TEORÍA DE LA INFORMACIÓN

INFORME de PROYECTO:
FIRMAS DIGITALES

Descripción del programa JCripto10 (cifrador)

Grupo de trabajo:
Gaudina, Gabriel
Tripicchio, Fernando
Sanchez, Eduardo

Índice de contenido

Comentarios previos sobre el cifrador

El cifrador fue programado utilizando el lenguaje Java en su versión 1.5.0_03 (Sun JDK1.5.0_03). Esto debe ser tenido en cuenta a la hora de la instalacíon de la máquina virtual (jvm) para la ejecución del programa, se recomienda el uso de Sun Java JRE 1.5.X_XX (o superior), ya que no garantizamos el buen funcionamiento del cifrador con otra máquina virtual que no sean las mencionadas (como en el caso de msjvm – MicroSoft Java Virtual Machine - o Sun JRE 1.4.X 1.3.X o 1.2.X), aunque en el caso particular de Sun JRE 1.4.3_08 el programa funciona correctamente al menos con los tipos de archivos mas utilizados (exe, doc, txt, zip, jpg, jar).

¿Por qué Java?
Porque por su potencia (estructuras implementadas), seguridad (implementada a traves del paradigma de objetos) y sobre todo independencia de plataforma (no nos obliga al uso de una arquitectura o sistema operativo), nos permite integrarlo en otras herramientas (como por ejemplo PHP) sin preocuparnos del servidor que lo ejecute y brindarle una interfaz mas amigable que la "línea de comandos" al usuario.

¿Por qué un método de cifrado nuevo?
Porque crear un método nuevo nos permitió ver cómo funciona todo el proceso de cifrado, y enfrentarnos a problemas de "seguridad" debido a fallas en el diseño para perfeccionarlo hasta el punto actual.

Fundamentos teóricos del cifrador

El cifrador presentado se basa en la función polinómica cúbica $x^2(b-x)$ y su representación gráfica es la siguiente:

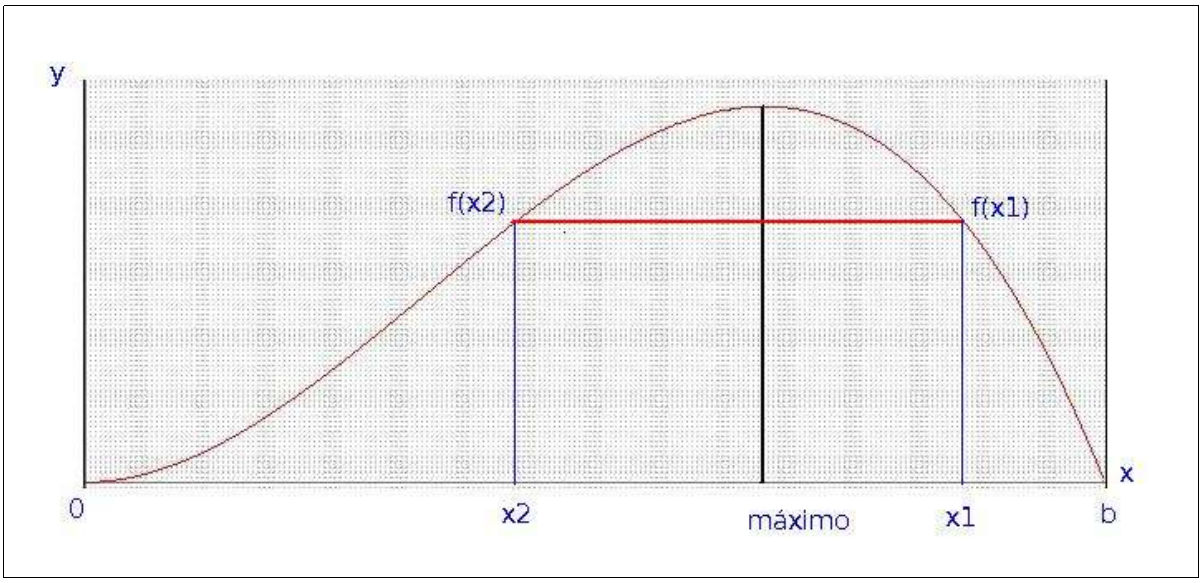

Ilustración - Función f(x)=x*x*(b-x)

Elegimos esta función debido a las siguientes particularidades:

El rango es limitado y parametrizado por el valor de "b" (y determina un máximo=2b/3).
Dado un valor para "x1" y calculando el correspondiente "f(x1)" se puede aproximar el correspondiente valor de "x2" (f(x1)=f(x2)), la exactitud de la aproximación depende del rango utilizado y fijado por el valor de "b".

En consecuencia, esta función nos permite, dando un valor a "b", asignar aleatoriamente un valor a "x1", en base al cual aproximar, a través de "f(x1)", el valor de "x2" (f(x2)=f(x1)+E, donde E=f(x1)-f(x2) el error en la aproximación) y así obtener dos "claves" (o mejor dicho, partes de claves) correspondientes una a la otra y permitir que el cifrado sea reversible.

Desde un punto de vista práctico, se puede dar a "b" un valor de un orden superior a, por ejemplo, 5000 bits (el número binario formado por un 1 seguido de 4999 ceros, es decir 2^{5000} = 1.4124670321394 26e+1505), el máximo estaría ubicado en $2(2^{5000})/3$ = 9.4164468809295069e+1504, dando un conjunto de 4.7082234404647535e+1504 números entre los que elegir "x1".

Si bien no hemos demostrado con fundamentos matemáticos firmes la seguridad del método implementado, sin duda el punto más fuerte que éste tiene es el tiempo computacional que insumiría "adivinar" el par de claves usados en todo el proceso del firmado digital (el "adivinar" las claves utilizando "fuerza bruta" implicaría generar un valor para "x1", obtener un valor para "x2" y así calcular el error E y tratar de descifrar un archivo).

Para el ejemplo suponemos lo siguiente:

El hacker ha decompilado el programa y sabe qué combinación de variables debe usar para cifrar-descifrar un archivo, conoce la función que las genera y sabe cómo armar los archivos que contienen las claves que el programa usa (sin embargo, NO conoce la estructura de un archivo cifrado).

Que del rango "[máximo, b]" descarte valores "poco probables", reduciendo el conjunto a, digamos, 1/1000 del total (es decir 4.7082234404647535e+1501 valores) lo cual es bastante pesimista para nosotros!

Que posea un poder de cálculo que le permita efectuar todos los pasos del "cracking" 1000000000000000000 (1 trillón) de veces por segundo.

En base a lo dicho, le tomaría unos 1.4929678591022176e+1473 MILENIOS!!!! "adivinar" las claves, mientras que para al programa le toma de uno a dos minutos en una computadora con un poder de cálculo mediano (PC hogareña) generar las claves de 5000 bits.

Funciones del cifrador

El cifrador cuenta con varias funcionalidades. Además de la generación de claves, cifrado, descifrado y firmado, puede también verificar las firmas, comprimir y descomprimir los archivos que se envían/reciben. Estas funcionalidades se detallan a continuación:

tener en cuenta que a los comandos abajo comentados se debe anteponer "java -jar** ..." para poder ejecutarlos.*

Generación de claves:

jcripto -g <cantidad de dígitos> <nombre de archivo de las claves>

Este comando le indica al cifrador que debe generar una clave pública y una privada de longitud igual a <cantidad de dígitos> (en realidad, el programa si bien genera claves de <cantidad de dígitos> de longitud, utiliza números de <cantidad de dígitos> x3, para compensar el descarte de dígitos producto de la aproximación de los parámetros mencionados mas arriba – f(x1) y f(x2)). Luego de generar dichas claves, el programa las almacena cada una en un archivo: "<nombre de archivo de las claves>.pri" y "<nombre de archivo de las claves>.pub" (clave privada y pública respectivamente).

Cifrado de un archivo:

jcripto -c <archivo a cifrar> <clave a utilizar>

En este caso se le indica al cifrador que tome el archivo <archivo a cifrar> y el archivo de clave <clave a utilizar> (un archivo de extensión "pri" o "pub", según se desee cifrar con la clave privada del remitente o con la clave pública del destinatario). Para tal fin se utiliza un método bastante simple (al menos en esta versión del programa) que consiste en "leer" el archivo <archivo a cifrar> en bloques de bytes de igual tamaño que la clave usada y contenida en <clave a utilizar>, aplicar entre ellos una operación lógica de XOR (hemos elegido ésta por ser reversible y por lo mismo asegurar la igualdad a nivel de bits del archivo original sin cifrar con el archivo descifrado), y "escribir" el bloque resultante en un archivo de salida llamado "<archivo a cifrar>.cif"

Descifrado de un archivo:

jcripto -d <archivo cifrado> <clave a utilizar>

Con estos parámetros el programa utiliza la clave "<clave a utilizar>" (un archivo .pri o .pub según corresponda) para descifrar el archivo "<archivo cifrado>" (utilizando la misma operación que se utilizó para cifrarlo, XOR) y generando un archivo cuyo nombre es igual al del archivo "<archivo cifrado>" pero sin la extensión "cif".

Firmado de un archivo:

jcripto -f <archivo>

Esto realiza el firmado del archivo "<archivo>" utilizando un método simple, con el cual se "lee" el archivo en bloques fijos de "n" bytes, y luego de suman estos bloques (tratándolos como números "grandes"), el resultado se almacena en un archivo "<archivo>.hash". El método utilizado es lo suficientemente preciso como para obtener un "firmado" distinto en caso de modificar incluso una letra en el archivo original.

Verificado de una firma:

jcripto -v <firma> <archivo descifrado>

En este caso el programa compara (a nivel de bits) la firma provista "<firma>" con la obtenida al realizar el "firmado" del archivo descifrado "<archivo descifrado>", si coinciden se establece la legitimidad del archivo recibido.

Compresión de los archivos que se enviarán:

jcripto -z <archivo1> <archivo2>

La intención de esta funcionalidad es la de comprimir (utilizando el algoritmo de Huffman-ZIP) el archivo cifrado y el que contiene la firma para que el envío por un medio digital (Internet, teléfono, etc.) sea menos costoso. El programa genera un archivo "zip" cuyo nombre corresponde al del primer archivo pasado como parámetro.

Descompresión de archivos recibidos:

jcripto -u <archivo>

Con esto el programa descomprime el archivo "<archivo>" en los dos originalmente comprimidos.

Comparado de dos archivos:

jcripto -cmp <archivo1> <archivo2>

Esta es una funcionalidad agregada con el solo propósito de verificación "manual" del correcto funcionamiento del cifrado principalmente al procesar archivos con formatos nuevos o de tamaños considerables.

Conclusión

La implementación del método descripto nos permitió entender el funcionamiento de todo el proceso de cifrado y firmado digital enfrentándonos a "problemas reales" en cuanto a seguridad dándonos una idea de lo que se busca en dicha área de la ciencia de la computación de una manera mas didáctica que simplemente implementar un método reconocido y en el que no hay algo para descubrir.