Análisis de un cifrador simétrico

Alfonso Manuel Mancilla Herrera*

Resumen

Matt Blaze, un sobresaliente investigador de los laboratorios AT&T y con mucha experiencia en el campo de la criptografía, realizó una propuesta sobre un cifrador de clave simétrica basado en un subproblema NP-completo. El autor describe una primitiva de seguridad sencilla y eficiente que tiene como base las redes Feistel. Además, propone dos criptosistemas de clave simétrica que usan dicha primitiva: Turtle y Hare.

El propósito fundamental de este artículo fue estudiar la propuesta de los cifradores Turtle y Hare. Nos apoyamos en la implementación expuesta por el autor con algunas modificaciones que realizamos, para el trabajo con archivos.

Palabras clave: Cifrador, redes Feistel, NP-completo, criptografía, criptoanálisis.

Abstract

Matt Blaze, is a outstanding researcher of the AT&T laboratories with much experience in the matter of cryptography, he made a proposal about a symmetric cipher based in a NP-complete subproblem. The author describes a simple and efficient security primitive based in Feistel Networks. Furthermore, he proposes two symmetric key cryptosystems that use this primitive: Turtle and Hare.

The main purpose of the present article is to study the proposal of the Turtle and Hare ciphers. We will support in the implementation exposed by the author with any modifications for the work with files.

Key words: Cipher, Feistel Networks, NP-complete, Cryptography and Cryptanalysis.

Fecha de recepción: 11 de marzo del 2002

INTRODUCCIÓN

Si analizamos el punto donde nos encontramos ahora en el ámbito de las comunicaciones y de acuerdo con esto realizamos una visión prospectiva en este campo,

* Licenciado en Ciencias de la Educación; Especialidad en Física y Matemática, Universidad del Atlántico, 1984; Ingeniero de Sistemas, Universidad del Norte, 1991; Especialista en Ingeniería del Software, Universidad Industrial de Santander, UIS, 1995; Msc en Ciencias de la Computación, ITESM-UNAB, 2002. Profesor del Departamento de Sistemas, Universidad del Norte *(amancill@uninorte.edu.co)*

fácilmente nos daremos cuenta de la importancia de realizar operaciones con el mayor grado de confiabilidad posible. Transacciones bancarias, comercio electrónico, instrucciones militares, manejo de información confidencial en las organizaciones a través de la web, etc., están expuestas a la vista de muchos que no deben participar en ella. Para evitar esto se pusieron en marcha elementos de protección a nivel de *software* de sistemas y de aplicaciones ubicados en el *hardware* de los equipos computacionales, que permiten detectar, e impedir el acceso, a los «intrusos». Lastimosamente, por diversos motivos, estos elementos de protección no son 100% seguros, y es precisamente ahí donde entran en juego otras medidas de seguridad; entre ellas tenemos la criptografía.

En la actualidad, la proliferación de poderosos computadores es considerada como un arma de doble filo. Así como nosotros podemos realizar operaciones con una mayor rapidez, también aquellos llamados «intrusos» lo pueden hacer y de esta manera violar medidas de seguridad implantadas. La criptografía se preocupa de que esto no ocurra; para ello intenta codificar los datos de tal manera que éstos se vean totalmente incomprensibles. La idea es que si el «intruso» llega a los datos violando su protección no pueda descubrir su significado real.

Hasta ahora se han desarrollado muchos cifradores, y para nadie es un secreto que los ataques contra ellos hacen parte de su evolución. La propuesta de Matt Blaze se aleja un tanto de lo tradicional, o por lo menos le adiciona un nuevo factor, la NP-completitud del algoritmo, lo que, según él, proporciona una primitiva de seguridad más confiable. Analicemos ahora dicho cifrador.

CONSIDERACIONES GENERALES

El cifrador propuesto a continuación tiene como características principales las siguientes:

- Clave simétrica
- Construido sobre redes Feistel
- Basado en un subproblema NP-completo

La primera especificación del cifrador no sólo hace referencia a la manera de distribución de la clave o llave, es decir, si es en forma privada o pública, sino también a cómo se va a realizar el diseño del cifrador. En este momento, muchos cifradores de clave simétrica son diseñados para resistir ataques criptoanalíticos conocidos; por este motivo, el análisis e implementación de estos cifradores es considerado como excepcionalmente difícil. *«Los algoritmos tienden a ser conceptualmente complejos, caracterizados por constantes mágicas, estructura irregular e ineficientes operaciones a nivel*

de bit. Es virtualmente imposible comprender completamente la justificación de varios parámetros».[1]

Las dos características restantes guardan estrecha relación; ellas muestran un cifrador con una primitiva de seguridad basada en redes Feistel (figura 1), para la cual cuando usamos ciclos mayores que dos, el recuperar su estado interno resulta ser un problema NP-completo. La estructura general para una red Feistel de cuatro ciclos, por ejemplo, es la siguiente:

$$
\begin{aligned}
R &= R + F1(L) \\
L &- L + F2(R) \\
R &= R + F3(L) \\
L &= L + F4(R)
\end{aligned}
$$

R y L son la mitad derecha e izquierda del bloque, respectivamente. F1, F2, F3 y F4 son funciones pseudoaleatorias (secretas) y el símbolo + representa la operación que se va a aplicar, usualmente el módulo 2^n o el OR-exclusivo.

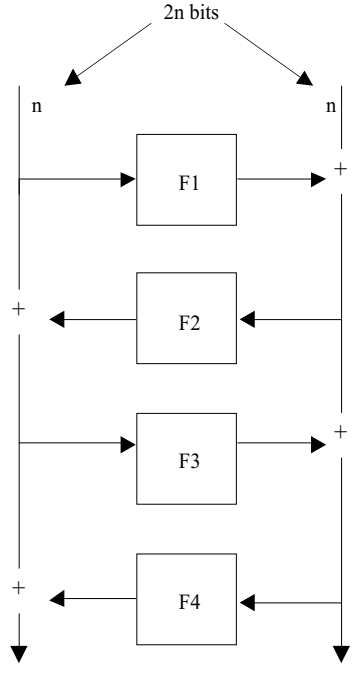

Figura 1. Encriptación Feistel de 2*n* bits con 4 funciones de *n* bits

[1] Matt Blaze.

¿Qué es NP-Completo?

La complejidad es una medida de la eficiencia de un algoritmo para dar solución a un problema; se ha dividido principalmente en dos clases: P y NP. La clase P corresponde a los problemas cuyos algoritmos de solución son de complejidad polinomial en tiempo, y los problemas NP son los problemas que hasta la fecha no han podido ser resueltos de manera exacta por medio de algoritmos deterministas eficientes, lo que significa que el algoritmo determinístico que solucione el problema, si existe, es de complejidad exponencial.

Los algoritmos han sido divididos como buenos o malos algoritmos. La comunidad computacional acepta que un buen algoritmo es aquel para el cual existe un algoritmo polinomial determinístico que lo resuelva. También se acepta que un mal algoritmo es aquel para el cual dicho algoritmo simplemente no existe.

Un problema se dice intratable si es muy difícil que un algoritmo de tiempo no polinomial lo resuelva.

TURTLE

Turtle es un cifrador con tamaño de bloque variable que usa una red Feistel de cuatro ciclos como primitiva de seguridad. Se define el pseudocódigo del bloque TURTLE para encriptar de la siguiente manera:

```
Turtle (X) = {
If Length (X) = w
                {
                n ← n+1
                return Fn(X)
                }
                else {
                        R ← Xright
                        L ← Xleft

                        R ← R ⊕ Turtle (L)
                        L ← L ⊕ Turtle (R)
                        R ← R ⊕ Turtle (L)
                        L ← L ⊕ Turtle (R)

                        Return R/L
                             }

}
```

HARE

Hare es un generador de flujo de llaves basado en el encriptamiento de un contador con el uso de la red Feistel de cuatro ciclos. Para ello utiliza dos conjuntos que representan, cada uno de ellos, los cuatro ciclos. El primer conjunto permanecerá activo y el otro inactivo, posteriormente cambiarán los papeles. El conjunto activo será usado para encriptar el contador, mientras que los elementos del inactivo son intercambiados. Cuando el cambio finiquita, el conjunto activo pasará a ser el inactivo y viceversa. La estructura del pseudocódigo sería:

```
Turtle (X) = {
If Length (X) = w
                {
                n ← n+1
                return Fn(X)
                }
                else {
                        R ← Xright
                        L ← Xleft

                        R ← R ⊕ Turtle (L)
                        L ← L ⊕ Turtle (R)
                        R ← R ⊕ Turtle (L)
                        L ← L ⊕ Turtle (R)

                        Return R/L
                }

}
```

IMPLEMENTACIÓN

Blaze propone un código en lenguaje C donde se encuentran las rutinas necesarias para la implementación del Turtle y Hare, además se encuentran los procesos de encriptación y desencriptación. Para todo esto utiliza las siguientes estructuras de datos principalmente:

Para el Turtle:

1. TK.
Una matriz de 256 x 256 así:

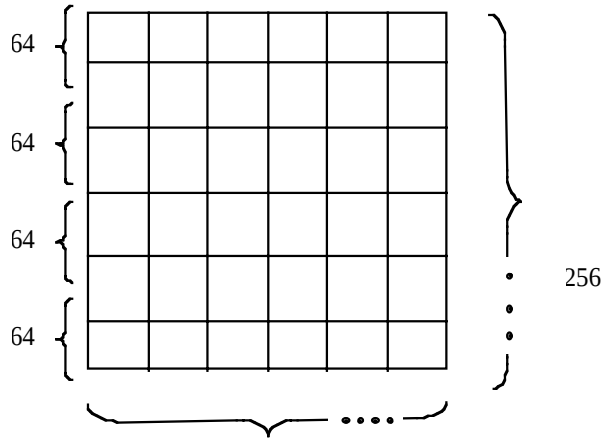

Figura 2. Estructura de datos utiliada para Turtle

Los bloques de 64 filas señalados anteriormente representan las cuatro funciones necesarias para la red Feistel de cuatro ciclos.

2. HK.
HK->sbox

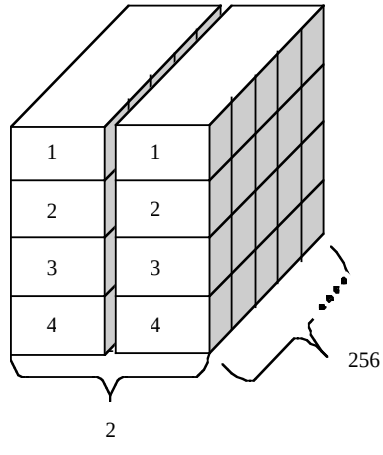

Figura 3. estructura de datos utilizada para Hare

Cada bloque representa los conjuntos activo e inactivo que son permutados.

Podemos observar que cada fila en un bloque también representa una función de la red Feistel.

PARAMETRIZACIÓN

La parametrización del proceso sería la siguiente:

→ Crea una clave turtle desde una clave pequeña:

```
TURTLEWORD unsigned char
Int turtle_key (shortkey,len,key,n)
Turtleword *shorkey
Int len
TK *Key
Int n
```

→ Genera una clave hare desde una clave pequeña, crea las tablas a partir del shortkey:

```
hare_key (shortkey, len, &harekey)
HK * harekey
Int len
Turtleword * shortkey
```

→ Genera *n* permutaciones sobre 2^wordbits; en el caso de la implementación wordbits=8, genera sbox de hare:

```
Keyperm( sbox, key, len, n)
TURTLEWORD sbox[][INTURTLEWORDS]
TURTLEWORD * key
Int len
Int n
```

→ Se basa también en la red Feistel; esta función crea flujos de claves hare y los coloca en sbox:

```
Turtleword hare_stream(key)
HK *key
```

→ Se basa en redes Feistel y es la que encripta el bloque; se apoya en r_turtle_encrypt:

```
Int turtle_encrypt (blk,key)
Turtleword *blk
TK *key
```

→ Aplica las primitivas Feistel en el orden contrario a turtle_encrypt para obtener el texto descifrado:

```
Int turtle_decrypt (blk,key)
Turtleword *blk
TK *key
```

→ Utiliza la tabla sbox para encriptar una palabra (1 byte):

```
Int turtle_encrypt (in, out, n, key)
Turtleword *in
Turtleword *out
Int n
TK *key
```

→ Aplica las primitivas Feistel en el orden contrario a turtle_encrypt para obtener el texto descifrado:

```
Int turtle_decrypt (blk,key)
Turtleword *blk
TK *key
```

En la figura 4 se muestra la estructura de llamado a las funciones en la implementación de los cifradores. A diferencia de otros cifradores que se basan en factorizar o encontrar logaritmos de números grandes, la solidez matemática de los cifradores estudiados está basada en redes Feistel y en el número y la complejidad de las funciones que integran esta red.

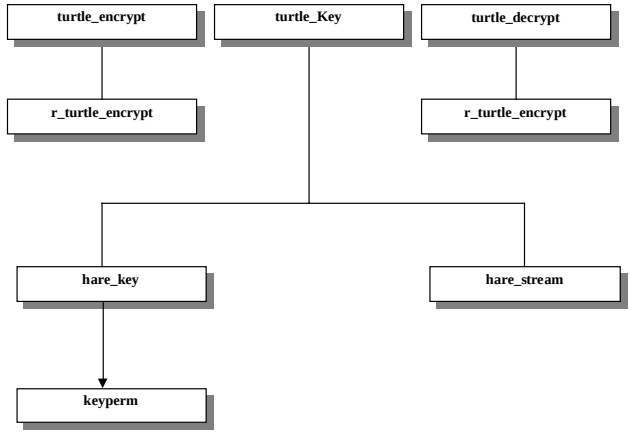

Figura 4. Esquematización del llamado a las funciones

CONCLUSIONES

Después de haber analizado los fundamentos del algoritmo, las estructuras de datos, funciones y su interrelación, hemos llegado a las siguientes conclusiones:

- A diferencia de otros cifradores que se basan en factorizar o encontrar logaritmos de números grandes, la solidez matemática del cifrador estudiado está basada en redes Feistel y en el número y la complejidad de las funciones que integran esta red.

- La solidez del algoritmo está en el problema de recuperar el estado interno para entradas y salidas dadas que es de complejidad NP-completo; esto significa que no hay un algoritmo determinístico de complejidad polinomial que lo resuelva, y los algoritmos que lo podrían resolver son de complejidad exponencial y requerirían muchos años o centurias de tiempo de cálculo para entradas moderadamente grandes, con la tecnología actual.

- La velocidad del algoritmo es muy buena, aproximadamente 60.000 bloques por segundo en un Pentium de 100 MHZ.

- El algoritmo crea un ciphertext diez veces mayor que el plaintext, lo cual proporciona mayor seguridad, aunque disminuye el rendimiento en tiempo.

- Para aplicaciones en redes se debe utilizar una implementación del algoritmo que cree un ciphertext del mismo tamaño del plaintext, esto aumentaría significativamente la velocidad del algoritmo y disminuiría el tráfico circulante en la red.

Terminología

- Criptología: Es el estudio de sistemas utilizados para las comunicaciones secretas. Se divide en Criptografía y Criptoanálisis.
- Criptografía: Ciencia que estudia el diseño de sistemas para la escritura secreta.
- Cifrador: Métodos por los cuales se hace secreta la escritura. En general, este término es utilizado para nombrar aquellas técnicas que aplican los procesos para el ciframiento y desciframiento de información.
- Ciframiento: Transformaciones especiales que operan sobre cualquier tipo de mensajes convirtiéndolo en una representación sin sentido del mismo.
- Ciphertext: Mensaje cifrado.
- Plaintext: Mensaje original.

- CRIPTOANÁLISIS: Ciencia que estudia los métodos y técnicas para determinar el plaintext o la llave desde el ciphertext, o determinar la llave desde el par plaintext_ciphertext.
- CRIPTOSISTEMA: Son todos los elementos que suministran la seguridad de una comunicación entre dos individuos que transmiten una información.
- DESCIFRAMIENTO: Proceso de transformar el ciphertext en plaintext.
- ALGORITMO DE CLAVE SIMÉTRICA: Algoritmo que utiliza la misma clave para encriptar y para desencriptar.
- CIFRADOR DE FLUJO: Algoritmo que encripta datos un byte a la vez.

Referencias

BLAZE, Matt. *Efficient Symmetric-Key Ciphers Based on an NP- Complete Subproblem.* AT&T Laboratories.

SCHILDT OSBORNE, Herbert. *Borland C++. Manual de Referencia.* Madrid, McGraw-Hill, 1997. 1118 p.

MARTÍNEZ, Hannia y VERBEL, Samuel. Análisis, diseño y desarrollo de un sistema criptográfico para un ambiente computacional. Tesis, Universidad del Norte, 1994.