



Alumno:	García González Aarón Antonio
Grupo:	3CM3
Unidad de Aprendizaje:	Crptography
Profesora:	Cortez Duarte Nidia Asunción
Practica #1:	Algoritmo Vigene, Affine y de Euclides extendido.
Fecha:	Viernes 30 de octubre de 2020.

## Resumen

A lo largo de esta primera práctica se realizó la codificación de los algoritmos de cifrado y descifrado Vigenere y Affine vistos en las sesiones teóricas previamente, en el caso de Affine se utiliza el algoritmo de Euclides extendido para verificar que  $\alpha$  y la longitud del alfabeto son números coprimos y, por ende, existirá un inverso multiplicativo (lógica modular) de  $\alpha$  y así un correcto descifrado.

Se utiliza el código ASCII como alfabeto base, donde el usuario puede seleccionar un número menor o igual que 256 para la longitud del alfabeto que desea cubrir, siempre considerando que dicho número seleccionado cubra todos los caracteres utilizados en la llave para el caso del algoritmo Vigenere, de lo contrario se observa que hay errores.

## Índice

<b>1. Probar AEE .....</b>	<b>2</b>
<b>2. Aritmética modular (Códigos).....</b>	<b>3</b>
a) Inverso aditivo .....	3
b) Inverso multiplicativo .....	4
<b>3. Algoritmo de Euclides y Euclides extendido (Explicación) .....</b>	<b>4</b>
<b>4. Interfaces de selección .....</b>	<b>5</b>
a) Index .....	5
b) Vigenere cifrado .....	5
c) Affine cifrado .....	6
d) Affine descifrado.....	7
<b>5. Cifrador Vigenere .....</b>	<b>7</b>
a) Explicación.....	7
b) Código cifrador .....	8
c) Código descifrador .....	8
d) Prueba .....	9
<b>6. Cifrador Affine.....</b>	<b>10</b>
a) Explicación.....	10
b) Código cifrador .....	10
c) Código descifrador .....	11
d) Pruebas.....	11
<b>7. Conclusiones .....</b>	<b>14</b>
<b>8. Tabla de cotejo.....</b>	<b>14</b>

## 1. Probar AEE

Dado que la interfaz de usuario solicitada esta en el nivel de abstracción mas alto, no incluye la validación directa de los parámetros  $\alpha$  y  $n$ , por lo que exclusivamente esta parte la realizaremos desde consola. Probar con:

- $\alpha = 5, n = 30$

```
int  $\alpha$  = 5;
int n = 20;
int mcd = AritmeticaModular.mcd( $\alpha$ , n)[0];
int inverso_multiplicativo = AritmeticaModular.mcd( $\alpha$ , n)[1];

if (mcd != 1){
    System.out.println("mcd(" +  $\alpha$  + ", " + n + ") = " + mcd);
    System.out.println("Dado lo anterior, " +  $\alpha$  + ", no tiene inverso multiplicativo en " + n);
}else{
    System.out.println("mcd(" +  $\alpha$  + ", " + n + ") = " + mcd);
    System.out.println("Inverso multiplicativo de " +  $\alpha$  + " = " + inverso_multiplicativo);
}
```

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(5, 20) = 5
Dado lo anterior, 5, no tiene inverso multiplicativo en 20

Process finished with exit code 0
```

- $\alpha = 97, n = 239$

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(97, 239) = 1
Inverso multiplicativo de 97 = 69

Process finished with exit code 0
```

- $\alpha = 11111, n = 12345$

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(11111, 12345) = 1
Inverso multiplicativo de 11111 = 2471

Process finished with exit code 0
```

- $a = 13, n = 99991$

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(13, 99991) = 1
Inverso multiplicativo de 13 = -38458

Process finished with exit code 0
```

- $a = 10009, n = 104729$

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(100009, 104729) = 1
Inverso multiplicativo de 100009 = -44177

Process finished with exit code 0
```

## 2. Aritmética modular (Códigos)

### a) Inverso aditivo

```
public static int InversoAditivo(int numero, int longitud_de_alfabeto)
{
    if(numero > 0) {
        return (longitud_de_alfabeto - numero);
    }
    else {
        return longitud_de_alfabeto + numero;
    }
}
```

### a) Máximo común divisor

```
1. /* Regresa array[d,a,b] donde, d = gcd(p,q), ap + bq = d
2.     donde a, de ser negativo no calcula el inverso aditivo.
3.     */
4. public static int[] mcd(int p, int q) {
5.     if(q==0) {
6.         return new int[] {p,1,0};
7.     }
8.     int [] vals = mcd(q, p%q);
9.     int d = vals[0];
10.    int a = vals[2];
11.    int b = vals[1] - (p/q) * vals[2];
12.
13.    return new int []{d,a,b};
14. }
```

## b) Inverso multiplicativo

```
1. public static int InversoMultiplicativo(int p, int n)
2. {
3.     int inverso_multiplicativo = mcd(p, n)[1];
4.
5.     if(inverso_multiplicativo < 0)
6.         return InversoAditivo(inverso_multiplicativo, n);
7.     else
8.         return inverso_multiplicativo;
9. }
```

## 3. Algoritmo de Euclides y Euclides extendido (Explicación)

El algoritmo de Euclides es un método eficiente de calcular el máximo común divisor entre dos números, este algoritmo no es muy eficiente de implementar en computadora, ya que se tienen que memorizar todos los valores de residuo.

### Algoritmo 1 de Euclides

**Entrada:** Valores  $a$  y  $b$  pertenecientes a un dominio euclídeo

**Salida:** Un máximo común divisor de  $a$  y  $b$

1.  $r_0 \leftarrow a, r_1 \leftarrow b$
2.  $i \leftarrow 1$
3. **Mientras**  $r_i \neq 0$  **haga lo siguiente:**
  1.  $r_{i+1} \leftarrow r_{i-1} \bmod r_i$
  2.  $i \leftarrow i + 1$
4. **El resultado es:**  $r_{i-1}$

Por otro lado, el Algoritmo de Euclides extendido, es una ligera modificación que permite además expresar al máximo común divisor como una combinación lineal, mediante las combinaciones lineales, al despejar los residuos nos permite obtener el inverso multiplicativo, y al mismo tiempo conocer el mcd.

### Algoritmo 2 de Euclides extendido

**Entrada:** Valores  $a$  y  $b$  pertenecientes a un dominio euclídeo

**Salida:** Un máximo común divisor de  $a$  y  $b$ , y valores  $s$  y  $t$  tales que  $\text{mcd}(a, b) = as + bt$

1.  $r_0 \leftarrow a, r_1 \leftarrow b, s_0 \leftarrow 1, t_0 \leftarrow 0, s_1 \leftarrow 0, t_1 \leftarrow 1$
2.  $i \leftarrow 1$
3. **Mientras**  $r_i \neq 0$  **haga lo siguiente:**
  1. Divida  $r_{i-1}$  entre  $r_i$  para obtener el cociente  $q_i$  y el residuo  $r_{i+1}$
  2.  $s_{i+1} \leftarrow s_{i-1} - q_i s_i$
  3.  $t_{i+1} \leftarrow t_{i-1} - q_i t_i$
  4.  $i \leftarrow i + 1$
4. **El resultado es:**  $r_{i-1}$  es un máximo común divisor de  $a$  y  $b$  y se expresa  $r_{i-1} = as_{i-1} + bt_{i-1}$

## 4. Interfaces de selección

### a) Index



The screenshot shows a web browser at localhost:8080/index. The page has a green header with the title "Cryptography 3CM3" and the following text: "Profesor - Nidia Asunción Cortez Duarte", "Práctica 1 - Cifrador Vegener, Affine y Aritmética modular", and "Alumno - Aarón Antonio García González". Below the header is a section titled "Inicio". The main content area is titled "Seleccionar operación" and contains two columns. The left column is titled "Vigenere" and has two buttons: "Cifrar" (red border) and "Descifrar" (blue border). The right column is titled "Affine" and also has two buttons: "Cifrar" (red border) and "Descifrar" (blue border).

### b) Vigenere cifrado



The screenshot shows a web browser at localhost:8080/encrypt\_vigenere. The page has a pink header with the title "Cryptography 3CM3" and the following text: "Profesor - Nidia Asunción Cortez Duarte", "Práctica 1 - Cifrador Vegener, Affine y Aritmética modular", and "Alumno - Aarón Antonio García González". Below the header is a section titled "Cifrador Vigenere". The main content area contains a form with the following elements: "Archivo" with a "Choose File" button and "No file chosen" text; "Longitud de alfabeto" with a text input field containing "Longitud de alfabeto"; "Password" with a text input field containing "Password"; and another "Password" label with a "Random" checkbox. At the bottom is a large red button labeled "Cifrar".

a) Vigenere descifrado

← → ↻ ⓘ localhost:8080/decrypt\_vigenere 🔍 ☆ 📄 🌐 ASP 🔴 📺 ⚙️ 👤 ⋮

## Cryptography 3CM3

Profesor - Nidia Asunción Cortez Duarte

Práctica 1 - Cifrador Vegener, Affine y Aritmética modular

Alumno - Aarón Antonio García González

### Descifrador Vigenere

Archivo  No file chosen

Longitud de alfabeto

Password

c) Affine cifrado

← → ↻ ⓘ localhost:8080/encrypt\_affine 🔍 ☆ 📄 🌐 ASP 🔴 📺 ⚙️ 👤 ⋮

## Cryptography 3CM3

Profesor - Nidia Asunción Cortez Duarte

Práctica 1 - Cifrador Vegener, Affine y Aritmética modular

Alumno - Aarón Antonio García González

### Cifrador Affine

Archivo  No file chosen

Longitud de alfabeto

Alpha

Beta

Alpha y Beta ☐ Random

## d) Affine descifrado

← → ↻ localhost:8080/decrypt\_affine

# Cryptography 3CM3

Profesor - Nidia Asunción Cortez Duarte

Práctica 1 - Cifrador Vegenere, Affine y Aritmética modular

Alumno - Aarón Antonio García González

## Descifrador Affine

Archivo  No file chosen

Longitud de alfabeto

Alpha

Beta

## 5. Cifrador Vigenere

### a) Explicación

Este es un algoritmo de sustitución poli alfabético, esto debido a que una misma letra es posible mapearse o cifrarse a más de una en común. El funcionamiento de este algoritmo es muy sencillo, para cifrar lo que se tiene que hacer es:

1. Calcular la longitud del texto en claro
2. Repetir la llave hasta hacer coincidir con la longitud del texto en claro
3. Comparar posición a posición de cada carácter tanto del texto en claro como de la llave obtenida en el paso #2, y sumar los valores
4. A cada número obtenido en #3, aplicar modulo longitud del alfabeto
5. Mapear cada número al valor correspondiente con el alfabeto

Para descifrar, lo que se hace es:

6. Calcular el inverso aditivo de cada carácter de la llave
7. Con base en la llave inverso aditivo obtenida en #6, repetir la llave hasta hacer coincidir con la longitud del texto cifrado
8. Comparar posición a posición de cada carácter tanto del texto cifrado como de la llave obtenida en el paso #7, y sumar los valores
9. A cada número obtenido en #8, aplicar modulo longitud del alfabeto
10. Mapear cada número al valor correspondiente con el alfabeto

## b) Código cifrador

```
1.  /* aplicando la formula de cifrado: C_n = P_n + K mod longitud_alfabeto, donde:
2.      C_n, es el carácter del texto cifrado resultante
3.      P_n, es la letra de la posicion n del texto en claro
4.      K, es el valor del carácter de la llave
5.      longitud_alfabeto, es el tamaño del alfabeto
6.  */
7.  public static String Encrypt(String plain_text, String key, int longitud_alfabeto)
8.  {
9.      StringBuilder cipher_text = new StringBuilder();
10.
11.      int counter_key_value = 0;
12.      for (int i=0; i<plain_text.length(); i++)
13.      {
14.          if(counter_key_value >= key.length()){
15.              counter_key_value = 0;
16.          }
17.
18.          int P_n = (int)plain_text.charAt(i);
19.          int K = (int)key.charAt(counter_key_value);
20.          int C_n;
21.
22.          if(longitud_alfabeto == 26){ // alfabeto ingles
23.              if(P_n < 97 || P_n > 122){
24.                  C_n = P_n;
25.              }else{
26.                  C_n = ( ( P_n + K - 194 ) % longitud_alfabeto ) + 97;
27.              }
28.          }else{
29.              C_n = (P_n + K) % longitud_alfabeto;
30.          }
31.
32.          cipher_text.append((char)C_n);
33.
34.          counter_key_value += 1;
35.      }
36.
37.      return cipher_text.toString();
38.  }
```

## c) Código descifrador

```
1.  /* aplicando la formula de des cifrado: P_n = C_n + (-K) mod longitud_alfabeto, donde:
2.      P_n, es la letra del texto claro resultante
3.      C_n, es la letra de la posicion n del texto cifrado
4.      -K, es el inverso aditivo del carácter de la llave
5.      longitud_alfabeto, es el tamaño del alfabeto
6.
7.  public static String Decrypt(String cipher_text, String key, int longitud_alfabeto)
8.  {
9.      StringBuilder decipher_text = new StringBuilder();
10.
11.      int counter_key_value = 0;
12.      for (int i=0; i<cipher_text.length(); i++)
13.      {
14.          if(counter_key_value >= key.length()){
15.              counter_key_value = 0;
16.          }
```

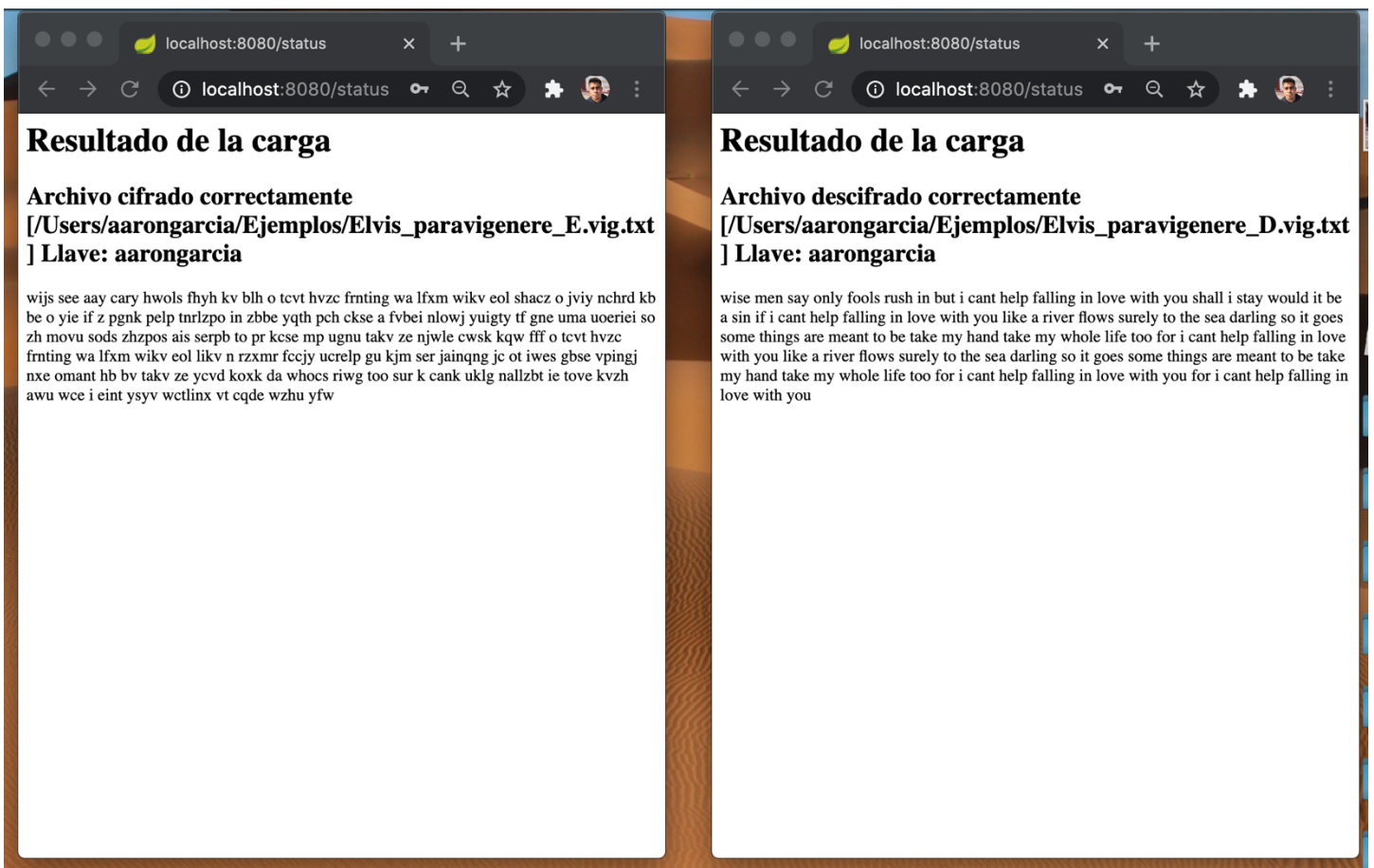


```

17.
18.     int C_n = (int)cipher_text.charAt(i);
19.     int K = AritmeticaModular.InversoAditivo((int)key.charAt(counter_key_value), longitud_alfabe
to);
20.     int P_n;
21.
22.     if(longitud_alfabeto == 26){
23.         if(C_n < 97 || C_n > 122){
24.             P_n = C_n;
25.         } else{
26.             P_n = ( ( C_n + K ) % longitud_alfabeto ) + 97;
27.         }
28.     }else{
29.         P_n = (C_n + K) % longitud_alfabeto;
30.     }
31.
32.     decipher_text.append((char)P_n);
33.
34.     counter_key_value += 1;
35. }
36.
37. return decipher_text.toString();
38. }

```

#### d) Prueba



## 6. Cifrador Affine

### a) Explicación

Este es algoritmo de sustitución mono-alfabética, esto debido a que todos los caracteres iguales de un texto se mapean a otro mismo con base en las funciones de cifrado y descifrado, este algoritmo es muy sencillo, la manera en que se puede cifrar es:

1. Verificar que  $\alpha$  y la longitud del alfabeto son coprimos, de serlo, el algoritmo puede continuar
2. A cada valor según la posición de cada carácter del texto en claro se multiplicará por el valor  $\alpha$ .
3. A cada valor obtenido en #2, se le sumara el valor de  $\beta$
4. A cada valor obtenido en #3, se le aplicara modulo longitud del alfabeto
5. Mapear cada número al valor correspondiente con el alfabeto

Para descifrar, lo que se debe hacer es:

6. Calcular el inverso multiplicativo de  $\alpha$
7. Calcular el inverso aditivo de  $\beta$
8. A cada valor según la posición de cada carácter del texto cifrado se le sumará el valor de  $\beta$  obtenido en #7
9. A cada valor de #8 se multiplicará por el valor obtenido de  $\alpha$  en #6
10. A cada valor obtenido en #9, se le aplicara modulo longitud del alfabeto
11. Mapear cada número al valor correspondiente con el alfabeto

### b) Código cifrador

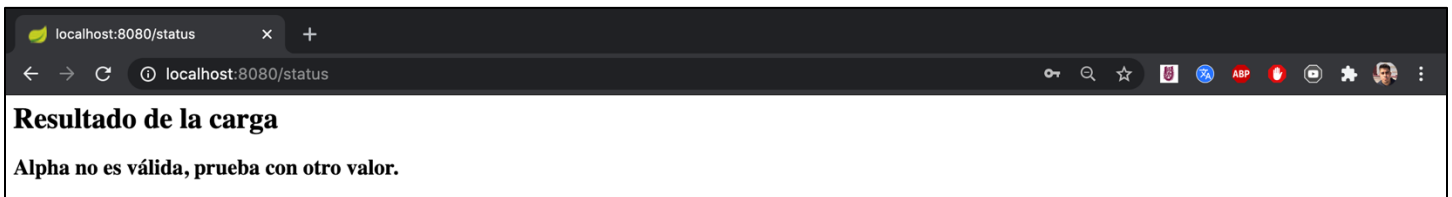
```
1.  /*
2.  * Funcion que cifra con base a una llave alpha valida
3.  *
4.  * aplicando la formula de cifrado:  $C_n = \alpha * P_n + \beta \bmod \text{longitud\_del\_alfabeto}$ , donde:
5.  *  $C_n$ , es el carácter del texto cifrado resultante
6.  *  $P_n$ , es la letra de la posicion n del texto en claro
7.  * alpha, es el valor multiplicativo de la llave
8.  * beta, es el valor aditivo de la llave
9.  */
10. public static String Encrypt(String plain_text, int alphabet_length, int alpha, int beta)
11. {
12.     StringBuilder cipher_text = new StringBuilder();
13.
14.     for (int i=0; i<plain_text.length(); i++)
15.     {
16.         int P_n = (int)plain_text.charAt(i);
17.         int C_n = ((alpha * P_n) + beta) % alphabet_length;
18.         cipher_text.append((char)C_n);
19.     }
20.
21.     return cipher_text.toString();
22. }
```

### c) Código descifrador

```
1.  /*
2.  * Aplicando la formula de des cifrado:  $P_n = \alpha^{-1} * [C_n + (-$ 
   *  $\beta)] \bmod \text{longitud\_del\_alfabeto}$ , donde:
3.  *  $C_n$ , es el carácter del texto en claro descifrado
4.  *  $P_n$ , es la letra de la posicion n del texto cifrado
5.  *  $\alpha$ , es el valor multiplicativo de la llave en inverso multiplicativo
6.  *  $\beta$ , es el valor aditivo de la llave en inverso aditivo
7.  * */
8.  public static String Decrypt(String cipher_text, int alphabet_length, int alpha, int beta)
9.  {
10.     StringBuilder decipher_text = new StringBuilder();
11.
12.     for (int i=0; i<cipher_text.length(); i++)
13.     {
14.         int C_n = (int)cipher_text.charAt(i);
15.         int alpha_inverso_multiplicativo = AritmeticaModular.InversoMultiplicativo(alpha, alphabet_length);
16.         int beta_inverso_aditivo = AritmeticaModular.InversoAditivo(beta, alphabet_length);
17.         int P_n = ( alpha_inverso_multiplicativo * ( C_n + beta_inverso_aditivo ) ) % alphabet_length;
18.
19.         decipher_text.append((char)P_n);
20.     }
21.     return decipher_text.toString();
22. }
```

### d) Pruebas

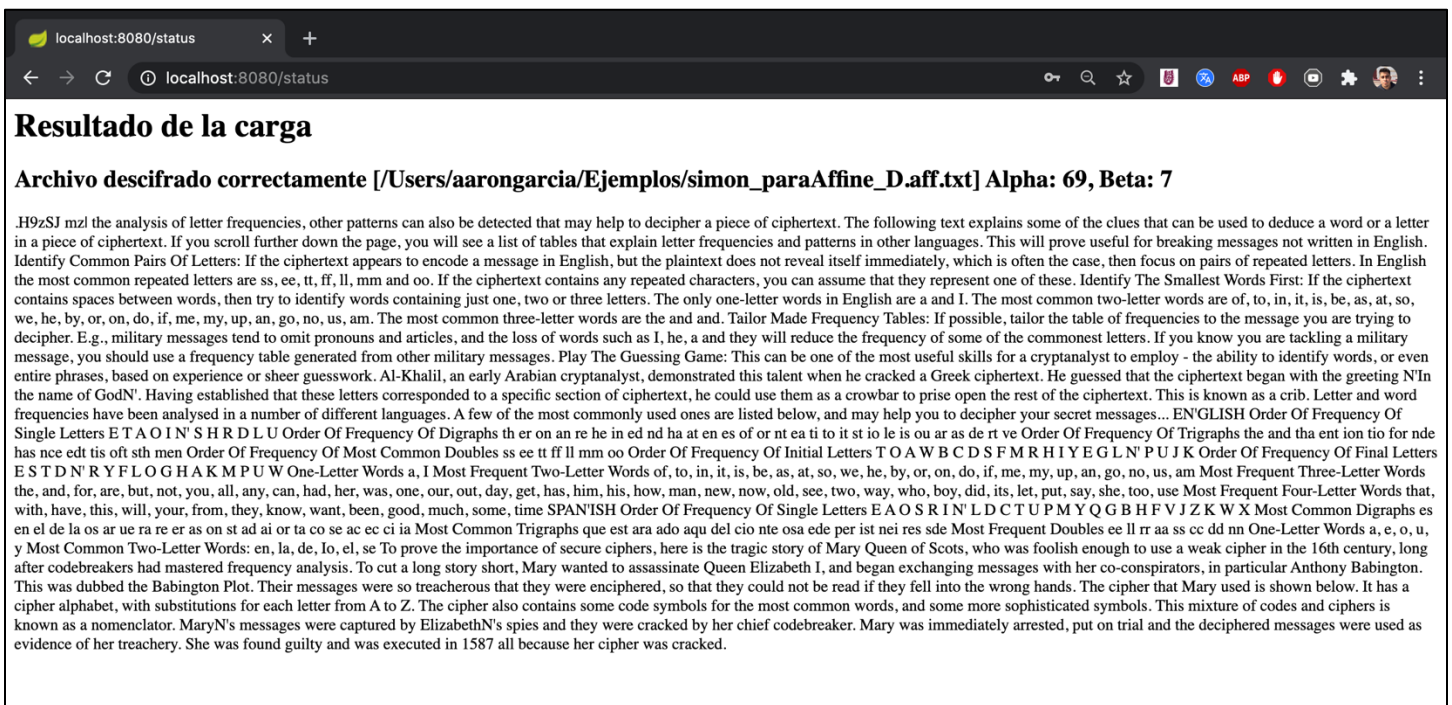
i.  $\alpha = 20, Z = 256, \beta = 3$



ii.  $\alpha = 69, Z = 128, \beta = 7$



Modificar los 10 primeros caracteres del texto cifrado y descifrar





i.  $\alpha = 11, Z = 256, \beta = 5$

localhost:8080/status

localhost:8080/status

Resultado de la carga

Archivo cifrado correctamente [/Users/aarongarcia/Ejemplos/simon\_paraAffine\_E.aff.txt] Alpha: 11, Beta: 3

İöc Zş§c.öcy{Zc.½.ş6ö ðeëc\$ZyYZëceZp Z½D ZðçëY{ZëCÖ.ÿyZë½äcD.½c.şðÊë9ZcOZYZDyZOcy{.ÿc².6c{ZşÖcyëcOZD Ö{Zëc.cÖ ZDZcëcD Ö{ZëyZ+ÿÿ q q {ZceÊşşÊ ½pcyZ+ÿcZ+Öş. ½öcðÊZcëcy{ZcDş Zöcy{.ÿcD.½c9Zc öZOcyëcOZO DZc.c ÊëOëEëc.eşZyYZëc ½c.cÖ ZDZcëcD Ö{ZëyZ+ÿÿcëcöÊ cöDëşşëcë äy{ZëcOÊ ½cy{ZcÖ.pZçc6Ê c şşcöZZc.cş öyçëcy.9şZöcy{.ÿcZ+Öş. ½cşZyYZëceZp Z½D Zöc.½OcÖ.ÿyZë½äc ½cy{Zcëş.½p pZöyç { öc şşcÖÊë Zc öZe şcëëc9Ez. ½pc²Zöc.pZöc½Ëyç é ÿyZ½c ½cö½pş öÿy q q&OZ½ÿ eëcäÊZ½ëcş. ëöçecGZYyZëö c&ëcy{ZcD Ö{ZëyZ+ÿc.ÖÖZ.ëöcyëcZ½DëOZc.cZöð.pZc ½cö½pş ö{ç9 ÿcy{ZcÖş. ½yZ+ÿcOÊZöc½ËÿcëZ Z.şc ½öZşec ºZO .ÿZş6çc { D{c ðeëYëZ½cy{ZcD.öZçy{Z½cëED öcÊ½cÖ. ëöçecëZÖZ.ÿZOcşZyYZëöyç&½cö½pş ö{cy{ZcÊöyëDÊZ½cëZÖZ.ÿZOcşZyYZëöc.ëZöðçcZZçyÿçeeççşşc²c.½OcÊËÿc&ëcy{ZcD Ö{ZëyZ+ÿcDÊ½ÿ. ½öc.½c6ëZÖZ.ÿZOcD{.ë.DyZëöçc6Ê cD.½c.öð Zcy{.ÿcy{ZöcëZÖëZ½ÿc½ËZcëcy{ZöZş q&OZ½ÿ eëc {Zc º.şşZöyçÄÊëOöc ëöy c&ëcy{ZcD Ö{ZëyZ+ÿcDÊ½ÿ. ½öcöÖ.DZöc9Zy ZZ½c ÊëOöçcy{Z½cyëëcyËc OZ½ÿ eëc ÊëOöcDÊ½ÿ. ½ ½pc öyçÊ½Zçyç ÊëEëcy{ëZZcşZyYZëöyç {ZcÊ½şöcÊ½ZöşZyYZëc ÊëOöc ½cö½pş ö{c.ëZc.c.½Oc&ÿç {ZcÊöyëDÊZ½ÿcÿ ÊöşZyYZëc ÊëOöc.ëZcëçyËçc ½çc ÿçc öçc9Zçc.öçc.ÿçcöËçc Zçc{Zçc9çcëËçcÊ½çcOËçc eççcZçc²çc Öçc.½ççpËçc½Ëçc öçc.ÿçc {ZcÊöyëDÊZ½ÿcÿ{ëZZöşZyYZëc ÊëOöc.ëZcy{Zc.½Oc.½Oy q . şÊëcR.OZc ëZp Z½D6c .9şZö çëccÖÊöð 9şZçyç. şÊëcy{Zcy.9şZcëccëZp Z½D ZöcyËcy{Zc²Zöð.pZc6Ê c.ëZcyç6 ½pcÿËcOZD Ö{Zëyçüÿÿç² ş .ÿëc²Zöð.pZöcyZ½OcÿËcÊ² .ÿcÖÊÊ½Ë ½öc.½Oc.ëÿ DşZöçc.½Ocÿ{ZcşÊëöçëëc ÊëOöcöD D{c.äc&çc{Zçc.c.½Ocÿ{Zöc şşcëZO DZcy{ZceëZp Z½D6cëëcÊëZcëëcy{ZcDÊZ½ÿZöşçZyYZëöyç&ëc6Ê c ½Ë c6Ê c.ëZcy.D ş ½pc.c² ş .ÿëc²Zöð.pZçc6Ê cö{Ê şOc öZc.cëZp Z½D6cy.9şZçpZ½Zë.ÿZOcëÊZ½ÿçy{Zëc² ş .ÿëc²Zöð.pZöÿ qş.6c {Zc Zöð ½pc .Z c { äcD.½c9ZcÊ½Zcëëcy{ZcDÊöyç öZe şcö şşöcëëc.cDëö½ÿ.½.ş6öyçËcZÖşÊëöçcy{Zc.9 ş .ÿçyËc OZ½ÿ eëc ÊëOöçcëëcZ Z½cZ½ÿ ëZcÖ{.ë.Döç9c.äZOcÊ½cZ+ÖZë Z½DZcëëcö{ZZçp Zöð Êë .ÿçşöc{. ş şçc.½cZ.ëşöcÊ.9 .½cDëö½ÿ.½.ş6öyçcOZ½ÿöÿ.ÿZOcy{ öcy.şZ½ÿç {Z½c{ZcDë.D Zöc.cëZZ cD Ö{ZëyZ+ÿÿç Zçp ZöðZOcy{.ÿcy{ZcD Ö{ZëyZ+ÿc9Zp.½c .ÿcy{Zc²çpZZy ½pc²&½cy{Zc½.Zcëëc Öc.ÿc. ½pcZöÿ.9ş ö{ZOcy{.ÿcy{ZöZçZyYZëöcDÊëëZöÊ½OZOcyËc.cöÖZD e DcöZDy Ê½cëëcD Ö{ZëyZ+ÿçc{ZcDÊ şOc öZcy{Zc.äc.cDÊë 9.ëcyÊëÖë öZeÖZ½ÿcy{ZcëZöyçëëcy{ZcD Ö{ZëyZ+ÿÿç { öc öc ½Ë ½c.äc.cDë 9ÿ qGZYyZëc.½Oc ÊëOöcëZp Z½D Zöç{.Zc9ZZ½c.½.ş6öZoc ½c.c½.ºZëcëëcO eëZëZ½ÿçş.½p pZöÿçlçez cëëcy{Zc²öyçDÊZ½ÿşçc öZOcÊ½Zöc.ëZçş öyZOc9Zşç çc.½Oc².6c{ZşÖc6Ê cyëcOZD Ö{Zëc6Ê cöZDëZçyZöð.pZöÿÿÿ q qü{G& qhëOZëchec ëZp Z½D6chec ½pşZcGZYyZëö qüç çlçc&cçc c c çicGc² qhëOZëchec ëZp Z½D6checi pé.Öö qÿ{cZëcÊ½c.½cëZc{Zc ½cZOc½Oc{.c.ÿcZ½cZöcëëcëë½ÿçZ.cÿ cyËc ½öçy ÊëşZc äcÊ.c.äc.äcOZOçÿcZ qhëOZëchec ëZp Z½D6chec ë pé.Öö qÿ{Zc½Ocÿ{.cZ½ÿç Ê½çÿ Êëëëë½OZc{.äc½DZcZOyçÿ öcëÿçöÿ{cZ½ qhëOZëchec ëZp Z½D6checRÊöyçcëZ½cçlÊ 9şZö qöðcZZçyçeeççşc²cëë qhëOZëchec ëZp Z½D6chec&½ y .şcGZYyZëö qçëcyËçc ½çc ÿçc öçc9Zçc.öçc.ÿçcöËçc Zçc{Zçc9çcëËçcÊ½ÿçcOËçc eççcZçc²çcçc Öçc.½ççpËçc½Ëçc öçc.² qRÊöyç ëZp Z½ÿç {ëZZöGZYyZëcÄÊëÖö qÿ{Zçc.½OçcëËçc.ëZçc9 ÿçc½ÿççc6Ê çc.şşçc.½ççcD.½çc{.Oçc{Zëçc .äçcÊ½ZçcÊ çcçÊ çcçÊ ½çcO.öççpZÿçc{.äçc{ ççc{ äçc{Ê çc².½çcçZ çc½Ë çcçÊşOçcöZZçyçËçc äçc .äçc {Ëçc9ÊöçcO Oçc ½öçcşZÿçcO ½çcö.ççcö{ZçyçËçc öZ qRÊöyç ëZp Z½ÿç Ê ööGZYyZëcÄÊöö qÿ{.ÿçc .ÿçc{.Zçyç{ öçc şşçc6Ê ççcëËçcÿçy{Zçcç ½Ë çc.½ÿçc9ZZ½ççpÊÊÖçc²D{çcöÊZçyç ZZ q q şl& qhëOZëchec ëZp Z½D6chec ½pşZcGZYyZëö qüççc c çcçc{Gçicäc çcçcRcÖc-c çÜc c çyçlçäc-cÄçc qRÊöyçcäZ½ÿçic pé.Öö qZöcZçcZşçZOZçc.cëäc.äc Zcë.cëZcZëc.äc½çyçc.Oc. cëëçc.cDëöZc.ZcDcZcD . qRÊöyçcäZ½ÿçc ë pé.Öö qR ZcZöyç.ë.c.Oëc.P cOZşcD Êç½ÿZëcö.cZOZcÖZëc öyç½Z cëZöcöZO qRÊöyç ëZp Z½ÿçËc 9şZö qZZçşëëëc.cöçcDDcOöc½½ qh½ZöGZYyZëcÄÊëÖö qçcZçcËçc çcç qRÊöyçcäZ½ÿçc ÊöGZYyZëcÄÊëÖö qZ½ççç.çcOZçc&ËçcZşçcöZ q q ÊcÖÊë Zcy{Zc ºÖËÿ.½DZcëëcDZD ÊZcD Ö{Zëöçc{ZëZc öcy{Zçyçp. DöÿËëëçcR.ëëc~ Z½ÿçëëc Dëÿöçc {ëc äcëÊëş ö{cZ½Ë p{cyËc öZc.c.Z. cD Ö{Zëc ½cy{Zc Uy{cDZ½ÿ öççcÊ½ÿçc.ëYZëcDëOZ9Ez. Zëëc{.Öc².öyZëZOcëëZp Z½D6c.½.ş6ö öy q çcD ÿc.cşÊ½ççöÿëëöçc{ÊëyçcR.ëëc ½yZOcyËc.äö.äö ½yZc~ ZZ½cöş A.9Zy{c&çc.½Oc9Zp.½cZD{.½p ½pcZöð.pZöc .ÿc{ZëcDëDÊ½ÿöð .ë.ÿËëöçc ½cÖ.ëÿ D ş.çcl½ÿ{Ê½cöÜ.9 ½pÿË½ÿç { öc äcO.99ZOcy{ZcÜ.9 ½pÿË½cşËÿçc {Z ëc²Zöð.pZöc ZëZcöËçcËcD{ZëÊ öcy{.ÿcy{Zöc ZëZcZ½D Ö{ZëZOçcöËçy{.ÿcy{ZöcDÊ şOc½Ëÿçc9ZcZc.Oc ecy{ZöcçZşçc ½yËcy{Zc ÊË½pc{.½Oöÿ q {ZcD Ö{Zëcy{.ÿcR.ëëc öZOc öcö{Ê ½c9ZşÊ .ÿc&ÿçc{.äc.cD Ö{Zëc.şÖ{.9Zçyç .ÿcö.9öÿ y .ÿ ½cöëëcëZcD{cşZyYZëcëËçcÿçyËçcËçc {ZcD Ö{Zëc.şöëcDÊ½ÿ. ½cöëëZcDëOZcöç9ÊëöçcËçc{ZcÊöyçDÊZ½ÿçc ÊëOöçc.½OcöÊZcÊëZcöÊÖ{ öy D.ÿZOcöç9Êëöçc { äc² +ÿ êZcëëcDëOZöc.½OcD Ö{Zëöc äc ½Ë ½c.äc.c½ËÿZ½Dş.ÿËÿç q R.ë6²cö²Zöð.pZöc ZëZcD.Öy ëZOc9çcäş A.9Zy{²cöcö Zöc.½Ocÿ{Zöc ZëZcDë.D ZOç9çc{ZëcD{ ZëcDëOZ9Ez. ZëçcR.ëëc äc ºZO .ÿZş6çc.ëëZöyçZöçÖ ÿcÊ½ÿçë .şc.½Ocÿ{ZcOZD Ö{ZëZOc²Zöð.pZöc ZëZc öZOc.äcZ OZ½ÿZcëëc{ZëçyçZD{Zëöÿç {Zc äcëÊ½Oc çp şÿçc.½Oc äcZ+ZD ÿZOc ½c Jk².c.şçc9ZD.öZc{ZëcD Ö{Zëc.äcDë.D ZOÿ

Modificar los 10 primeros caracteres del texto cifrado y descifrar

localhost:8080/status

localhost:8080/status

Resultado de la carga

Archivo descifrado correctamente [/Users/aarongarcia/Ejemplos/simon\_paraAffine\_D.aff.txt] Alpha: 11, Beta: 3

~ÄÄö %ÖÖP the analysis of letter frequencies, other patterns can also be detected that may help to decipher a piece of ciphertext. The following text explains some of the clues that can be used to deduce a word or a letter in a piece of ciphertext. If you scroll further down the page, you will see a list of tables that explain letter frequencies and patterns in other languages. This will prove useful for breaking messages not written in English. Identify Common Pairs Of Letters: If the ciphertext appears to encode a message in English, but the plaintext does not reveal itself immediately, which is often the case, then focus on pairs of repeated letters. In English the most common repeated letters are ss, ee, tt, ff, ll, mm and oo. If the ciphertext contains any repeated characters, you can assume that they represent one of these. Identify The Smallest Words First: If the ciphertext contains spaces between words, then try to identify words containing just one, two or three letters. The only one-letter words in English are a and I. The most common two-letter words are of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am. The most common three-letter words are the and and. Tailor Made Frequency Tables: If possible, tailor the table of frequencies to the message you are trying to decipher. E.g., military messages tend to omit pronouns and articles, and the loss of words such as I, he, a and they will reduce the frequency of some of the commonest letters. If you know you are tackling a military message, you should use a frequency table generated from other military messages. Play The Guessing Game: This can be one of the most useful skills for a cryptanalyst to employ - the ability to identify words, or even entire phrases, based on experience or sheer guesswork. Al-Khalil, an early Arabian cryptanalyst, demonstrated this talent when he cracked a Greek ciphertext. He guessed that the ciphertext began with the greeting 'In the name of God'. Having established that these letters corresponded to a specific section of ciphertext, he could use them as a crowbar to prise open the rest of the ciphertext. This is known as a crib. Letter and word frequencies have been analysed in a number of different languages. A few of the most commonly used ones are listed below, and may help you to decipher your secret messages... ENGLISH Order Of Frequency Of Single Letters E T A O I N S H R D L U Order Of Frequency Of Digraphs th er on an re he in ed na t en of nt ei te it st io le is ou ar as de rt ve Order Of Frequency Of Trigraphs the and tha ent ion tio for nde has nce edt is oft sth men Order Of Frequency Of Most Common Doubles ss ee tt ff ll mm oo Order Of Frequency Of Initial Letters T O A W B C D S F M R H I Y E G L N P U J K Order Of Frequency Of Final Letters E S T D N R Y F L O G H A K M P U V One-Letter Words a, I Most Frequent Two-Letter Words of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am Most Frequent Three-Letter Words the, and, for, are, but, not, you, all, any, can, had, her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way, who, boy, did, its, let, put, say, she, too, use Most Frequent Four-Letter Words that, with, have, this, will, your, from, they, know, want, been, good, much, some, time SPANISH Order Of Frequency Of Single Letters E A O S R I N L D C T U P M Y Q G B H F V J Z K W X Most Common Digraphs es en el de la os ar ue ra er as on st ad ai or ta co se ac ec ci ia Most Common Trigraphs que est ara ado aqu del cio nte osa ede per ist nei res sde Most Frequent Doubles ee ll rr aa ss cc dd nn One-Letter Words a, e, o, u, y Most Common Two-Letter Words: en, la, de, lo, el, se, to, I, e, a Most Frequent Three-Letter Words the, and, for, are, but, not, you, all, any, can, had, her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, long after codebreakers had mastered frequency analysis. To cut a long story short, Mary wanted to assassinate Queen Elizabeth I, and began exchanging messages with her co-conspirators, in particular Anthony Babington. This was dubbed the Babington Plot. Their messages were so treacherous that they were enciphered, so that they could not be read if they fell into the wrong hands. The cipher that Mary used is shown below. It has a cipher alphabet, with substitutions for each letter from A to Z. The cipher also contains some code symbols for the most common words, and some more sophisticated symbols. This mixture of codes and ciphers is known as a nomenclator. Mary's messages were captured by Elizabeth's spies and they were cracked by her chief codebreaker. Mary was immediately arrested, put on trial and the deciphered messages were used as evidence of her treachery. She was found guilty and was executed in 1587 all because her cipher was cracked.

## 7. Conclusiones

En lo personal si tuve algunas dificultades al implementar el algoritmo de Vigenere, ya que la especificación de la practica acoto a utilizar el alfabeto ingles, mis programas funcionan a partir del código ASCII, mi cifrado/descifrado Vigenere funciona en todo el alfabeto ASCII, por lo que añadí una condición para verificar cuando la longitud del alfabeto sea igual a 26, tratarlo de manera diferente, y también dado que permite elegir el tamaño del alfabeto, hay que tener cuidado que la clave este dentro de los primeros n caracteres que especifico el usuario, de lo contrario dará errores, para el cifrado Affine, dado que el alfabeto es mas general, todo fue mas sencillo, es decir a la expresión original para descifrar no le modifique nada

Para mi gusto ambos cifradores tienen lo suyo, creo que es una excelente aproximación a lo que va de esta unidad de aprendizaje, también tuve que investigar algunas cosas adicionales, dado que la parte de la interfaz grafica la hago en web junto con Spring, meter un archivo a un formulario junto con otros datos y casillas de verificación en conjunto, lo desconocía.

## 8. Tabla de cotejo

1. Documento	tú	
La portada tiene: nombre completo, materia, nombre de profesor, fecha, logotipos, título de práctica y un resumen. (ver ejemplo anexo)	1	1
En media cuartilla con tus palabras explicar el AE y AEE, qué es y para qué sirve cada uno de ellos.	1	1
Código correspondiente a las dos funciones AE y AEE y captura de pantalla de las ejecuciones de las pruebas solicitadas.	2	2
Media cuartilla con tus palabras sobre el algoritmo Affine	1	
Código correspondiente a las dos funciones (cifrado y descifrado) así como el cálculo de la llave de descifrado y captura de pantalla de las ejecuciones de las pruebas solicitadas.	2	2
Media cuartilla con tus palabras sobre el algoritmo Vigenère	1	1
Código correspondiente a las dos funciones (cifrado y descifrado) así como el cálculo de la llave de descifrado y captura de pantalla de las ejecuciones de las pruebas solicitadas.	2	2
Conclusiones en donde se exprese principalmente las dificultades de la implementación de la práctica (si es que las hubo) así como una reflexión sobre la diferencia entre cifradores de sustitución mono alfabética y poli alfabético.	1	1
El código tiene formato (sugiero utilizar <a href="http://www.planetb.ca/syntax-highlight-word">http://www.planetb.ca/syntax-highlight-word</a> )	1	1
Todas las imágenes en el documento tienen título y se referencian en alguna parte del mismo. (Ej. “en la imagen 1 se muestra ...”)	1	0
Programa		
El programa cuenta con interfaz gráfica que le permita al usuario elegir la opción deseada: algoritmo y cifrado o descifrado y que reciba los parámetros (en caso de ser necesario)	2	2
La interfaz permite seleccionar el archivo que se va a cifrar/descifrar	2	2
Su programa genera el archivo cifrado con el mismo nombre del archivo de entrada más la extensión .vig o .aff	2	2
Las funciones AE y AEE mandan mensaje “prueba con otro valor” en caso de que alpha no sea coprimo con n	1	1
TOTAL	20	19

García González Aarón Antonio