

Classical Cryptography

Miranda Mojica Erick

03 Febrero 2020

Primera practica de laboratorio. Se trabajo con algoritmos de cifrado clásico como lo son Vigenere cipher y Affine cipher. En mi caso hice uso de clases para llevar acabo la practica, asi que para los punto 1,2,5,6 son métodos de una clase. Para el caso de de los puntos 3, 4 cree métodos de clase que puedan ser usados globalmente y sin instanciar. Todos los puntos los uní en un solo programa pero con la posibilidad de reutilizar el código para futuros usos.

Clase Vigenere

```
1 #ifndef VIGENERE_H_
2 #define VIGENERE_H_
3
4 #include "Cipher.h"
5
6 class Vigenere: public Cipher{
7
8     protected:
9         string key;
10        string file;
11    public:
12        Vigenere(string , string , string);
13        Vigenere(string , string);
14        void encrypt();
15        void decrypt();
16
17 };
18
19
20 #endif
```

La clase Vigenere hereda de Cipher, una clase base que tiene lo básico para los algoritmos clásicos.

1 Design a function to encrypt using the Vigenère cipher. The function must receive the plaintext, and the key. The output must be the ciphertext.

```
1 void Vigenere::encrypt(){
2     for(int i = 0; i < menssage.length(); i++)
3         cMensaje.push_back(alphabet[(alphabet.find_first_of(menssage[i]) + alphabet.find_first_of(key[
4             i % key.length()])) % alphabet.length()]);
5 }
```

La función *encrypt* es un método de la clase Vigenere, por lo cual no se pasan las variables puesto que estas ya forman parte de la clase. La clase toma el mensaje sin cifrar, tomando el valor correspondiente de cada carácter y sumado con el carácter correspondiente de la cadena de la llave.

```

message.txt  message.vig
1 | HOLA MUNDO  1 | PCXBPCXBPCXB

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

erimimo@DESKTOP-QOSEVD9:/mnt/c/Users/Eri_m/OneDr
systems$ make run_vig_en
./main alphabet.txt message.txt 1
¿Random key?(y/n): n
Introduce la llave(Cadena): HOLA

```

Figure 1: Cifrado con Vigenere

En esta función podemos elegir una llave aleatoria o una llave propuestas por nosotros. En este caso proponemos la llave y del lado derecho tenemos el mensaje cifrado. El mensaje cifrado es guardado en un archivo con extensión ".vig".

2 Design a function to decrypt using the Vigenère cipher. The function must receive the cipher-text, and the key. The output must be the plaintext.

```

1  for(int i = 0; i < cMensaje.length(); i++)
2      mensaje.push_back(alphabet[mod(alphabet.find_first_of(cMensaje[i]) - alphabet.find_first_of(
3      key[i % key.length()]), alphabet.length())]);

```

La función *decrypt* es un método de la clase Virgenere, por lo cual no se pasan las variables puesto que estas ya forman parte de la clase. Muy parecido al algoritmo de cifrado, tomado el valor de cada carácter del mensaje, es restado con el carácter correspondiente de la cadena de la llave.

```

message.txt  message.vig
1 | HOLA MUNDO  1 | PCXBPCXBPCXB

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

erimimo@DESKTOP-QOSEVD9:/mnt/c/Users/Eri_m/OneDr
systems$ make run_vig_de
./main alphabet.txt message.vig 3
Introduce la llave(Cadena): HOLA

```

Figure 2: Descifrado con Vigenere

3 Design a function to verify that a candidate key for the affine cipher is a valid key.

```

1  int Affine::modInverse(int a, int m){
2      a = mod(a, m);
3      int x, y;
4      int g = gcdExtended(a, m, &x, &y);

```

```

5     if (g != 1)
6         return -1;
7     else
8         // m is added to handle negative x
9         return (x % m + m) % m;
10 }
11
12 int Affine::gcdExtended(int a, int b, int *x, int *y){
13
14     if (a == 0){
15         *x = 0, *y = 1;
16         return b;
17     }
18
19     int x1, y1;
20     int gcd = gcdExtended(b % a, a, &x1, &y1);
21
22     *x = y1 - (b / a) * x1;
23     *y = x1;
24
25     return gcd;
26 }

```

Para este punto hacemos uso de dos funciones, básicamente se trata de llamar la función de modulo inverso y esperar que regrese el inverso del modulo para saber que la llave es valida.

4 Design a function that receives a valid key for the affine cipher $K = (a,b)$ and n and calculate $a^{-1} \bmod n$.

```

1 int Affine::modInverse(int a, int m){
2     a = mod(a, m);
3     int x, y;
4     int g = gcdExtended(a, m, &x, &y);
5     if (g != 1)
6         return -1;
7     else
8         // m is added to handle negative x
9         return (x % m + m) % m;
10 }

```

Con ayuda del algoritmo extendido de Euclides calculamos el modulo inverso de $a \bmod n$.

Clase Affine

```

1 #ifndef AFFINE_H_
2 #define AFFINE_H_
3
4 #include "Cipher.h"
5
6 class AffineKey{
7     public:
8         int a, b;
9 };
10
11 class Affine: public Cipher{
12
13     protected:
14         AffineKey key;
15         string file;
16     public:
17         Affine(AffineKey, string, string);
18         Affine(AffineKey, string);
19         static int modInverse(int, int);
20         static int gcdExtended(int, int, int*, int*);
21         void encrypt();
22         void decrypt();

```

```

23     };
24 };
25
26 #endif

```

La clase *Affine* hereda de *Cipher*, una clase base que tiene lo básico para los algoritmos clásicos.

5 Design a function to encrypt using the affine cipher. The function must receive the plaintext, and a valid key. The output must be the ciphertext.

```

1 void Affine::encrypt(){
2     for(int i = 0; i < message.length(); i++)
3         cMessage.push_back(alphabet[mod(((alphabet.find_first_of(message[i]) * key.a) + key.b),
4                                     alphabet.length())]);

```

La función *encrypt* es un método de la clase *Affine*, por lo cual no se pasan las variables puesto que estas ya forman parte de la clase. La clase toma el mensaje sin cifrar, tomando el valor correspondiente de cada carácter y aplicando el algoritmo de Affine cipher.

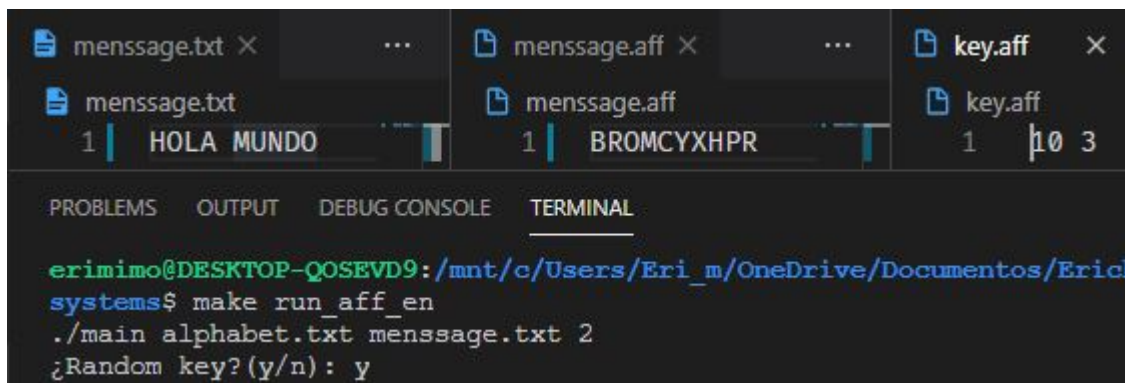


Figure 3: Cifrado con Affine

En este programa podemos elegir una llave aleatoria valida para cifrar el mensaje de la izquierda, al final la llave es mandada a un archivo de nombre "key.aff" y el mensaje cifrado a un archivo con extensión ".aff".

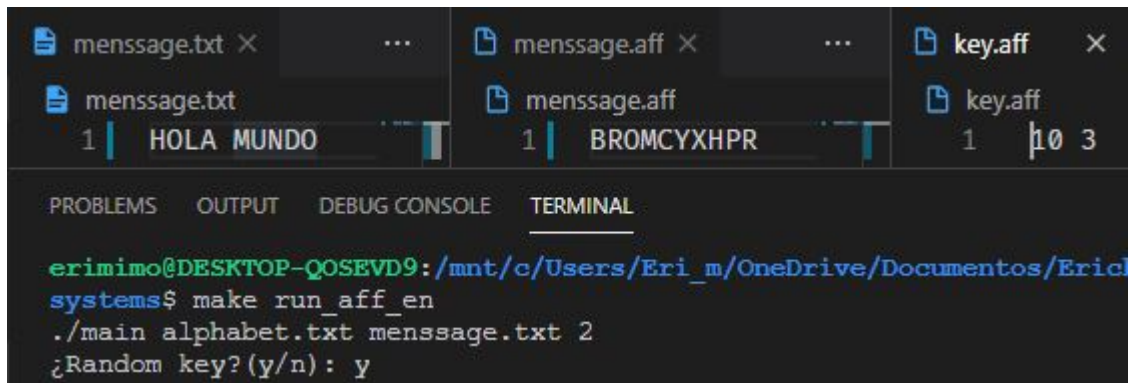
6 Design a function to decrypt using the affine cipher. The function must receive the ciphertext, and the key. The output must be the plaintext.

```

1 void Affine::decrypt(){
2     for(int i = 0; i < cMessage.length(); i++)
3         message.push_back(alphabet[mod(((alphabet.find_first_of(cMessage[i]) - key.b) * modInverse(
4             key.a, alphabet.length()))), alphabet.length())]);

```

La función *decrypt* es un método de la clase *Virgenere*, por lo cual no se pasan las variables puesto que estas ya forman parte de la clase. Muy parecido al algoritmo de cifrado, tomado el valor de cada carácter del mensaje y aplicando el algoritmo de descifrado de Affine cipher.



```
message.txt 1 | HOLA MUNDO
message.aff 1 | BROMCYXHPR
key.aff 1 | 10 3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

erimimo@DESKTOP-QOSEVD9:/mnt/c/Users/Eri_m/OneDrive/Documentos/Ericksystems$ make run_aff_en
./main alphabet.txt message.txt 2
¿Random key? (y/n): y
```

Figure 4: Descifrado con Affine

Introducimos la llave aleatoria creada en el punto anterior y el mensaje descifrado es el que se ve a la izquierda.