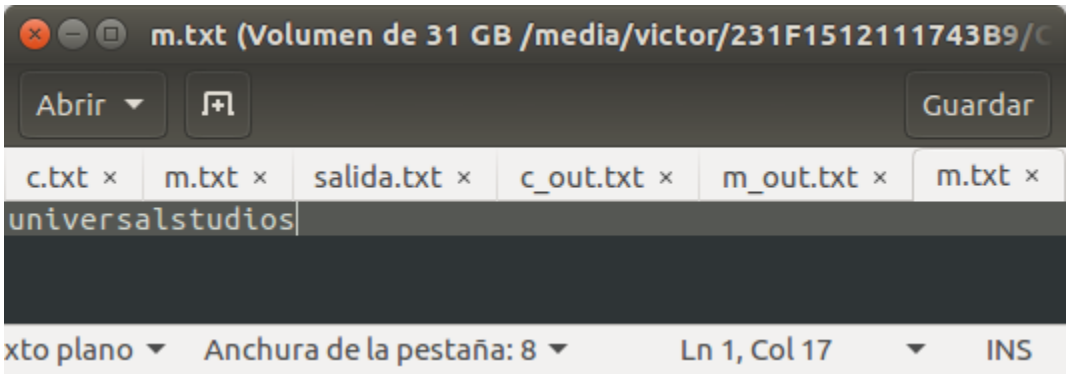


Figure 1: Image of the plain text

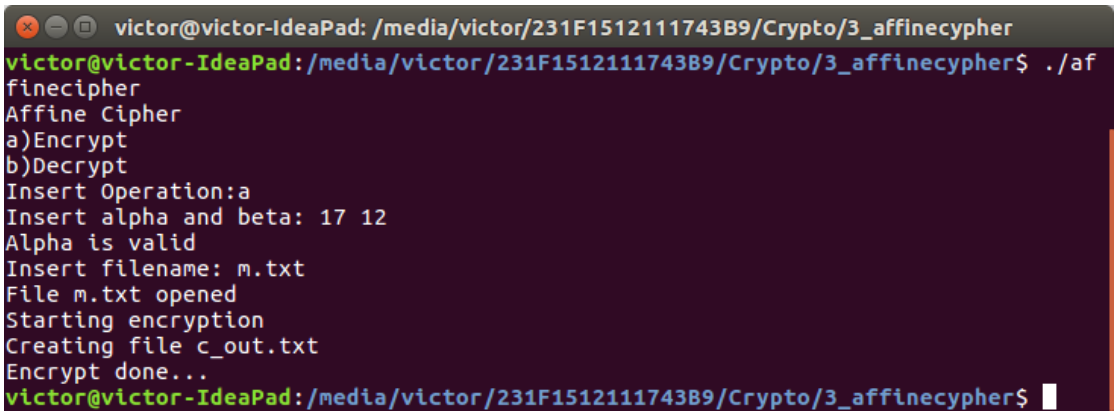


Figure 2: Encryption process

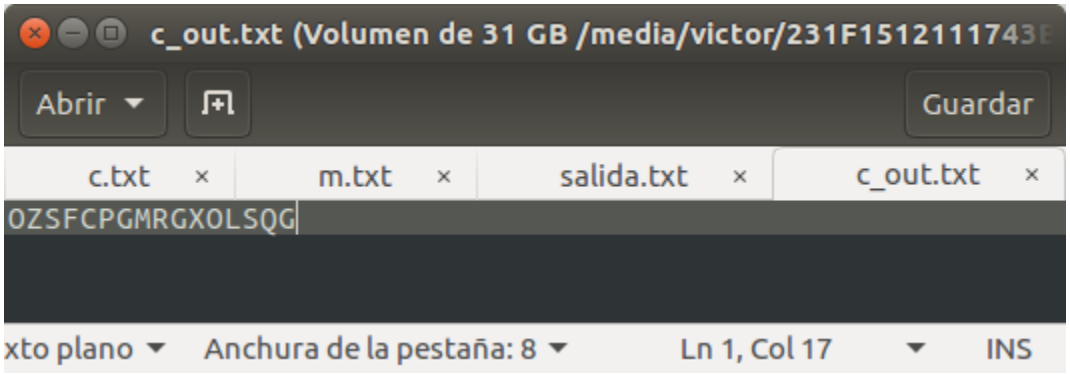


Figure 3. Encrypted cipher text

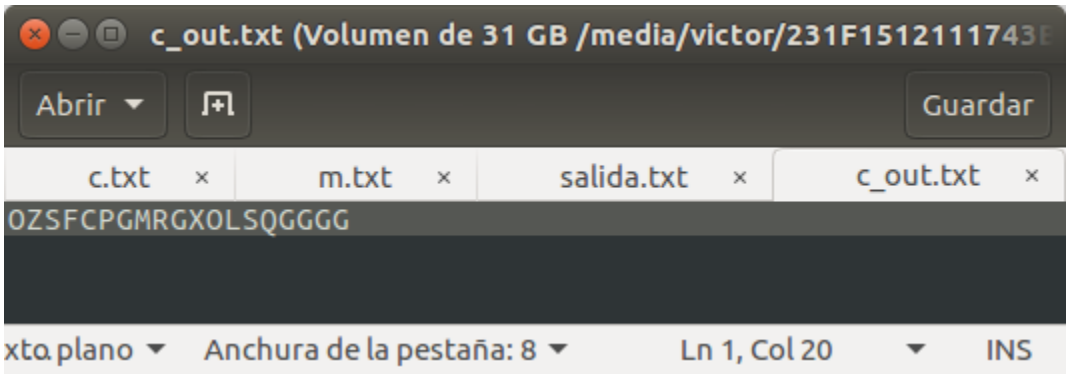


Figure 4. Adding extra data at the end of the string

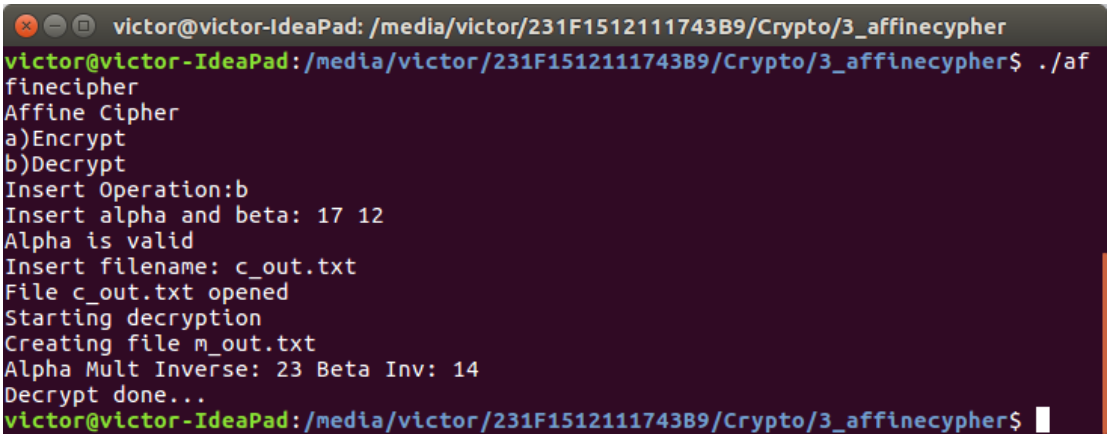


Figure 5. Decryption process

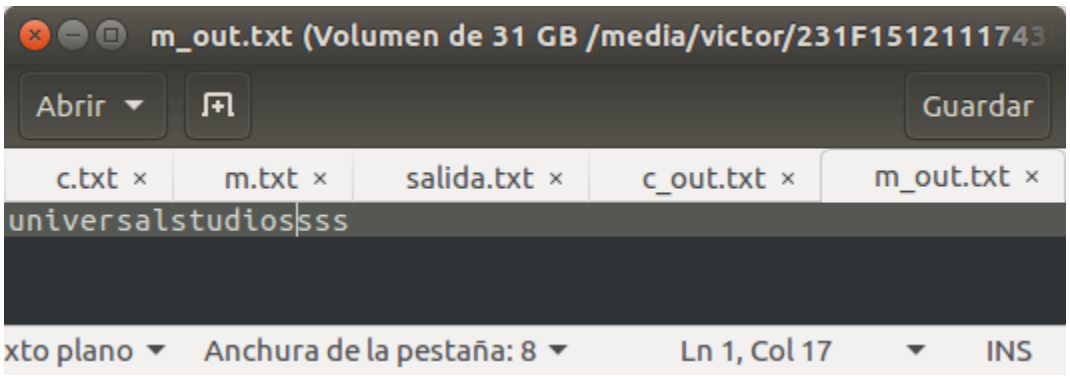


Figure 6. Decrypted data

Conclusions

In this practice, I accepted the hypothesis because I was able to observe the behavior of the Affine cipher, and in the program I was able to encrypt and decrypt the message with the correct key, also if we put extra characters of the same letter at the end of the cipher text, it starts to repeat the same letter. This because it's a monoalphabetic cipher method.

I learn the use and implementation of another basic cryptographic method, which includes more math knowledge, and it's an introduction to more complex methods. It can be used in real life to encrypt/decrypt simple plain texts.

Reference

[1] "Affine cipher", *Wikipedia,* 2016. [Online]. Available: https://en.wikipedia.org/wiki/Affine_cipher . [Accessed: 13- Sep- 2016].

Code

Without Euclidean Algorithm

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <ctype.h>
4.  int  isValidAlpha(int alpha);
5.  int gcd(int a, int b);
6.  void encrypt(FILE *file,int alpha,int beta,char *letters);
7.  void decrypt(FILE *file,int alpha,int beta,char *letters);
8.  int findMultInv(int alpha);
9.  int findAddInv(int beta);
10.
11. int main(){
12.     char letters_lowercase[]={'a','b','c','d','e','f','g','h','i','j','k','l','m',
    'n','o','p','q','r','s','t','u','v','w','x','y','z'};
13.     char letters_uppercase[]={'A','B','C','D','E','F','G','H','I','J','K','L','M',
    'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
14.     char opt;
15.     int alpha,beta;
16.     printf("Affine Cipher\n");
17.     printf("a)Encrypt\nb)Decrypt\n");
18.     printf("Insert Operation:");
19.     scanf("%c",&opt);
20.     printf("Insert alpha and beta: ");
21.     scanf("%d %d",&alpha,&beta);
22.
23.     //Alpha Validation
24.     if(!isValidAlpha(alpha)){
25.             printf("Alpha is not valid.Exiting\n");
26.             return 0;
27.     }
28.
29.     //File validation
30.     printf("Alpha is valid\nInsert filename: ");
31.     char filename[256];
32.     scanf("%s",filename);
33.     FILE *file;
34.     file = fopen(filename, "r");
```

```
    if(file!=NULL){
        printf("File %s opened\n",filename);
    }else{
        perror("Fail to open file:");
        return 0;
    }

    //Start of encrypt-decrypt
    if(opt=='a'){
        printf("Starting encryption\n");
        encrypt(file,alpha,beta,letters_uppercase);

    }else if(opt=='b'){
        printf("Starting decryption\n");
        decrypt(file,alpha,beta,letters_lowercase);

    }else{
        printf("Invalid Option\n");
        return 0;
    }

    return 0;
}

int isValidAlpha(int alpha){
    return gcd(26,alpha)==1;
    }

int gcd(int a, int b){
    int t;
    while (b != 0){
       t = b;
       b = a % b;
       a = t;
    }
    return a;
}


void encrypt(FILE *file,int alpha,int beta,char *letters){
    //Creating file c.txt
    printf("Creating file c_out.txt\n");
    FILE *output_file;
    output_file= fopen("c_out.txt","w");
    if (output_file==NULL){
        perror("Can't create file");
        exit(0);
        }

    int caracter,aux;
    while((caracter=fgetc(file))!=EOF){
        if(isalpha(caracter)){
            aux=letters[(alpha*(caracter-97)+beta)%26];
            fputc(aux,output_file);
            }
        }
    printf("Encrypt done...\n");
    return;
}

void decrypt(FILE *file,int alpha,int beta,char *letters){
```

```
96.     printf("Creating file m_out.txt\n");
97.     FILE *output_file;
98.     output_file= fopen("m_out.txt","w");
99.     if (output_file==NULL){
100.                perror("Can't create file");
101.                exit(0);
102.                }
103.
104.           int caracter,aux;
105.           int alpha_MultInv=findMultInv(alpha);
106.           int beta_AddInv=findAddInv(beta);
107.           printf("Alpha Mult Inverse: %d Beta Inv: %d\n",alpha_MultInv,beta_AddI
    nv);
108.           while((caracter=fgetc(file))!=EOF){
109.               if(isalpha(caracter)){
110.                   aux=letters[(alpha_MultInv*((caracter-65)+beta_AddInv))%26];
111.                   fputc(aux,output_file);
112.                   }
113.               }
114.           printf("Decrypt done...\n");
115.
116.
117.           }
118.
119.       int findMultInv(int alpha){
120.               int i;
121.               for(i=1;i<26;i++){
122.                   if((alpha*i)%26==1)
123.                       return i;
124.               }
125.               return 0;
126.           }
127.
128.       int findAddInv(int beta){
129.               return 26-beta;
130.
131.           }
```

With Euclidian Algorithm

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <ctype.h>
4.  int  isValidAlpha(int alpha);
5.  int gcd(int a, int b);
6.  void encrypt(FILE *file,int alpha,int beta,char *letters);
7.  void decrypt(FILE *file,int alpha,int beta,char *letters);
8.  int findMultInv(int alpha);
9.  int findAddInv(int beta);
10. int invert_mod(int a, int p) ;
11.
12. int main(){
13.     char letters_lowercase[]={'a','b','c','d','e','f','g','h','i','j','k','l','m',
    'n','o','p','q','r','s','t','u','v','w','x','y','z'};
14.     char letters_uppercase[]={'A','B','C','D','E','F','G','H','I','J','K','L','M',
    'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
15.     char opt;
16.     int alpha,beta;
17.     printf("Affine Cipher\n");
18.     printf("a)Encrypt\nb)Decrypt\n");
```

```
19.     printf("Insert Operation:");
20.     scanf("%c",&opt);
21.     printf("Insert alpha and beta: ");
22.     scanf("%d %d",&alpha,&beta);
23.
24.     //Alpha Validation
25.     if(!isValidAlpha(alpha)){
26.             printf("Alpha is not valid.Exiting\n");
27.             return 0;
28.     }
29.
30.     //File validation
31.     printf("Alpha is valid\nInsert filename: ");
32.     char filename[256];
33.     scanf("%s",filename);
34.     FILE *file;
35.     file = fopen(filename, "r");
36.     if(file!=NULL){
37.         printf("File %s opened\n",filename);
38.     }else{
39.         perror("Fail to open file:");
40.         return 0;
41.     }
42.
43.     //Start of encrypt-decrypt
44.     if(opt=='a'){
45.         printf("Starting encryption\n");
46.         encrypt(file,alpha,beta,letters_uppercase);
47.
48.     }else if(opt=='b'){
49.         printf("Starting decryption\n");
50.         decrypt(file,alpha,beta,letters_lowercase);
51.
52.     }else{
53.         printf("Invalid Option\n");
54.         return 0;
55.     }
56.
57.     return 0;
58. }
59.
60. int isValidAlpha(int alpha){
61.     return gcd(26,alpha)==1;
62.     }
63.
64. // Implementation found in: https://es.wikibooks.org/wiki/Implementaci%C3%B3n_de_a
    lgoritmos_de_teor%C3%ADa_de_n%C3%BAmeros/Algoritmo_de_Euclides#En_lenguaje_C
65. int gcd(int a, int b){
66.     int t;
67.     while (b != 0){
68.        t = b;
69.        b = a % b;
70.        a = t;
71.     }
72.     return a;
73. }
74.
75.
76. void encrypt(FILE *file,int alpha,int beta,char *letters){
77.     //Creating file c.txt
78.     printf("Creating file c_out.txt\n");
```

```
79.     FILE *output_file;
80.     output_file= fopen("c_out.txt","w");
81.     if (output_file==NULL){
82.         perror("Can't create file");
83.         exit(0);
84.         }
85.
86.     int caracter,aux;
87.     while((caracter=fgetc(file))!=EOF){
88.         if(isalpha(caracter)){
89.             aux=letters[(alpha*(caracter-97)+beta)%26];
90.             fputc(aux,output_file);
91.             }
92.         }
93.     printf("Encrypt done...\n");
94.     return;
95. }
96.
97. void decrypt(FILE *file,int alpha,int beta,char *letters){
98.     printf("Creating file m_out.txt\n");
99.     FILE *output_file;
100.            output_file= fopen("m_out.txt","w");
101.            if (output_file==NULL){
102.                perror("Can't create file");
103.                exit(0);
104.                }
105.
106.            int caracter,aux;
107.            int alpha_MultInv=invert_mod(alpha,26);
108.            int beta_AddInv=findAddInv(beta);
109.            printf("Alpha Mult Inverse: %d Beta Inv: %d\n",alpha_MultInv,beta_AddI
    nv);
110.            while((caracter=fgetc(file))!=EOF){
111.                if(isalpha(caracter)){
112.                    aux=letters[(alpha_MultInv*((caracter-65)+beta_AddInv))%26];
113.                    fputc(aux,output_file);
114.                    }
115.                }
116.            printf("Decrypt done...\n");
117.
118.
119.            }
120.
121.
122.        int findAddInv(int beta){
123.                return 26-beta;
124.
125.            }
126.
127.        //Implementation found in : https://en.wikipedia.org/wiki/Extended_Euclide
    an_algorithm#Simple_algebraic_field_extensions
128.        int invert_mod(int a, int p) {
129.            int new = 1, old = 0, q = p, r, h;
130.            int pos = 0;
131.            while (a > 0) {
132.                r = q%a;
133.                q = q/a;
134.                h = q*new + old;
135.                old = new;
136.                new = h;
137.                q = a;
```

```
138.                 a = r;
139.                 pos = !pos;
140.             }
141.             return pos ? old : (p - old);
142.         }
```

Seal of approval