

Implementación de Renyi Maps acoplados (modificación) y aplicación de pruebas NIST.

Marcos Daniel Calderón Calderón

Abstract

En este reporte, se explica a detalle la implementación de varios mapas caóticos, además, se hace un análisis del valor acoplado H que se genera en cada algoritmo.

1. Introducción.

A continuación, se especifican algunos criterios para la implementación de mapas caóticos acoplados.

1.1. Criterio 1.

El primer criterio utilizado es el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, \dots, X_{N,j-1}) \quad (1)$$

donde:

$$H(X_{i,j-1}, \dots, X_{N,j-1}) = \sum_{i=1}^N (X_{i,j-1} \text{ mód } 256). \quad (2)$$

además, ϵ es un valor aleatorio del conjunto: $\{-1, 0, 1\}$

1.2. Criterio 2.

El segundo criterio utilizado es el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, \dots, X_{N,j-1}) \quad (3)$$

donde:

$$H(X_{i,j-1}, \dots, X_{N,j-1}) = \bigoplus_{i=1}^N (X_{i,j-1}). \quad (4)$$

donde \bigoplus representa la operación *XOR*, además ϵ es un valor aleatorio del conjunto: $\{-1, 0, 1\}$

Email address: marcos.calderon@cimat.mx (Marcos Daniel Calderón Calderón)

1.3. Criterio 3.

El tercer criterio utilizado es el siguiente:

$$X_{i,j} = \hat{\gamma} \bigoplus f(X_{i,j-1}) + \gamma \bigoplus H(X_{i,j-1}, \dots, X_{N,j-1}) \quad (5)$$

donde:

$$H(X_{i,j-1}, \dots, X_{N,j-1}) = \bigoplus_{i=1}^N (X_{i,j-1}). \quad (6)$$

donde \bigoplus representa la operación *XOR*, además para el cálculo del valor de γ , necesitamos un ϵ que será un valor aleatorio donde $1 \geq \epsilon \geq 32$, el rango es establecido de acuerdo al tipo de dato utilizado cuando se traduce el sistema a un lenguaje de programación elegido. Ahora, podemos calcular γ de la siguiente manera:

$$\gamma = 2^\epsilon - 1 \quad (7)$$

$$\hat{\gamma} = M - \gamma \quad (8)$$

donde $M = 2^{32} - 1$.

2. Detalles de implementación.

A continuación, se especifican algunos detalles importantes a tomar en cuenta a la hora de la implementación de los mapas acoplados:

- En el criterio 1 y 2, es importante obtener valores de H que no sean muy grandes, esto se busca con la finalidad de no hacer perturbaciones muy grandes. Es necesario analizar los valores de H que son generados.
- Se buscaron 80000 valores en la ejecución del programa; además, cada valor está compuesto de 32 bits cuando se guarda la información en un archivo binario. Por las características mencionadas anteriormente, se pueden leer **2,560,000 bits** para la aplicación de las pruebas NIST.
- Para equipos de cómputo de 32 bits, se utilizó el tipo de dato **unsigned long**, que está conformado de 4 bytes. Si el equipo de cómputo utilizado es de 64 bits, se recomienda utilizar el tipo de dato **unsigned int**, todavía no hay un estándar definido para el tamaño de los tipos de datos en los equipos de 64 bits.

3. Resultados.

3.1. Aplicación de las pruebas NIST a los mapas acoplados del Criterio 1.

Recordemos que el criterio 1 era el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, \dots, X_{N,j-1}) \quad (9)$$

donde:

$$H(X_{i,j-1}, \dots, X_{N,j-1}) = \sum_{i=1}^N (X_{i,j-1} \text{ mód } 256). \quad (10)$$

además, ϵ es un valor aleatorio del conjunto: $\{-1, 1\}$

En este ejemplo, se hizo una modificación en la forma en cómo se obtuvo H. Se ejecutó el siguiente código:

- `./assess 2560000`
- User Prescribed Input File: **binarioSUMAmod.dat**
- Enter 0 if you DO NOT want to apply all of the statistical tests to each sequence and 1 if you DO. Enter chice: **1**
- How many bitstreams? **1**
- Input File Format: [0] ASCII - A sequence of ASCII 0's and 1's [1] Binary - Each byte in data file contains 8 bits of data
- Select input mode: **1**

Se obtuvieron los siguientes resultados:

Cuadro 1: Resultados de las pruebas de aleatoriedad NIST a los datos binarioSUMAmod.dat .

| PRUEBA APLICADA | P-VALOR | EXITO? |
|---------------------------|--|--------|
| APROXIMATE ENTROPY | 0.561487 | ✓ |
| BLOCK FREQUENCY | 0.123096 | ✓ |
| CUMULATIVE SUMS | FORWARD TEST: 0.393884, REVERSE TEST: 0.215339 | ✓ |
| FFT | 0.235138 | ✓ |
| FREQUENCY | 0.216366 | ✓ |
| LINEAR COMPLEXITY | 0.639128 | ✓ |
| LONGEST RUN | 0.680758 | ✓ |
| NON OVERLAPPING TEMPLATE | P-VALORES ACEPTADOS: 148 DE 148 | ✓ |
| OVERLAPPING TEMPLATE | 0.679278 | ✓ |
| RANDOM EXCURSIONS | P-VALORES ACEPTADOS: 8 DE 8 | ✓ |
| RANDOM EXCURSIONS VARIANT | P-VALORES ACEPTADOS: 18 DE 18 | ✓ |
| RANK | 0.286919 | ✓ |
| RUNS | 0.091564 | ✓ |
| SERIAL | P-VALORES ACEPTADOS: 2 DE 2 | ✓ |
| UNIVERSAL | 0.990027 | ✓ |

3.2. Aplicación de las pruebas NIST a los mapas acoplados del Criterio 2.

El criterio 2 era el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, \dots, X_{N,j-1}) \quad (11)$$

donde:

$$H(X_{i,j-1}, \dots, X_{N,j-1}) = \bigoplus_{i=1}^N (X_{i,j-1}). \quad (12)$$

donde \bigoplus representa la operación *XOR*, además ϵ es un valor aleatorio del conjunto: $\{-1, 1\}$

En este ejemplo, se hizo una modificación en la forma en cómo se obtuvo H. Se ejecutó el siguiente código: Para el criterio 2, se ejecutó el siguiente código:

- ./assess 2560000
 - User Prescribed Input File: **binarioXORmod.dat**
 - Enter 0 if you DO NOT want to apply all of the statistical tests to each sequence and 1 if you DO. Enter chice: **1**
 - How many bitstreams? **1**
 - Input File Format: [0] ASCII - A sequence of ASCII 0's and 1's [1] Binary - Each byte in data file contains 8 bits of data
- Select input mode: **1**

Se obtuvieron los siguientes resultados:

Cuadro 2: Resultados de las pruebas de aleatoriedad NIST a los datos binarioXORmod.dat .

| PRUEBA APLICADA | P-VALOR | EXITO? |
|---------------------------|---|--------|
| APROXIMATE ENTROPY | 0.955023 | ✓ |
| BLOCK FREQUENCY | 0.932697 | ✓ |
| CUMULATIVE SUMS | FORWARD TEST:0.532788, REVERSE TEST: 0.466893 | ✓ |
| FFT | 0.349856 | ✓ |
| FREQUENCY | 0.935243 | ✓ |
| LINEAR COMPLEXITY | 0.014630 | ✓ |
| LONGEST RUN | 0.014097 | ✓ |
| NON OVERLAPPING TEMPLATE | P-VALORES ACEPTADOS: 145 DE 148 | ✓ |
| OVERLAPPING TEMPLATE | 0.129367 | ✓ |
| RANDOM EXCURSIONS | P-VALORES ACEPTADOS: 8 DE 8 | ✓ |
| RANDOM EXCURSIONS VARIANT | P-VALORES ACEPTADOS: 14 DE 18 | ✓ |
| RANK | 0.001981 | No. |
| RUNS | 0.865007 | ✓ |
| SERIAL | P-VALORES ACEPTADOS: 2 DE 2 | ✓ |
| UNIVERSAL | 0.425674 | ✓ |

3.3. Aplicación de las pruebas NIST a los mapas acoplados del Criterio 3.

EL criterio 3 era el siguiente:

$$X_{i,j} = \hat{\gamma} \bigoplus f(X_{i,j-1}) + \gamma \bigoplus H(X_{i,j-1}, \dots, X_{N,j-1}) \quad (13)$$

donde:

$$H(X_{i,j-1}, \dots, X_{N,j-1}) = \bigoplus_{i=1}^N (X_{i,j-1}). \quad (14)$$

Para el criterio 3, se ejecutó el siguiente código:

- ./assess 2560000
- User Prescribed Input File: **binarioXORcomp.dat**
- Enter 0 if you DO NOT want to apply all of the statistical tests to each sequence and 1 if you DO. Enter chice: **1**
- How many bitstreams? **1**
- Input File Format: [0] ASCII - A sequence of ASCII 0's and 1's [1] Binary - Each byte in data file contains 8 bits of data
- Select input mode: **1**

Se obtuvieron los siguientes resultados:

Cuadro 3: Resultados de las pruebas de aleatoriedad NIST a los datos binarioXORcomp.dat .

| PRUEBA APLICADA | P-VALOR | EXITO? |
|---------------------------|--|--------|
| APROXIMATE ENTROPY | 0.979174 | ✓ |
| BLOCK FREQUENCY | 0.283892 | ✓ |
| CUMULATIVE SUMS | FORWARD TEST: 0.320736, REVERSE TEST: 0.476676 | ✓ |
| FFT | 0.990848 | ✓ |
| FREQUENCY | 0.286876 | ✓ |
| LINEAR COMPLEXITY | 0.142002 | ✓ |
| LONGEST RUN | 0.106050 | ✓ |
| NON OVERLAPPING TEMPLATE | P-VALORES ACEPTADOS: 145 DE 148 | ✓ |
| OVERLAPPING TEMPLATE | 0.879647 | ✓ |
| RANDOM EXCURSIONS | P-VALORES ACEPTADOS: 8 DE 8 | ✓ |
| RANDOM EXCURSIONS VARIANT | P-VALORES ACEPTADOS: 18 DE 18 | ✓ |
| RANK | 0.398102 | ✓ |
| RUNS | 0.964539 | ✓ |
| SERIAL | P-VALORES ACEPTADOS: 2 DE 2 | ✓ |
| UNIVERSAL | 0.603939 | ✓ |

4. Resultados para H.

En este ejemplo, se tomaron 20,000 valores generados de H en cada uno de los criterios, como cada uno de estos valores está representado en 32 bits, el archivo binario estuvo compuesto de 640,000 bits.

4.1. Analisis de los valores H generados para el criterio 1.

Cuadro 4: Resultados de las pruebas de aleatoriedad NIST a los datos HdeSuma.dat .

| PRUEBA APLICADA | P-VALOR | EXITO? |
|---------------------------|---|--------|
| APROXIMATE ENTROPY | 0.935134 | ✓ |
| BLOCK FREQUENCY | 0.122908 | ✓ |
| CUMULATIVE SUMS | FORWARD TEST:0.738816, REVERSE TEST: 0.596414 | ✓ |
| FFT | 0.818546 | ✓ |
| FREQUENCY | 0.439818 | ✓ |
| LINEAR COMPLEXITY | 0.813844 | ✓ |
| LONGEST RUN | 0.108021 | ✓ |
| NON OVERLAPPING TEMPLATE | P-VALORES ACEPTADOS: 147 DE 148 | ✓ |
| OVERLAPPING TEMPLATE | 0.058303 | ✓ |
| RANDOM EXCURSIONS | P-VALORES ACEPTADOS: 8 DE 8 | ✓ |
| RANDOM EXCURSIONS VARIANT | P-VALORES ACEPTADOS: 18 DE 18 | ✓ |
| RANK | 0.847382 | ✓ |
| RUNS | 0.697088 | ✓ |
| SERIAL | P-VALORES ACEPTADOS: 2 DE 2 | ✓ |
| UNIVERSAL | 0.673427 | ✓ |

4.2. Analisis de los valores H generados para el criterio 2.

Cuadro 5: Resultados de las pruebas de aleatoriedad NIST a los datos HdeXOR.dat .

| PRUEBA APLICADA | P-VALOR | EXITO? |
|---------------------------|---|--------|
| APROXIMATE ENTROPY | 0.197412 | ✓ |
| BLOCK FREQUENCY | 0.009954 | No. |
| CUMULATIVE SUMS | FORWARD TEST:0.105981, REVERSE TEST: 0.148928 | ✓ |
| FFT | 0.543220 | ✓ |
| FREQUENCY | 0.075898 | ✓ |
| LINEAR COMPLEXITY | 0.863083 | ✓ |
| LONGEST RUN | 0.072816 | ✓ |
| NON OVERLAPPING TEMPLATE | P-VALORES ACEPTADOS: 147 DE 148 | ✓ |
| OVERLAPPING TEMPLATE | 0.173029 | ✓ |
| RANDOM EXCURSIONS | NO APLICABLE | |
| RANDOM EXCURSIONS VARIANT | NO APLICABLE | |
| RANK | 0.217328 | ✓ |
| RUNS | 0.953301 | ✓ |
| SERIAL | P-VALORES ACEPTADOS: 2 DE 2 | ✓ |
| UNIVERSAL | 0.993846 | ✓ |

4.3. Analisis de los valores H generados para el criterio 3.

Cuadro 6: Resultados de las pruebas de aleatoriedad NIST a los datos HdeXORcomp.dat .

| PRUEBA APLICADA | P-VALOR | EXITO? |
|---------------------------|---|--------|
| APROXIMATE ENTROPY | 0.161036 | ✓ |
| BLOCK FREQUENCY | 0.944077 | ✓ |
| CUMULATIVE SUMS | FORWARD TEST:0.294090, REVERSE TEST: 0.645302 | ✓ |
| FFT | 0.415401 | ✓ |
| FREQUENCY | 0.640142 | ✓ |
| LINEAR COMPLEXITY | 0.527570 | ✓ |
| LONGEST RUN | 0.948689 | ✓ |
| NON OVERLAPPING TEMPLATE | P-VALORES ACEPTADOS: 147 DE 148 | ✓ |
| OVERLAPPING TEMPLATE | 0.371232 | ✓ |
| RANDOM EXCURSIONS | NO APLICABLE | |
| RANDOM EXCURSIONS VARIANT | NO APLICABLE | |
| RANK | 0.301543 | ✓ |
| RUNS | 0.787370 | ✓ |
| SERIAL | P-VALORES ACEPTADOS: 2 DE 2 | ✓ |
| UNIVERSAL | 0.896941 | ✓ |

5. Conclusiones.

En general, los criterios 1 y 3 arrojan mejores resultados que el criterio 2. Si se tuviera que elegir entre el criterio 1 y el 3. Se puede concluir que el criterio 3 obtuvo un mejor desempeño de acuerdo a los p-valores obtenidos.

Para el caso de los valores de H , otra vez, se obtuvieron mejores resultados en el criterio 1 y en el criterio 3. Sin embargo, las pruebas "Random Excursions" y "Random Excursions Variant" no siempre se pudieron aplicar.

6. Anexos.

6.1. Código del criterio 1.

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

#define RENYI_MAP(var, parametro, j) ((var)*(parametro)+((var)>>(j)))

#define MAX 4294967295 /*El valor de 2^32-1.*/
#define TAMANIOCICLO 4294967296 /*EL valor de 2^32.*/
```



```

#define noMapas 4 /*NUmero de mapas a utilizar.*/
#define ITtotales 80000 /*Iteraciones totales para NIST.*/
#define tamanoH 20000 /*Tamano del arreglo de H.*/

/*
File:    main.c
Author: daniel
El siguiente programa es un ejemplo de cuatro mapas caoticos RENYI acoplados.
Se utiliza la suma para lograr el acoplado.
Para declarar nuestras variables, utilizamos los siguientes componentes:
-Un arreglo que guarda el valor de los parametros para cada mapa.
-Un arreglo que guarda el valor de los valores calculados para cada mapa.
*/

int main(){

    /*Declaramos los arreglos que vamos a utilizar para guardar esto.*/
    unsigned long Xn[noMapas];
    unsigned long Xtotal[ITtotales];
    unsigned long parametros[noMapas];
    unsigned int i;
    unsigned long arregloH[tamanoH];
    int epsilon;
    FILE *  archivobin;
    FILE *  archivoH;

    unsigned long H=0;
    unsigned int j=9; /*No mas de 16.*/
    unsigned long iteraciones=0;
    unsigned long IT = 80000;

    /* Apertura del fichero de destino, para escritura en binario.*/
    archivobin = fopen ("binarioSUMAmod.dat", "wb");
    if (archivobin==NULL)
    {
        perror("No se puede abrir binarioSUMAmod.dat");
        return -1;
    }

```

```

/* Apertura del archivo H para su escritura en binario.*/
archivoH = fopen ("HdeSuma.dat", "wb");
if (archivoH==NULL)
{
perror("No se puede abrir HdeSuma.dat");
return -1;
}

/*Inicializamos nuestros parametros, en este punto se aplica el uso
de una llave, en este ejemplo todavia no elegimos una. Tambien,
los parametros son fijos en este ejemplo.*/
parametros[0]=131071;
Xn[0]=653;
parametros[1]=104729;
Xn[1]=769;
parametros[2]=524287;
Xn[2]=227;
parametros[3]=65537;
Xn[3]=823;

int contDeH=0;

/*Primero, hacemos un ciclo inicial para calcular un nuevo valor para cada
uno de los mapas y calcular, por primera vez, el resultado de la operacion
XOR.*/
for( i =0; i<noMapas; i++){
    Xn[i]= RENYI_MAP(Xn[i],parametros[i],j);
    H+=Xn[i];
}

arregloH[contDeH]=H;
contDeH++;

/*Tambien al primer H se le aplica el mod.*/
H%=256;

/*Elegimos un epsilon aleatorio entre 1 y 16 (para operaciones de 32 bits).

```

```

En este caso, elegimos uno que este entre 0 y 15.*/
epsilon = 1;

unsigned int k;
unsigned long newH;

do {

    newH = 0;
    for(k=0;k<noMapas; k++){
        Xn[k]= RENYI_MAP(Xn[k],parametros[k],j) + epsilon*H;
        Xtotal[iteraciones++] = Xn[k];

        newH+=(Xn[k]);
    }
    H = newH;
    arregloH[contDeH]=H;

printf("\nla iteracion %d para %lu H es: \n", contDeH,arregloH[contDeH] );

    contDeH++;

    /*BUscamos un H pequeño: la perturbación no debe ser
    tan fuerte. Por tal motivo, se aplica una operación módulo.*/
    H%=256;

    /*Ahora, elegimos un valor para epsilon.*/
    epsilon = (H & 1)?1:-1;

    //printf("%d epsilon %lu \n", epsilon, iteraciones-1);

} while (iteraciones < IT);

/*Escribimos la informacion.*/
fwrite(Xtotal,4,80000,archivobin);
fwrite(arregloH,4,20000,archivoH);

if(!fclose(archivobin)){

```

```

        printf( "\nArchivo binario de Mapas cerrado\n" );
    }
    else{
        printf( "\nError: Archivo binario de Mapas no cerrado \n" );
        return 1;
    }

    if(!fclose(archivoH)){
        printf( "\nArchivo binario DE H cerrado\n" );
    }
    else{
        printf( "\nError: Archivo binario DE H no cerrado \n" );
        return 1;
    }

return 0;
}

```

6.2. Código del criterio 2.

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define RENYI_MAP(var, parametro, j) ((var)*(parametro)+((var)>>(j)))
#define MAX 4294967295 /*El valor de 2^32-1.*/
#define TAMANIOCICLO 4294967296 /*EL valor de 2^32.*/
#define noMapas 4 /*NUmero de mapas a utilizar.*/
#define ITtotales 80000 /*Iteraciones totales para NIST.*/
#define tamanoH 20000 /*Tamano del arreglo de H.*/

/*
File:    main.c
Author: daniel
El siguiente programa es un ejemplo de cuatro mapas caoticos RENYI acoplados.
Se utiliza la suma para lograr el acoplado.
Para declarar nuestras variables, utilizamos los siguientes componentes:
-Un arreglo que guarda el valor de los parametros para cada mapa.
-Un arreglo que guarda el valor de los valores calculados para cada mapa.
*/

int main(){

```

```

/*Declaramos los arreglos que vamos a utilizar para guardar esto.*/
unsigned long Xn[noMapas];
unsigned long Xtotal[ITtotales];
unsigned long parametros[noMapas];
unsigned long arregloH[tamanoH];
unsigned int i;
int epsilon;
FILE * archivobin;
FILE * archivoH;

unsigned long H=0;
unsigned int j=9; /*No mas de 16.*/
unsigned long iteraciones=0;
unsigned long IT = 80000;

/*Apertura del fichero de destino, para escritura en binario.*/
archivobin = fopen ("binarioXORmod.dat", "wb");
if (archivobin==NULL)
{
perror("No se puede abrir binarioXORmod.dat");
return -1;
}

/* Apertura del archivo H para su escritura en binario.*/
archivoH = fopen ("HdeXOR.dat", "wb");
if (archivoH==NULL)
{
perror("No se puede abrir HdeXOR.dat");
return -1;
}

/*Inicializamos nuestros parametros, en este punto se aplica el uso
de una llave, en este ejemplo todavia no elegimos una. Tambien,
los parametros son fijos en este ejemplo.*/
parametros[0]=131071;
Xn[0]=653;
parametros[1]=104729;
Xn[1]=769;
parametros[2]=524287;
Xn[2]=227;
parametros[3]=65537;
Xn[3]=823;
int contDeH=0;

```

```

/*Primero, hacemos un ciclo inicial para calcular un nuevo valor para cada
uno de los mapas y calcular, por primera vez, el resultado de la operacion
XOR.*/
for( i =0; i<noMapas; i++){
    Xn[i]= RENYI_MAP(Xn[i],parametros[i],j);
    H^=Xn[i];
}

arregloH[contDeH]=H;
contDeH++;

/*Tambien al primer H se le aplica el mod.*/
H%=256;

/*Elegimos un epsilon aleatorio entre 1 y 16 (para operaciones de 32 bits).
En este caso, elegimos uno que este entre 0 y 15.*/
epsilon = 1;

unsigned int k;
unsigned long newH;

do {

    newH = 0;
    for(k=0;k<noMapas; k++){
        Xn[k]= RENYI_MAP(Xn[k],parametros[k],j) + epsilon*H;
        Xtotal[iteraciones++] = Xn[k];
        newH^=Xn[k];
    }
    H = newH;
    arregloH[contDeH]=H;
    printf("\nla iteracion %d para %lu H es: \n", contDeH,arregloH[contDeH] );
    contDeH++;
    /*BUscamos un H pequeño: la perturbación no debe ser
    tan fuerte. Por tal motivo, se aplica una operación módulo.*/
    H%=256;

    /*Ahora, elegimos un valor para epsilon.*/
    epsilon = (H & 1)?1:-1;

} while (iteraciones < IT);

```

```

/*Escribimos la informacion.*/
fwrite(Xtotal,4,80000,archivobin);
fwrite(arregloH,4,20000,archivoH);

if(!fclose(archivobin)){
    printf( "\nArchivo binario cerrado\n" );
}
else{
    printf( "\nError: Archivo binario no cerrado \n" );
    return 1;
}

if(!fclose(archivoH)){
    printf( "\nArchivo binario DE H cerrado\n" );
}
else{
    printf( "\nError: Archivo binario DE H no cerrado \n" );
    return 1;
}

return 0;
}

```

6.3. Código del criterio 3.

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define RENYI_MAP(var, parametro, j) ((var)*(parametro)+((var)>>(j)))
#define MAX 4294967295 /*El valor de 2^32-1.*/
#define TAMANIOCICLO 4294967296 /*EL valor de 2^32.*/
#define noMapas 4 /*NUmero de mapas a utilizar.*/
#define ITtotales 80000 /*Iteraciones totales para NIST.*/
#define tamañoH 20000 /*Tamaño del arreglo de H.*/

```

/*

File: main.c

Author: daniel

El siguiente programa es un ejemplo de cuatro mapas caóticos RENYI acoplados.

Para declarar nuestras variables, utilizamos los siguientes componentes:

-Un arreglo que guarda el valor de los parámetros para cada mapa.

-Un arreglo que guarda el valor de los valores calculados para cada mapa.

```
*/
```

```
int main(){
```

```
    /*Declaramos los arreglos que vamos a utilizar para guardar esto.*/
```

```
    unsigned long Xn[noMapas];
```

```
    unsigned long Xtotal[ITtotales];
```

```
    unsigned long parametros[noMapas];
```

```
    unsigned long arregloH[tamanoH];
```

```
    unsigned int i;
```

```
    unsigned int epsilon;
```

```
    unsigned long gamma;
```

```
    unsigned long gammaComp;
```

```
    FILE * archivobin;
```

```
    FILE * archivoH;
```

```
    unsigned long H=0;
```

```
    unsigned int j=9; /*No mas de 16.*/
```

```
    unsigned long iteraciones=0;
```

```
    unsigned long IT = 80000;
```

```
    /* Apertura del fichero de destino, para escritura en binario.*/
```

```
    archivobin = fopen ("binarioXORcomp.dat", "wb");
```

```
    if (archivobin==NULL)
```

```
    {
```

```
        perror("No se puede abrir binarioXORcomp.dat");
```

```
        return -1;
```

```
    }
```

```
    /* Apertura del archivo H para su escritura en binario.*/
```

```
    archivoH = fopen ("HdeXORcomp.dat", "wb");
```

```
    if (archivoH==NULL)
```

```
    {
```

```
        perror("No se puede abrir HdeXORcomp.dat");
```

```
        return -1;
```

```
    }
```

```
    /*Inicializamos nuestros parametros, en este punto se aplica el uso
```

```
    *de una llave, en este ejemplo todavia no elegimos una. Tambien,
```

```
    *los parametros son fijos en este ejemplo.*/
```

```
    parametros[0]=131071;
```



```

Xn[0]=653;
parametros[1]=104729;
Xn[1]=769;
parametros[2]=524287;
Xn[2]=227;
parametros[3]=65537;
Xn[3]=823;
int contDeH=0;

/*Primero, hacemos un ciclo inicial para calcular un nuevo valor para cada
uno de los mapas y calcular, por primera vez, el resultado de la operacion
XOR.*/
for( i =0; i<noMapas; i++){
    Xn[i]= RENYI_MAP(Xn[i],parametros[i],j);
    H^=Xn[i];
}

arregloH[contDeH]=H;
contDeH++;

/*Elegimos un epsilon aleatorio entre 1 y 16 (para operaciones de 32 bits).
En este caso, elegimos uno que este entre 0 y 15.*/
epsilon = 5;

gamma= pow(2,epsilon);
gamma-=1;
gammaComp= MAX-gamma;

unsigned int k;
unsigned long newH;

do {

    newH = 0;
    for(k=0;k<noMapas; k++){
        Xn[k]= gammaComp^RENYI_MAP(Xn[k],parametros[k],j)+ gamma^H;
        Xtotal[iteraciones++] = Xn[k];
        newH^=Xn[k];
    }
    H = newH;
    arregloH[contDeH]=H;
    printf("\nla iteracion %d para %lu H es: \n", contDeH,arregloH[contDeH] );
    contDeH++;
}

```

```

        /*Ahora, elegimos un valor para epsilon entre 1 y 8 bits.*/
        epsilon = (H % 8) + 1;

    } while (iteraciones < IT);

    /*Escribimos la informacion.*/
    fwrite(Xtotal,4,80000,archivobin);
    fwrite(arregloH,4,20000,archivoH);

    if(!fclose(archivobin)){
        printf( "\nArchivo binario cerrado\n" );
    }
    else{
        printf( "\nError: Archivo binario no cerrado \n" );
        return 1;
    }

    if(!fclose(archivoH)){
        printf( "\nArchivo binario DE H cerrado\n" );
    }
    else{
        printf( "\nError: Archivo binario DE H no cerrado \n" );
        return 1;
    }

return 0;
}

```