Ejercicios Criptografia I

Ulkei Szabolcs

Erasmus

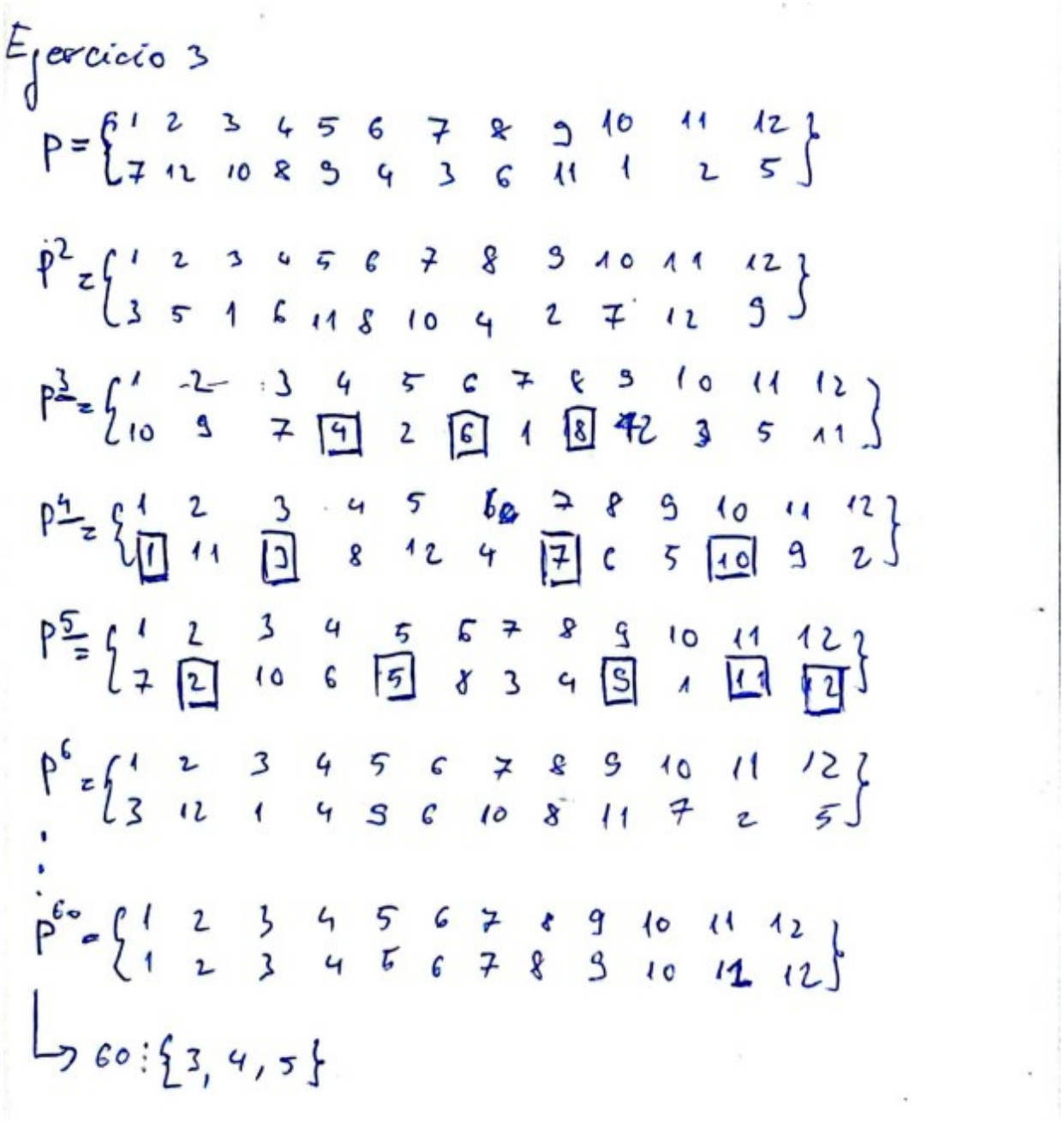
D210871

Ejercicio 3

Dado el siguiente sistema de transposición por permutación:

- $1 \rightarrow 7$
- $2 \rightarrow 12$
- $3 \rightarrow 10$
- $4 \rightarrow 8$
- $5 \rightarrow 9$
- $6 \rightarrow 4$
- $7 \rightarrow 3$
- $8 \rightarrow 6$
- $9 \rightarrow 11$
- $10 \rightarrow 1$
- $11 \rightarrow 2$
- $12 \rightarrow 5$

¿Podría ocurrir que al repetirse varias veces se llegue al mensaje origen? Si fuera así, ¿tras cuántas aplicaciones del cifrado?



Como las ubitaciones se repetiren a la pas 60 la codigo se repetire.

```
60:
1 2 3 4 5 6 7 8 9 10 11 12
1 2 3 4 5 6 7 8 9 10 11 12
```

Codigo generador en la fin de documento.

Ejercicio 5

a) El siguiente es un cifrado afín: $c_i(m_i) = (3m_i + 5)\ mod\ 27$. Calcular la función afín de descifrado.

b) Por otro lado: ¿El cifrado César con clave es un tipo de cifrado afín? Razonarlo.

c) Y de igual forma, ¿el cifrado Atbash es un tipo de cifrado afín? Razonarlo.

Ejercicio 5

a.) $c_i(m_i) = (3m_i + 5) \bmod 27$

Para descifrar habrá que realizar el proceso inverso que es $D(x) = (a^{-1}(x - b)) \bmod n$

a^{-1} es inverso multiplicativo en aritmética modular

~~$\Rightarrow (a^{-1} \cdot a) \bmod n = 1$~~

~~$\Rightarrow (a^{-1} \cdot 5) \bmod$~~

$m_i = (a^{-1} \cdot (c_i - b)) \bmod n = (a^{-1} \cdot (c_i - 5)) \bmod 27$

$\Rightarrow m_i(c_i) = (a^{-1} \cdot c_i) \bmod 27 + ((27 - 5) \cdot a^{-1}) \bmod 27 \Rightarrow$

$\Rightarrow (a^{-1} \cdot 22) \bmod 27 = 1 \iff a^{-1} \cdot 22 = 27k + 1,\ k \in \mathbb{N}$

~~Sea $k = 5 \Rightarrow 22 \cdot a^{-1} = 28$~~

$a^{-1} = \frac{27k + 1}{22}$. Si $a^{-1} \in \mathbb{N} \Rightarrow \frac{27k + 1}{22} \in \mathbb{N}$

Sea $k = 13 \Rightarrow a^{-1} = \frac{27 \cdot 13 + 1}{22} = \frac{351 + 1}{22} = 16$

$\Rightarrow m_i(c_i) = (16 \cdot (c_i - 5)) \bmod 27$

b.)

El cifrado Cesar con clave está un tipo de cifrado afín porque podemos modelar que:

$c_i(m_i) = (a \cdot m_i + b) \bmod n$, dónde $a = 1$, $b =$ clave$\in N$, y n está numero de simbolos

c.)

El cifrado Atbash es un caso especial del cifrado afín porque podemos modelar que:

$a = m - 1$

$b = m - 1$

$n =$ numero de simbolos

$\Rightarrow c_i(m_i) = ((m-1)m_i - (m-1)) \bmod m$

$= -(m_i + 1) \bmod m$

Ejercicio 6

Descifrar el siguiente texto, que ha sido cifrado con el sistema afín.

"EDREIKZKIWOWEZNKTEJEDMBGYEBIKOKARMZOMECNRKTEDMVWZNYR
MKIIWTEZNMDPYEREDMNWJMOEZNENMRTWKEOVEBMZTKMCERIKZCWTER
MTKMVRWZIWVWKCTEDCWADKFWF"

Resuelta:

MXEM<AYA<DKDMYVAJM1MXFS7IMS<AKACEFYKFMHVEAJMXFODYVIEFA<<DJMYVFX[IMEMXFVD1FKMYVMVFEJDAMKOMSFYJAFHME<AYHDJMEFJAFOEDY<DODAHJMXHDCXA'D'@

```
PS D:\UPT\SEM2\Teoria_Codigos_Criptografia\Teoria_Codigos_Ciclo> ./afin.exe
MXEM<AYA<DKDMYVAJM1MXFS7IMS<AKACEFYKFMHVEAJMXFODYVIEFA<<DJMYVFX[IMEMXFVD1FKMYVMVFEJDAMKOMSFYJAFHME<AYHDJMEFJAFOEDY<DODAHJMXHDCXA'D'@
```

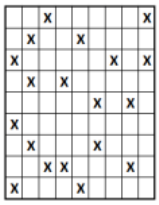
Codigo en el fin de documento

Ejercicio 7

Nos dan dos opciones de cifrado para utilizar:

Por un lado, un cifrado afín en el que usamos el alfabeto castellano A=0, B=1, C=2,...,M=12, N=13, O=14,..., X=23, Y=24, Z=25. El cifrado es según la regla $c_i=6m_i+5 \bmod 26$, donde m_i es el fragmento i-ésimo del mensaje, la letra i-ésima, y su relativo cifrado es c_i.

El otro tipo de cifrado es del tipo Cardano, y nos dan la rejilla siguiente:



¿Cuál usarías y por qué?

3

Es dependiente de metoda de utilizacion del codigo. Por mejor seguramente usariase Cardano pero esto tiene la desavantaje que las descriptores son unicos, entonces el informacion o no puede llegar por muchos personas o no tiene la mismo seguridad que antes.

Por grupos mas grandes que tienen la necesitad de enviar informacion cifrado por mas gentes el cifrado afin es mejor por que es mas facil implementada y tambien mas facil de cambiar para permanecer seguridad.

Ejercicio 8

Partiendo del alfabeto [A,B,C,...,M,N,O,P,...,U,V,W,X,Y,Z], desde [0...25], utilizar el cifrado de Hill para descifrar el mensaje cifrado siguiente:

"OCXRGQUKLPKMGPEOHXPYSAXWMLPOPJCSEKQGWKFTIDLGRMLTHBFJ
FPUKRDRITOHRXMFRGNLBHYNXBENRHSVNRYNHDAZBGGWGLVSYMECYOF
JWXDRPLMTXTEXRYOKOKCTQUJSDTNPGYCIAVLPRHGPJIHKTZTDPTEKQ
JAMMRQZFXMSEURGHVUKQWMYMMDDVIJSNKUOCKNUIDTTEESPSLHPMYL
SQATKBXBNEMMFIXAAQBDXGAOYGIHWNAUGSNNZTAUGSBMQSHAVLMJDC
BQKVMRJBPHRRJVVYYIFYHOSWPADHZQSUGVNVOYDWRGOCAHTFCIMGJA
YYAOYGUFINATPHBZWUZQGINPPNPDZONAAKGXXXNMWGAZXLVYWIGXXX
NMWGEYZOBIAINWYAVPWH"

Tener en cuenta como clave de cifrado la matriz $K = \begin{pmatrix} 8 & 5 & 11 & 3 \\ 7 & 11 & 15 & 12 \\ 15 & 18 & 8 & 14 \\ 19 & 22 & 1 & 7 \end{pmatrix}$.

Resuelta:

VYFDMDWQHAEVXRVCKWAQDHHSHXDVLQMBHLWUXNWMRJIIZHASOLOSVZBQJJECEEIWOBIZUEAATFQ
GTXZBKTTFULJHOFLZVAWGQWYSNIEZDEKNMVEFRFUHBUKXKNQRLLBFATBRBJVWTEWWZAWDUXDVAZF
JRZCXVGEGSTNXEJIVGJUAZDDRIZXNUVJNIXXTUVJNGFDREEVI

```
PS D:\UPT\SEM2\Teoria_Codigos_Criptografia\Teoria_Codigos_Ciclo> ./hill.exe
VYFDMDWQHAEVXRVCKWAQDHHSHXDVLQMBHLWUXNWMRJIIZHASOLOSVZBQJJECEEIWOBIZUEAATFQGTXZBKTTFULJHOFLZVAWGQWYS
NIEZDEKNMVEFRFUHBUKXKNQRLLBFATBRBJVWTEWWZAWDUXDVAZFJRZCXVGEGSTNXEJIVGJUAZDDRIZXNUVJNIXXTUVJNGFDREEVI
```

Codigo Ejercicio 3

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

int main(){
    int p[] = {7,12,10,8,9,4,3,6,11,1,2,5};

    int k = 60;

    int pprev[12];

    for(int j = 0; j < 12; j ++)
        pprev[j] = p[j];


    for(int i = 0; i < 60; i++){
        int ptemp[12];
        for(int j = 0; j < 12; j++)
            ptemp[j] = pprev[p[j] - 1];

        printf("\n\n%d:\n", i+2);
        for(int j = 0; j < 12; j++){
            printf("%d ", j+1);
        }
        printf("\n");
        for(int j = 0; j < 12; j++){
            printf("%d ", ptemp[j]);
        }



        for(int j = 0; j < 12; j ++)
            pprev[j] = ptemp[j];

    }


}
```

Codigo ejercicio 6

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

int mi(int ci){
    int citemp = (ci - 5 >= 0) ? (ci - 5) : (27 + ci - 5);
    return (16 * (citemp - 5)) % 27;
}

int main(){
    FILE* f;
    char ci;

    f = fopen
        int __cdecl fgetc(FILE *)
        Character Input and Output Functions
    do{
        ci = fgetc(f);
        printf("%c", 'A' + mi(ci - 'A'));
    }
    while(ci != EOF);


}
```

Codigo ejercicio 8

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

int** mul_matrix(int** mat_A, int n_A, int m_A,  int** mat_B, int n_B, int m_B){
    int n_res = n_A;
    int m_res = m_B;

    int** mat_res;
    mat_res = (int**)malloc(n_res * sizeof(int*));
    for(int i = 0; i < n_res; i++){
        mat_res[i] = (int*)calloc(m_res, sizeof(int));
    }

    for(int i_A = 0; i_A < n_A; i_A++){
        for(int j_B = 0; j_B < m_B; j_B++){
            for(int j_A = 0; j_A < m_A; j_A++){
                //printf("\n\n%d   *   %d\n\n",mat_A[i_A][j_A], mat_B[j_B][j_A]);
                mat_res[i_A][j_B] += mat_A[i_A][j_A] * mat_B[j_A][j_B] ;
            }
        }
    }
    return mat_res;
}

```

```c
int main(){
    int a[4][4] = {
        {6, 20, 25, 6},
        {5, 4, 17, 9},
        {23, 3, 18, 1},
        {13, 7, 21, 0}
        };

    int** a_mat;
    int** b_mat;
    a_mat = (int**)malloc(4 * sizeof(int*));
    b_mat = (int**)malloc(4 * sizeof(int*));

    for(int i = 0; i < 4; i++){
        a_mat[i] = (int*)malloc(4 * sizeof(int));
        b_mat[i] = (int*)malloc(sizeof(int));
        for(int j = 0; j < 4; j++)
            a_mat[i][j] = a[i][j];
    }

    FILE* f;
    char ch;

    f = fopen("Hill_Cypher.txt", "r");
    do{
        for(int i = 0; i < 4; i++){
            ch = fgetc(f);
            b_mat[i][0] = ch - 'A';
        }
        for(int i = 0; i < 4; i++){
            ch = fgetc(f);
        }

        int** res;
        res = mul_matrix(a_mat, 4, 4, b_mat, 4, 1);

        for(int i = 0; i < 4; i++)
            printf("%c", ((res[i][0]) % 26) + 'A');

    }
    while (ch != EOF);
}
```