MANUAL TÉCNICO
“SECURE CHAT”

CONTENIDO

INTRODUCCIÓN

La finalidad de todo manual técnico es la de proporcionar al lector la lógica con la que se ha desarrollado una aplicación, la cual se sabe que es propia de cada programador; por lo que se considera necesario ser documentada

Este manual describe los pasos necesarios para cualquier persona que tenga ciertas bases de sistemas pueda realizar la instalación de la aplicación, creada para establecer la comunicación entre dos personas a través de un canal de comunicación seguro. El presente manual técnico tiene como finalidad describir el diseño del prototipo para la gestión de SecureChat en ambientes web.

La implementación de SecureChat en ambientes móviles (web) se basa en una adaptación para del mismo tipo, es decir para dispositivos de cómputo, ya que se han implementado diversas aplicaciones que están en en el mismo ramo que SecureChat, y éstas han dado excelentes resultados de uso. Las aplicaciones de éste tipo arrojan resultados favorables en las personas que las consumen, es por ello que se optó por la implementación de dicha aplicación de la manera ya mencionada.

La implementación de la aplicación cuenta con una protección de la información por lado del servidor, así como también se toma en cuenta los posibles ataques que pueden surgir, y la utilización de la misma es responsabilidad de cada usuario. Resulta ser bastante fácil de implementar puesto solo es necesario instalar los paquetes básicos disponibles en cualquier distribución de Ubuntu.
Es importante tener en cuenta que en el presente manual se hace mención a las especificaciones mínimas de hardware y software para la correcta instalación de la aplicación.

OBJETIVO:

Proporcionar una guía para el lector, del desarrollo de la interfaz y de la instalación del sitio web del ISTU.

Aspectos del Análisis

Si bien el correo electrónico y los diversos servicios que nos brinda la Nube para intercambiar datos siguen constituyendo herramientas muy válidas cuando deseamos trasmitir cantidades relativamente grandes y detalladas de información, en la era de las comunicaciones prima la inmediatez.

En este sentido, la mensajería instantánea está más en auge que nunca, pues nos ofrece la posibilidad de entablar rápidamente contacto con otros usuarios tanto a través de equipos informáticos como de móviles de factura reciente.

La aplicación web se puede utilizar en dispositivos PC que soporten el sistema operativo LINUX en cualquiera de sus versiones. En estos tiempos en que la tecnología ha avanzado la mayoría de los dispositivos pueden accesar a la obtención de dicho sistema operativo.
La aplicación está orientada a éste sistema operativo debido a que es más confiable y es mayormente utilizado para el desarrollo de aplicaciones del mismo tipo que la nuestra. Es por ello que esta aplicación va orientada a ese tipo de dispositivos. Una de las ventajas que se presentan en este tipo de tecnología es que tiene mayor seguridad, confiabilidad y por eso muchos usuarios la utilizan.
El motivo de usar este sistema operativo, se debe a la creciente popularidad la cual incrementa a un ritmo exponencial que conlleva esta plataforma respecto a sus competidores, también otra de las grandes ventajas que tiene esta plataforma es que se habla de un software libre

Especificaciones Técnicas

1. REQUERIMIENTOS MÍNIMOS DE HARDWARE

- Procesador: Core
- Memoria RAM: 1 Gigabytes (GB) (Mínimo)
- Disco Duro: 500 Gb.

2. REQUERIMIENTOS MÍNIMOS DE SOFTWARE

- Sistema Operativo: Linux Ubuntu 12.04 Precise Pangolin o versiones superiores.
- Lenguaje de Programación: C
- Entorno de programación y compilador: GNU/gcc

Herramientas utilizadas para el desarrollo

- ## Lenguaje C

  Es el lenguaje de programación de propósito general asociado al sistema operativo UNIX

  Es un lenguaje de medio nivel. Trata con objetos básicos como caracteres, números. . .; también con bits y direcciones de memoria

  Posee una gran portabilidad

  Se utiliza para la programación de sistemas: construcción de intérpretes, compiladores, editores de texto, etc.

El lenguaje C consta de:

El lenguaje C propiamente dicho: tipos de datos, expresiones y estructuras de control

Extensiones en forma de macros y un amplio conjunto de librerías predefinidas

- ## Arquitectura cliente / servidor

  Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes. Ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos.

  - ✓ Facilita la integración entre sistemas diferentes y comparte información, permitiendo por ejemplo que las máquinas ya existentes puedan ser utilizadas pero utilizando interfaces más amigables el usuario. De esta manera, se puede integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operativo.

- ✓ Al favorecer el uso de interfaces gráficas interactivas, los sistemas construidos bajo este esquema tienen una mayor y más intuitiva con el usuario. En el uso de interfaces gráficas para el usuario, presenta la ventaja, con respecto a uno centralizado, de que no siempre es necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.
- ✓ La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

INSTALACIÓN DE GCC EN UBUNTU/LINUX

Descargamos los archivos

gcc-5.3.0.tar.bz2 | ecj-4.9.jar

Extraemos y configuramos

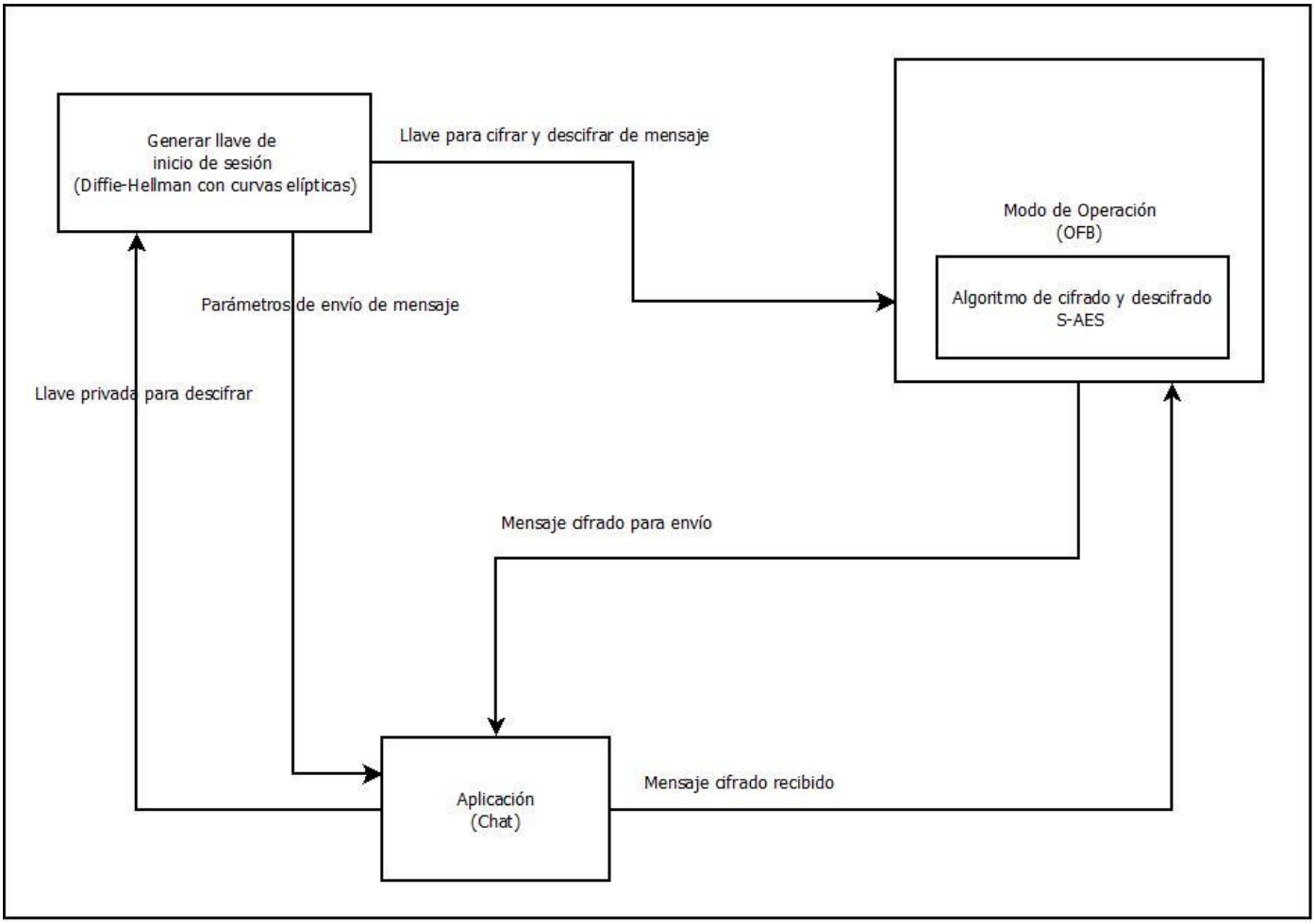
```
$ tar jxvf gcc-5.3.0.tar.bz2
$ cp ecj-4.9.jar gcc-5.3.0/ecj.jar
$ mkdir gcc-build_5.3.0
$ cd gcc-build_5.3.0
$ ../gcc-5.3.0/configure --enable-shared --enable-threads=posix --enable-__cxa_atexit \
--enable-clocale=gnu --enable-languages=fortran,java,objc --enable-java-awt=gtk --prefix=/opt/gcc-5.3.0
```

Comandos utilizados:

cp ecj-4.9.jar gcc-5.3.0/ecj.jar : Copiamos el archivo jar que contiene el compilador Java al directorio raíz del código fuente de **GCC**. Dicho compilador es utilizado por **GCJ** para analizar los archivos de código fuente escritos en **Java**. En el proceso de instalación, dicho archivo será ubicado en **/opt/gcc-5.3.0/share/java/ecj.jar**, y **GCJ** lo utilizará a través de un ejecutable binario ubicado en la ruta **/opt/gcc-5.3.0/libexec/gcc/i686-pc-linux-gnu/5.3.0/ecj1**.

mkdir gcc-build_5.3.0 : Creamos un directorio de compilación, ya que **GCC** no permite que se compile directamente en el directorio de las fuentes.
--enable-shared: Compila las librerías compartidas.
--enable-threads=posix: Selecciona la librería genérica POSIX/Unix98 para el soporte de hilos.
--enable-__cxa_atexit: Opción necesaria para una correcta compilación de c++.
--enable-clocale=gnu: Evita un error en la generación de las locales, en el caso de que estén incompletas.
--enable-languages=fortran,java,objc : Compila los lenguajes de programación **Fortran**, **Java**, y **Objetive C**, además de los predefinidos, **C** y **C++**.
--enable-java-awt=gtk: Añade el soporte de **AWT** con **GTK+** como librería de interfaz gráfica a **Libgcj**.
--prefix=/opt/gcc-5.3.0: Instala el compilador en **/opt/gcc-5.3.0**.

ARQUITECTURA PRESENTADA PARA LA APLICACIÓN



El primer módulo, se diseñó para generar las llaves de inicio de sesión, cabe destacar que éstas llaves serán diferentes y nuevas cada que se solicite por parte de los usuarios el inicio de sesión.

El segundo módulo de modos de operación se utilizará para cifrar y descifrar el mensaje.

Por último el módulo de chat, será utilizado para poder realizar el envío de mensajes entre los dos usuarios que estén realizando una conversación.

El primer módulo generará las llaves de inicio de sesión, dichas llaves se estarán generando con el protocolo Diffie-Hellman implementado sobre curvas elípticas, cada que se solicite por parte de los usuarios el inicio de sesión para iniciar la comunicación entre usuarios, se generará una llave de sesión automática, ésta tendrá que ser compartida entre los dos usuarios que van a conversar para que así puedan cifrar y descifrar los mensajes que se envíen y reciban, cabe resaltar que esta llave se genera a partir de punto generador sobre la curva, ésta llave se debe enviar al otro módulo que es el de modo de operación, para que así se pueda proceder con el proceso de cifrado y descifrado, de ésta manera se pueda llevar a cabo dicha operación. Un detalle del módulo de modo de operación, es que el algoritmo no admite los puntos que el primer módulo le envía, es por eso que se tomarán los dos valores de los puntos y se convertirán en cadenas binarias, ya que el algoritmo que ocupamos para realizar el proceso de cifrado y descifrado es S-AES, el tamaño de bloque de la llave es de 16 bits, lo que significa que podremos representar los dos puntos “(x , y)” como números binarios de 8 bits y así poder ocupar los puntos en S-AES.

ESPECIFÍCACION DE LOS ARCHIVOS DE LA APLICACIÓN "cliente.c, server.c y ElipticCurves.c"

CLIENTE	cliente.c

/*
3CM2
Aguirre Cruz Eder Jonathan
Saules Cortes Jhonatan
Armenta García Guadalupe Javier
Cárdenas Castillo Victor Hugo
*/
 	
#include"stdio.h"
#include"stdlib.h"
#include"sys/types.h"
#include"sys/socket.h"
#include"string.h"
#include"netinet/in.h"
#include"netdb.h"
#include"pthread.h"
#include "DiffieC.h"
#include "socketHandle.h"
#define PORT 3490
#define BUF_SIZE 2000
 	
void * receiveMessage(void * socket) {
int sockfd, ret;
char buffer[BUF_SIZE];
sockfd = (int) socket;
memset(buffer, 0, BUF_SIZE);
for (;;) {
ret = recvfrom(sockfd, buffer, BUF_SIZE, 0, NULL, NULL);
if (ret < 0) {
printf("Error receiving data!\n");
} else {
printf("Servidor: ");
	/* Programa ejecutable que participa activamente en el establecimiento de las conexiones. Envía una petición junto con la llave de inicio de sesión al servidor y se queda esperando por una respuesta. Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo.*/

```
   fputs(buffer, stdout);
  }
 }
}

int main(int argc, char**argv) {
 struct sockaddr_in addr, cl_addr;
 int sockfd, ret;
 char buffer[BUF_SIZE];
 char * serverAddr;
 pthread_t rThread;

 if (argc < 2) {
  printf("usage: client < ip address >\n");
  exit(1);
 }

 serverAddr = argv[1];

 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfd < 0) {
  printf("Error creating socket!\n");
  exit(1);
 }
 printf("Socket created...\n");

 memset(&addr, 0, sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_addr.s_addr = inet_addr(serverAddr);
 addr.sin_port = PORT;

 ret = connect(sockfd, (struct sockaddr *) &addr,
sizeof(addr));
 if (ret < 0) {
  printf("Error connecting to the server!\n");
  exit(1);
 }
 printf("Connected to the server...\n");
 memset(buffer, 0, BUF_SIZE);

  int gx,gy,Ax,Ay, Bx,By,b=getRandom(),csX,csY;

  gx=receiveDiffie(sockfd,buffer);
  gy=receiveDiffie(sockfd,buffer);
  Ax=receiveDiffie(sockfd,buffer);
  Ay=receiveDiffie(sockfd,buffer);
```

/*Creación del socket para establecer la comunicación con el servidor */

/*Función receiveDiffie, es la función donde se recibe la llave publica del otro usuario para poder realizar el envío de mensajes */

/*Función sendDiffie, es la función donde se envía la llave publica del usuario para poder ser enviada al destinatario y pueda abrir mensajes enviado */

```
  privateKey(&gx,&gy,&Bx,&By,b);
  sendDiffie(sockfd,buffer,Bx,addr,sizeof(addr));
  sendDiffie(sockfd,buffer,By,addr,sizeof(addr));
  privateKey(&Ax,&Ay,&csX,&csY,b);

  printf("A(%d,%d,)*(b=%d): Cs(%d,%d)\n",Ax,Ay,b,csX,csY
);




 //creating a new thread for receiving messages from the
server
 ret = pthread_create(&rThread, NULL, receiveMessage,
(void *) sockfd);
 if (ret) {
  printf("ERROR: Return Code from pthread_create() is
%d\n", ret);
  exit(1);
 }

 while (fgets(buffer, BUF_SIZE, stdin) != NULL) {
  ret = sendto(sockfd, buffer, BUF_SIZE, 0, (struct sockaddr
*) &addr, sizeof(addr));
  if (ret < 0) {
   printf("Error sending data!\n\t-%s", buffer);
  }
 }

 close(sockfd);
 pthread_exit(NULL);

 return 0;
}
```

SERVIDOR	server.c
/* 3CM2 Aguirre Cruz Eder Jonathan Saules Cortes Jhonatan Armenta Garcia Guadalupe Javier	/*Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la

Cardenas Castillo Victor Hugo
*/
 	
#include"stdio.h"
#include"stdlib.h"
#include"sys/types.h"
#include"sys/socket.h"
#include"string.h"
#include"netinet/in.h"
#include"pthread.h"
#include "DiffieC.h"
#include "socketHandle.h"
#define PORT 3490
#define BUF_SIZE 2000
#define CLADDR_LEN 100
 	
void * receiveMessage(void * socket) {
 int sockfd, ret;
 char buffer[BUF_SIZE];
 sockfd = (int) socket;
 memset(buffer, 0, BUF_SIZE);
 for (;;) {
 ret = recvfrom(sockfd, buffer, BUF_SIZE, 0, NULL, NULL);
 if (ret < 0) {
 printf("Error receiving data!\n");
 } else {
 printf("Cliente: ");
 fputs(buffer, stdout);
 }
 }
}
 	
void main() {
 struct sockaddr_in addr, cl_addr;
 int sockfd, len, ret, newsockfd;
 char buffer[BUF_SIZE];
 pid_t childpid;
 char clientAddr[CLADDR_LEN];
 pthread_t rThread;
 	
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfd < 0) {
 printf("Error creating socket!\n");
 exit(1);
	petición desde la red, realiza el servicio y devuelve el resultado al solicitante. Al ser posible implantarlo como aplicaciones de programas, puede ejecutarse en cualquier sistema donde exista TCP/IP y junto con otros programas de aplicación. El servidor comienza su ejecución antes de comenzar la interacción con el cliente. Su tiempo de vida o de interacción es “interminable”.
	*/

```
 }
 printf("Socket created...\n");

 memset(&addr, 0, sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_addr.s_addr = INADDR_ANY;
 addr.sin_port = PORT;

 ret = bind(sockfd, (struct sockaddr *) &addr, sizeof(addr));
 if (ret < 0) {
  printf("Error binding!\n");
  exit(1);
 }
 printf("Binding done...\n");

 int gx,gy,a=getRandom(),Ax,Ay,Bx,By,csX,csY;
 findGenerator(&gx,&gy);
 privateKey(&gx,&gy,&Ax,&Ay,a);
 printf("G(%d,%d,)*%d: (%d,%d)\n",gx,gy,a,Ax,Ay );

 printf("Waiting for a connection...\n");
 listen(sockfd, 5);

 len = sizeof(cl_addr);
 newsockfd = accept(sockfd, (struct sockaddr *) &cl_addr,
&len);
 if (newsockfd < 0) {
  printf("Error accepting connection!\n");
  exit(1);
 }

 inet_ntop(AF_INET, &(cl_addr.sin_addr), clientAddr,
CLADDR_LEN);
 printf("Connection accepted from %s...\n", clientAddr);
 memset(buffer, 0, BUF_SIZE);


  sendDiffie(newsockfd,buffer,gx,cl_addr,len);
  sendDiffie(newsockfd,buffer,gy,cl_addr,len);
  sendDiffie(newsockfd,buffer,Ax,cl_addr,len);
  sendDiffie(newsockfd,buffer,Ay,cl_addr,len);
  Bx=receiveDiffie(newsockfd,buffer);
  By=receiveDiffie(newsockfd,buffer);
  privateKey(&Bx,&By,&csX,&csY,a);
  printf("B(%d,%d,)*(a=%d): cs:(%d,%d)\n",Bx,By,a,csX,csY
);
```

/*Se llaman a las funciones findGenerator para encontrar el punto generador bajo la curva y privateKey para enviar ese punto y obtener la llave privada para el servidor */

printf("Enter your messages one by one and press return key!\n");
 	
//creating a new thread for receiving messages from the client
ret = pthread_create(&rThread, NULL, receiveMessage, (void *) newsockfd);
if (ret) {
printf("ERROR: Return Code from pthread_create() is %d\n", ret);
exit(1);
}
 	
while (fgets(buffer, BUF_SIZE, stdin) != NULL) {
ret = sendto(newsockfd, buffer, BUF_SIZE, 0, (struct sockaddr *) &cl_addr, len);
if (ret < 0) {
printf("Error sending data!\n");
exit(1);
}
}
 	
close(newsockfd);
close(sockfd);
 	
pthread_exit(NULL);
return;
}
	/* sendDiffie función que envía la llave publica al cliente para poder realizar el envío de un mensaje */
	
	/*receiveDiffie función que recibe la llave publica del cliente para poder terminar la recepción de un nuevo mensaje */

Curvas Elípticas	ElipticCurves.c

/*
3CM2
Aguirre Cruz Eder Jonathan
Saules Cortes Jhonatan
Armenta Garcia Guadalupe Javier
Cardenas Castillo Victor Hugo
*/
#include "ElipticCurves.h"
 	
int MultMod(int a, int b, int n){
int v=0;
	/* Contiene las operaciones que se realizan con puntos generados sobre las curvas elípticas */

```
	if(n>=2){
		v=a*b;
		if(v<0){
			v=abs(v);
			v=n-(v%n);
			if(v==n)
				v=v%n;
		}
		else
			v=v%n;
		return v;
	}
	else
		return -1;
}
int SumaMod(int a, int b, int n){
	int v=0;
	if(n>=2){
		v=a+b;
		if(v<0){
			v=abs(v);
			v=n-(v%n);
			if(v==n)
				v=v%n;
		}
		else
			v=v%n;
		return v;
	}
	else
		return -1;
}

void savePointsFile(char * file){
	int i,j,re[102],y2[102];
	int contador=0;
	char punt[500],p[20];
	FILE *archivo;
	printf("\n \n \n La curva es: y^2=x^3+2x+5");
	printf("\n \n Calculando Residuos cuadraticos...\n
\n");
	for(i=1; i<103; i++){
		re[i]=MultMod(i,i,103);
	printf("\n Residuo de %d ^2 es %d",i,re[i]);
	}
```

/* Calcula los residuos cuadráticos que existen sobre un campo determinado*/

```
        printf( "\n \n Calculando Puntos sobre la curva...\n
\n");

        archivo=fopen(file,"w");
        for(j=1; j<103; j++){                   y2[j]=
SumaMod(MultMod(MultMod(j,j,103),j,103),SumaMod(M
ultMod(2,j,103),5,103), 103);                   for(i=1; i<103;
i++){
                        if(y2[j]==re[i]){
                                contador++;


                                fprintf(archivo,"%d %d\n",j,i);

                        }
                }
        }
        printf("Fueron %d puntos\n", contador);
        fclose(archivo);

}
int* readPointsFile(char * file ,int* rows, int* cols){

        FILE *fichero;

        int i,j,numPuntos=0;
        fichero=fopen(file,"r");
        while ( fscanf(fichero,"%d %d\n",&j,&i)  != EOF) {
     numPuntos++;
   }
   printf("Total de puntos: %d \n",numPuntos );
        *rows= numPuntos;
        *cols=2;
   int*  matrixPuntos = crearMatriz(numPuntos,2);
   readMatrix(matrixPuntos,numPuntos,2,file);

   printf("Imprimiendo matriz....\n");
   imprimeMatriz(matrixPuntos,numPuntos,2 );
   return matrixPuntos;
}
void  saveGeneradores(char *file,int* matrixPuntos,int
rows,int cols,int p,int a){
        printf("La matriz contiene %d puntos\n",rows );
        int *elmsGrupo= creaArray(rows);        int
*auxiliar = creaArray(rows);
        int i ;
```

/*Ciclo FOR que recorre todos los puntos sobre el campo Zp */
/*Sustituimos los valores de x para obtener y^2, los valores de x van del 0 al 102 en la ecuación de la curva elíptica x^3 + 2x + 5 mod 103 */

/* Se archivan en una matriz dentro de un archivo todos los puntos encontrados sobre la curva */

```
	for (i = 0; i < rows; i++){
		if(validaGenerador(matrixPuntos,rows,cols,
i, auxiliar, p,a) == 1){
			elmsGrupo[i]++;
		}
	}

	FILE *fichero;
	fichero=fopen(file,"w");
	printf("Imprimiendo generadores\n");
	for(i=0; i< rows;i++){
		if (elmsGrupo[i] !=0){
			printf("pos %d : %d\n",i,
elmsGrupo[i] );
			fprintf(fichero,"%d
%d\n",devuelveM(matrixPuntos,cols,i, 0
),devuelveM(matrixPuntos,cols,i, 1 ) );
		}
	}
	fclose(fichero);
}
int validaGenerador(int* matrixPuntos,int rows,int cols,int
punto,int * auxiliar, int p,int a){
	printf("Rellenando con ceros el arreglo
auxiliar...\n");
	int i ;
	for(i=1 ; i<rows; i++)		{
		auxiliar[i]=0;  }
	int auxX=0,auxY=0;
	auxX= devuelveM(matrixPuntos,cols,punto, 0 );
	auxY= devuelveM(matrixPuntos,cols,punto, 1 );
	auxiliar[punto]=1;
	printf("Verificando si el punto %d es generador\n",
punto );
	for (i = 0; i < rows; i++){
		printf("iteracion %d \n",i );
		printf("Sumando punto %d  consigo mismo
...\n", punto);

	sumaPuntos(devuelveM(matrixPuntos,cols,punto,
0 ),devuelveM(matrixPuntos,cols,punto, 1
),auxX,auxY,&auxX,&auxY,p,a);

		printf("Buscando punto generado para
actualizar arreglo auxiliar\n");
```

/*Ya que utilizamos el mismo arreglo para los elementos de la curva se inicializa con valores de ceros*/

```
        buscaPunto(auxiliar,matrixPuntos,rows,cols,auxX
,auxY);

        }

        printf("Verificando si el punto puesto a prueba es
generador\n");
        for(i=0; i < rows;i++){
                if(auxiliar[i]== 0){
                        return -1;
                }
        }
                return 1;
}

void buscaPunto(int* auxiliar,int* matrixPuntos,int
rows,int cols,int auxX,int auxY){
        int i,j;
        for (i = 0; i < rows; i++){

                if( auxX== devuelveM(matrixPuntos,cols,i,
0 ) && auxY== devuelveM(matrixPuntos,cols,i, 1 ) ){
                        printf("genere 1 punto\n");
                        auxiliar[i]=1;
                }
        }

}
void sumaPuntos(int x1, int y1, int x2, int y2, int* x3, int
*y3, int p,int a){ //p == primo
        int xp=0,yp=0,l=0;
        //printf("Suma de puntos\n");
        //printf("P(%d,%d) + P(%d,%d)=",x1,y1,x2,y2 );


        if(x2 == -1 && y2 == -1){
                *x3=x1;
                *y3=y1;
        }
        else{


        //caso 1 x1 == x2
        if(x1 == x2){
                //caso 1.1 y1 == y2
```

/*ahora se recorre el arreglo para ver si genero todos los puntos sobre la curva*/

/*Función que almacena, sólo a los puntos generadores de la curva establecida */

```
                if(y1 == y2){

                        l=multiplicacion(
adition(multiplicacion(3,multiplicacion(x1,x1,p),p) ,a,p)  ,
inverseMultiplicative(multiplicacion(2,y1,p),p) ,p);
                        xp= substraction(
substraction(multiplicacion(l,l,p), x1,p)  , x2 , p);
                        yp= substraction(
multiplicacion(l,substraction(x1,xp,p),p) ,y1,p)  ;
                }
                //caso 1.2
                if(  y1+y2 == p ){

                        //punto en el infinito O
                        printf("Punto en el infinito\n");
                        xp=-1;
                        yp=-1;

                }

        }
        //caso 2 x1 != x2
        else{

                l= multiplicacion( substraction(y2 , y1, p),
inverseMultiplicative( substraction(x2 , x1, p),  p) , p);
                xp= substraction(
substraction(multiplicacion(l,l,p), x1,p)  , x2 ,p );
                yp= substraction(
multiplicacion(l,substraction(x1,xp,p),p) ,y1,p)  ;
        }
        *x3=xp;
        *y3=yp;


        }

        //printf("P(%d,%d)\n",*x3,*y3 );

}
```

/*Función que realiza la suma de dos puntos sobre la curva distintos */

/*Función que realiza la multiplicación de dos puntos sobre la curva establecida */